III. B.Com. Computer Applications V-Semester
Course: 15 – Web Application Development Tools
Web Interface Designing using JavaScript & PHP
W.E.F AY:2025-2026

Sri Venkateswara University, Tirupati.

UNIT 1: Introduction of JavaScript

JavaScript is a popular programming language used by developers all over the world. It's a lightweight and easy-to-learn language that can run on both the Client-Side (in your browser) and the Server-Side (on the server). JavaScript was created in 1995 by Brendan Eich.

A Script is a simple language for to perform data validation within HTML forms. The Script languages are very helpful to create dynamic or interactive web pages. JavaScript supports Client Side as well as Server Side. The Scripts are executed by web browser. The web browsers convert the data into machine language by using interpreter that means the Script programming lines are executed line by line with the help of interpreter.

While using HTML forms the entered data are validation in the Client Side browser window itself. So the Script programs will fast process data between the Server and Client. There are many Script languages like JavaScript, PERL, VBScript, Live Script, Active Server Page (ASP), Java Server Page (JSP) etc.

Script Language: The Script Languages are mainly classified into two types:

1. Client-Side Script.
2. Server-Side Script.

1. **Client-Side Script:** The Client-Side Script validates the user inputs on HTML form (in your browser).

2. **Server-Side Script:** Where as Server-Side Script handles the received from input and responds required information to the Client (on the server).

History of JavaScript:

- JavaScript originates from a language called 'Live Script', the idea was to find a language which could be used to provide client side browser application, but which was not as complicated as Java.
- JavaScript is a fairly simple language which is only suitable for fairly simple tasks. The language is best suited to tasks which run for short time and is most commonly used to manipulate the pieces of the Document Object Model (DOM).
- Script can only manipulated objects on the page because of the Document Object Model (DOM). This was developed by the World Wide Web Consortium (W3C).
- Web pages which have embedded JavaScript and which must run on both of the major browsers.

Some of Listed Scripting Languages:

There are many alternative solutions to the problem of making web sites interactive and dynamic. Some of the scripting solutions which might be considered as competitors to JavaScript are listed below:

- **PERL (Practical Extraction Report Language):** PERL was developed by "Larry Wall" in 1987. A complex language that is commonly used for server-side CGI scripting. PERL is available for client-side work through a subset called PerlScript which can also be used when writing Active Server Pages (ASP).

- **JavaScript:** JavaScript was developed in 1995 by "Netscape" as a Server-Side Scripting Language that was originally called Live Script. Later Netscape & Sun Micro Systems formed an alias and jointly announced that Live Script would be renamed as "JavaScript". The JavaScript Syntax is similar to C++ and Java. But technically, functionally & behaviorally Java Script is very different from this language. Java Script supports both Server-Side and Client-Side Scripting Languages. These languages will be supported to any web browser.

- **VBScript and ASP:** VBScript is one of the Microsoft Technologies for creating Client & Server Side Script. In additional VBScript is the default scripting program used in Active Server Page (ASP). It run inside Internet Explorer (IE).

- **Python:** This language is still progressing in the area of writing CGI (Common Gateway Interface) by using these web browser, we can run Python Applet. In the few years python will boom in the area of Client-Side Scripting.

- **TCL (Transmission Control Language):** This has been a popular choice for system programming; we can easily downloading in TCL plug-in from the internet.

***** Benefits /Merits/ Advantages of JavaScript:** JavaScript has a number of big benefits to anyone who wants to make their web site dynamic.

- It is widely supported in Web Browsers.
- It gives easily access to the document objects and can manipulated most of them.
- JavaScript can give interesting animations without the long download times associated with many multimedia data types.
- Web surfers don't need a special plug-in to use your browser.
- JavaScript is relatively secure: JavaScript can neither read from your local hard drive nor write to it, and you can't get a virus infection directly from JavaScript.

Problems / Demerits / Disadvantages of JavaScript:

- If your script doesn't work then your page is useless.
- Most scripts rely upon manipulation the elements of the Document Object Model (DOM).
- Script can run slowly and complex scripts can take a long time to startup.
- JavaScript doesn't have any multithreading.
- It does not allow the reading or writing of files.
- It does not support networking applications.

(Q) Basic Concept of JavaScript / Skeleton of JavaScript / Structure of JavaScript?

JavaScript is designed to add the interactively or embedded in the HTML pages. JavaScript is very much similar to a programming language. JavaScript originates from a language called LiveScript. JavaScript is a platform independent and can be run everywhere. JavaScript is used for client-side programming. Let us write the first JavaScript by which some message will be displayed on the web page.

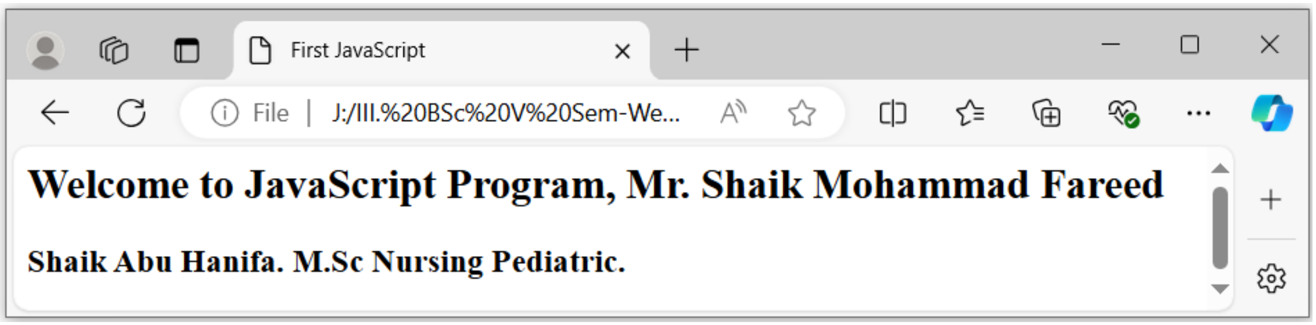
Skeleton / Structure of JavaScript:

```
<HTML>
    <title>
        . . . . . . . . .  }  Title Section.
    </title>
        <head>
                . . . . . . . . .  }  Head Section.
        </head>
                <body>
                      <script language = "JavaScript">
                                  . . . . . . . . .
                                  . . . . . . . . .
                                  . . . . . . . . .       }  JavaScript Code
                                  . . . . . . . . .
                                  . . . . . . . . .
                         </script>
                </body>
   </HTML>
```

Example: Display the message on the page by using JavaScript program?

```
<html>
<title> First JavaScript </title>
<body>
        <script language ="javascript">
            document.writeln("<h2> Welcome to JavaScript Program, Mr. Shaik Mohammad Fareed </h2>");
        </script>
        <h3> Shaik Abu Hanifa. M.Sc Nursing Pediatric. </h3>
</body>
</html>
```

Output:



Welcome to JavaScript Program, Mr. Shaik Mohammad Fareed

Shaik Abu Hanifa. M.Sc Nursing Pediatric.

How to Add JavaScript in HTML Document?

To add JavaScript in HTML document, several methods can be used. These methods include embedding JavaScript directly within the HTML file or linking an external JavaScript file.

1. JavaScript Code Inside <head> Tag: Placing JavaScript within the <head> section of an HTML document ensures that the script is loaded and executed as the page loads. This is useful for scripts that need to be initialized before the page content is rendered.

Example:

```
<html>
        <head>
                <script>
                        - - - - - - - - -
                        - - - - - - - - -        Within Head Section
                        - - - - - - - - -
                </script>
        </head>
        <body>
                - - - - - - - - -
                - - - - - - - - -
        </body>
</html>
```

2. JavaScript Code Inside <body> Tag: JavaScript can also be placed inside the <body> section of an HTML page. Typically, scripts placed at the end of the <body> load after the content, which can be useful if your script depends on the DOM (Document Object Model) being fully loaded.

Example:

```
<html>
        <head> - - - - - - - - - </head>
                <body>
                        - - - - - - - - -
                        - - - - - - - - -
                        <script>
                                - - - - - - - - -        Within Body Section
                                - - - - - - - - -
                                - - - - - - - - -
                        </script>
                </body>
        </html>
```

3. External JavaScript (Using External File): For larger projects or when reusing scripts across multiple HTML files, you can place your JavaScript code in an external ".js" file. This file is then linked to your HTML document using the src attribute within a <script> tag.

Example:

```
<html>
        <head>
                <script src = filename.js>
                        - - - - - - - - -
                        - - - - - - - - -        Within Head Section
                        - - - - - - - - -
                </script>
```

```
                    </head>
                    <body>
                              - - - - - - - - - -
                              - - - - - - - - - -
                    </body>
          </html>
```

Basics / Importance of JavaScript:

JavaScript is one of the 3 languages all web developers:

1. HTML (Hyper Text Markup Language) to define the content of web pages
2. CSS (Cascading Style Sheets) to specify the layout of web pages
3. JavaScript to program the behavior of web pages

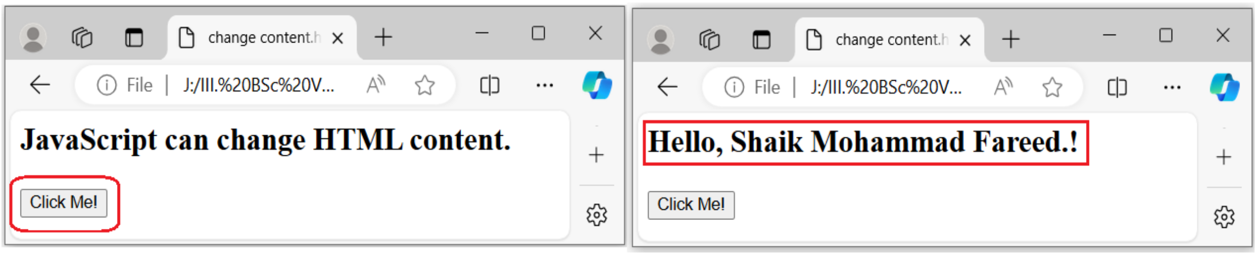
(1). JavaScript Can Change HTML Content: One of many JavaScript HTML methods is getElementById(). Finds an HTML element (with id = "shaik"), and changes the element content (innerHTML) to "Hello Shaik Mohammad Fareed.!"

Example: document.getElementById("shaik").innerHTML = "Hello Shaik Mohammad Fareed.!";

NOTE: JavaScript accepts both double and single quotes.

```
<html>
<body>
         <h2> <p id = "shaik">JavaScript can change HTML content.</p> </h2>
         <button type="button" onclick='document.getElementById("shaik").innerHTML = "Hello, Shaik
           Mohammad Fareed.!"'>Click Me! </button>
</body>
</html>
```

Output:



JavaScript can change HTML content.

Click Me!

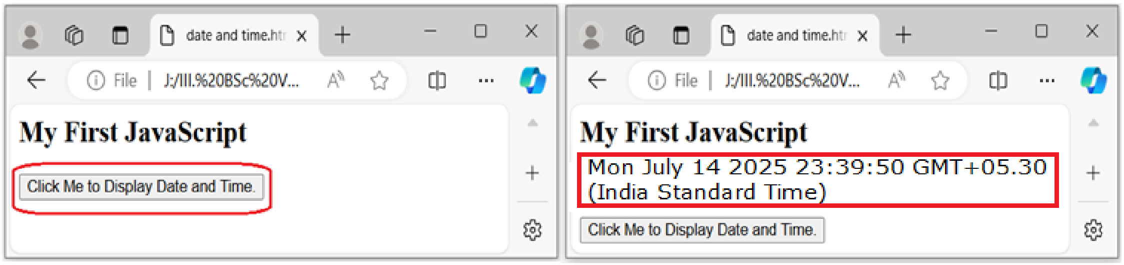
Hello, Shaik Mohammad Fareed.!

Click Me!

Example:

```
<html>
<body>
         <h2>Display Date & Time </h2>
         <p id = "shaik"></p>
         <button type="button" onclick="document.getElementById('shaik').innerHTML = Date()">
         Click Me to Display Date and Time.</button>
</body>
</html>
```

Output:



My First JavaScript

Click Me to Display Date and Time.

My First JavaScript

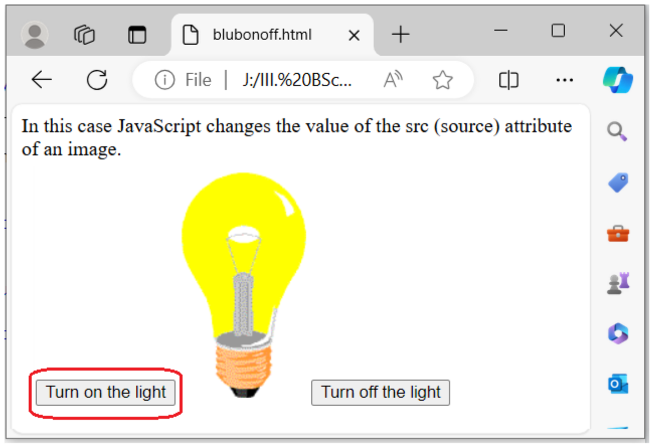
Mon July 14 2025 23:39:50 GMT+05.30 (India Standard Time)

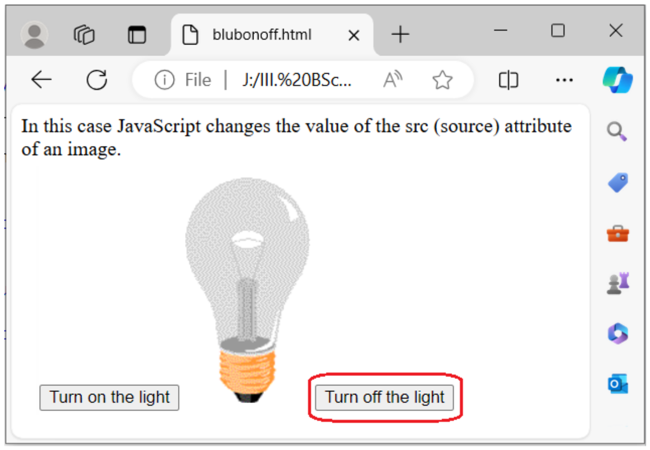
Click Me to Display Date and Time.

(2). JavaScript Can Change HTML Attribute Values: JavaScript changes the value of the src (source) attribute of an <img> tag:

```
<html>
<body>
        <p> In this case JavaScript changes the value of the src (source) attribute of an image.</p>
        <button onclick="document.getElementById('myImage').src="J:\III. B.Com V Sem Web Application
          Development Tools Notes 2025\bulb_on.jpg"> Turn on the light </button>
        <img id="myImage" src="pic_bulboff.gif" style="width:100px">
        <button onclick="document.getElementById ('myImage').src="J:\III. B.Com V Sem Web Application
          Development Tools Notes 2025\bulb_off.jpg">Turn off the light </button>
</body>
</html>
```

Output:


In this case JavaScript changes the value of the src (source) attribute of an image.

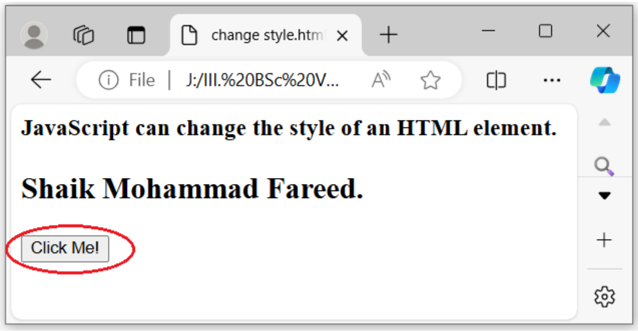

In this case JavaScript changes the value of the src (source) attribute of an image.

(3). JavaScript Can Change HTML Styles (CSS): Changing the style of an HTML element, is a variant of changing an HTML attribute.

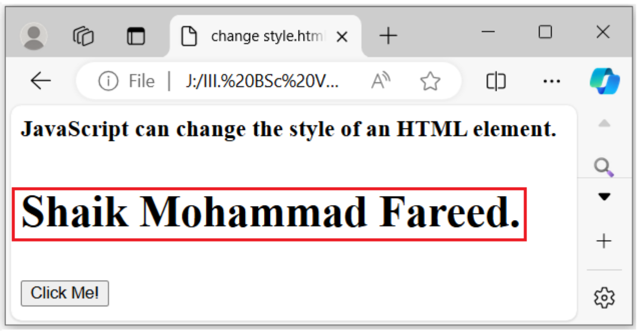
Example: document.getElementById("demo").style.fontSize = "35px";

```
<html>
<body>
   <h3> JavaScript can change the style of an HTML element. </h3>
   <h2>  <p id="demo">Shaik Mohammad Fareed.</p> </h2>
   <button type="button" onclick="document.getElementById('demo').style.fontSize='35px'"> Click Me!
</button>
</body>
</html>
```

Output:


JavaScript can change the style of an HTML element.

Shaik Mohammad Fareed.


JavaScript can change the style of an HTML element.

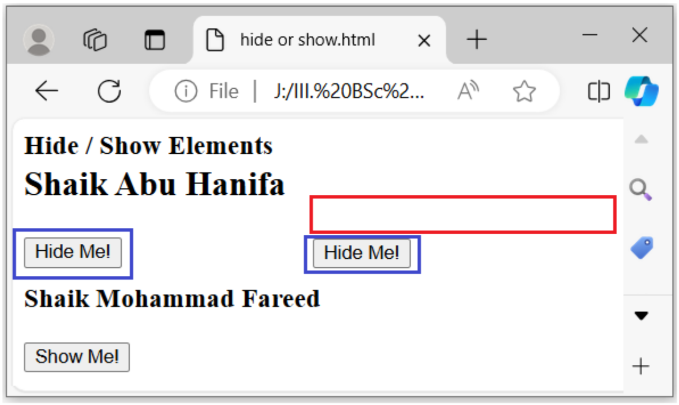
Shaik Mohammad Fareed.

(4). JavaScript Can Hide / Show HTML Elements: Hiding / Showing hidden HTML elements can be done by changing the display style: none / block;

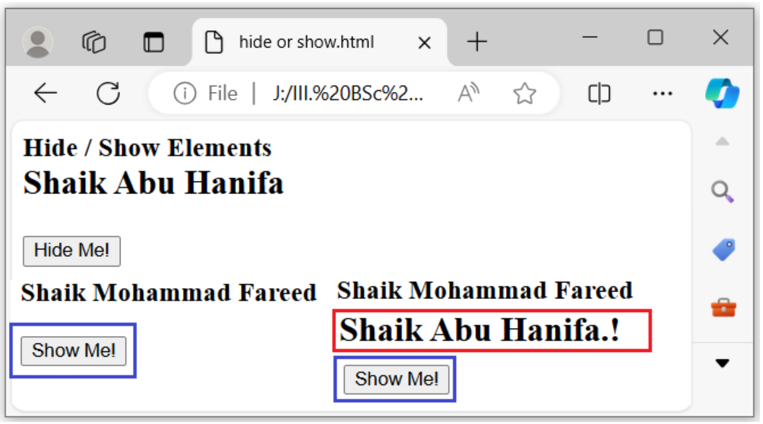
Example: document.getElementById("shaik").style.display = "none"/ "block";

```
<html>
<body>
   <h3> Hide / Show Elements </h3>
   <h2> <p id="shaik"> Shaik Abu Hanifa </p> </h2>
        <button type="button" onclick="document.getElementById('shaik').style.display='none'">
          Hide Me! </button>
    <h3> <p> Shaik Mohammad Fareed </p> </h3>
    <h2> <p id="smf" style="display:none">Shaik Abu Hanifa.! </p> </h2>
          <button type="button" onclick="document.getElementById('smf').style.display='block'">
            Show Me!</button>
</body>
</html>
```

Output:



```
Hide / Show Elements
Shaik Abu Hanifa
Hide Me!                    Hide Me!
Shaik Mohammad Fareed
Show Me!
```



```
Hide / Show Elements
Shaik Abu Hanifa
Hide Me!
Shaik Mohammad Fareed     Shaik Mohammad Fareed
                          Shaik Abu Hanifa.!
Show Me!                  Show Me!
```

(Q) Explain about JavaScript Output?

JavaScript Output: JavaScript Display Possibilities. JavaScript can display data in different ways:

- Writing into an HTML element, using innerHTML or innerText.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().

(1). Using innerHTML: To access an HTML element, JavaScript can use the document.getElementById(id) method. The id attribute defines the HTML element. The innerHTML property defines the HTML content:

Example: <p id="shaik"> </p>
    <script language="javascript"> document.getElementById("shaik").innerHTML = 10 + 20; </script>

Output: 30

(2). Using document.write(): For testing purposes, it is convenient to use document.write():

Example: <script language="javascript"> document.write (10 + 20); </script>

Output: 30

(3). Using window.alert(): You can use an alert box to display data:

Example: <script language="javascript"> window.alert (10 + 20); </script>

Output: 30

NOTE: In JavaScript, the window object is the global scope object. This means that variables, properties, and methods by default belong to the window object. This also means that specifying the window keyword is optional:

Example: <script language="javascript"> alert (10 + 20); </script>

Output: 30

IMP**(Q) How to define JavaScript Variables?**

These are data items that we can store and manipulated data as the program runs. Variables are Containers for Storing Data.

Rules of Variable Names: There are strict rules governing how you name your variables in JavaScript.

- The Variable Names must begin with an alphabetical letter or the underscore (_).
- Space can't be used in Variable Name.
- Names in JavaScript are case-sensitive; that means upper case and lower case are distinguished.
- You can't use reserved words as a variable name.

JavaScript Variables can be declared in 4 ways:

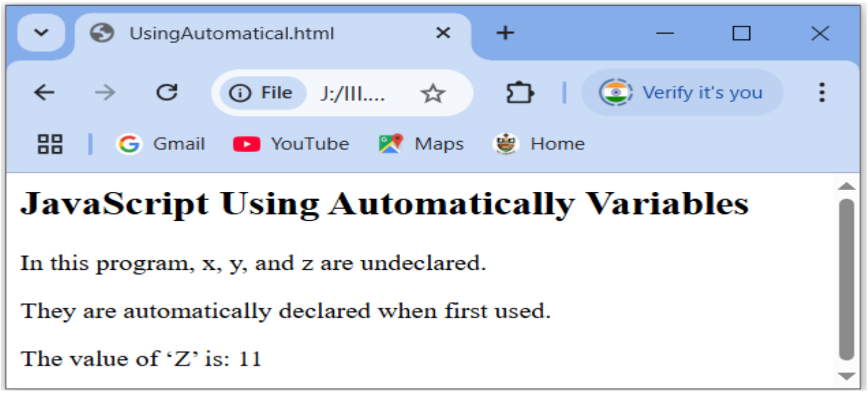
1. Automatically
2. Using var
3. Using let
4. Using const

Note:
- The var keyword was used in all JavaScript code from 1995 to 2015.
- The let and const keywords were added to JavaScript in 2015.
- The var keyword should only be used in code written for older browsers.

(I). Using Automatically Declaration:

```
<html>
<body>
        <h2> JavaScript Using Automatically Variables </h2>
                <p> In this program, x, y, and z are undeclared. </p>
                <p> They are automatically declared when first used. </p>
        <p id="shaik"> </p>
        <script>
                x = 5;
                y = 6;
                z = x + y;
                document.getElementById("shaik").innerHTML = "The value of 'Z' is: " + z;
        </script>
</body>
</html>
```

Output:



JavaScript Using Automatically Variables

In this program, x, y, and z are undeclared.

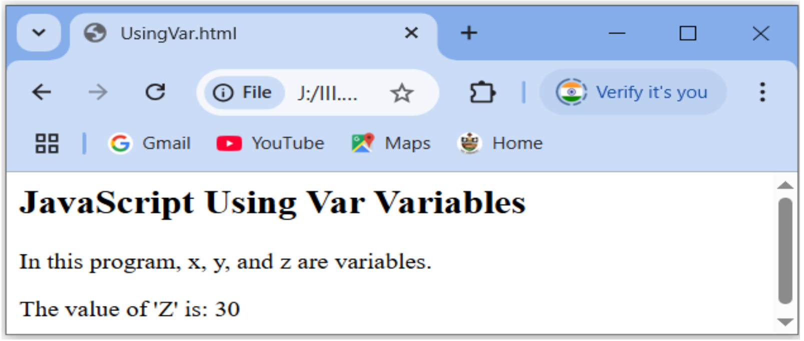
They are automatically declared when first used.

The value of 'Z' is: 11

(II) Using Var Declaration:

```
<html>
<body>
        <h2> JavaScript Using Var Variables </h2>
                <p> In this program, x, y, and z are variables. </p>
        <p id="SMF"> </p>
        <script>
                var x = 10;
                var y = 20;
                var z = x + y;
                document.getElementById("SMF").innerHTML = "The value of 'Z' is: " + z;
        </script>
</body>
</html>
```

Output:



JavaScript Using Var Variables

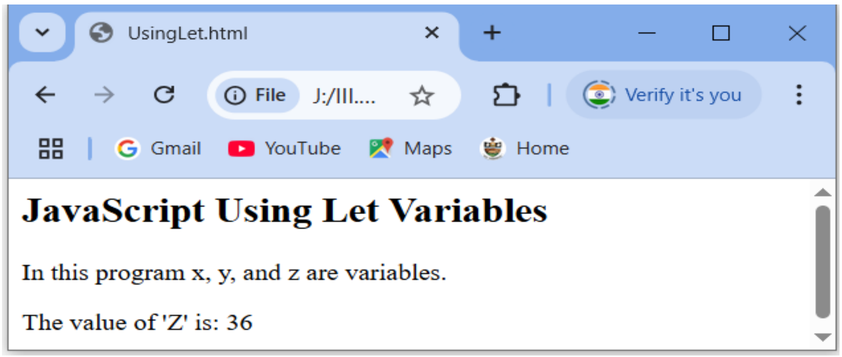
In this program, x, y, and z are variables.

The value of 'Z' is: 30

(III). Using Let Declaration:

```
<html>
<body>
        <h2> JavaScript Using Let Variables </h2>
                <p> In this program x, y, and z are variables. </p>
        <p id ="fareed"> </p>
        <script>
                let x = 10;
                let y = 26;
                let z = x + y;
                document.getElementById("fareed").innerHTML = "The value of 'Z' is: " + z;
        </script>
</body>
</html>
```

Output:



JavaScript Using Let Variables

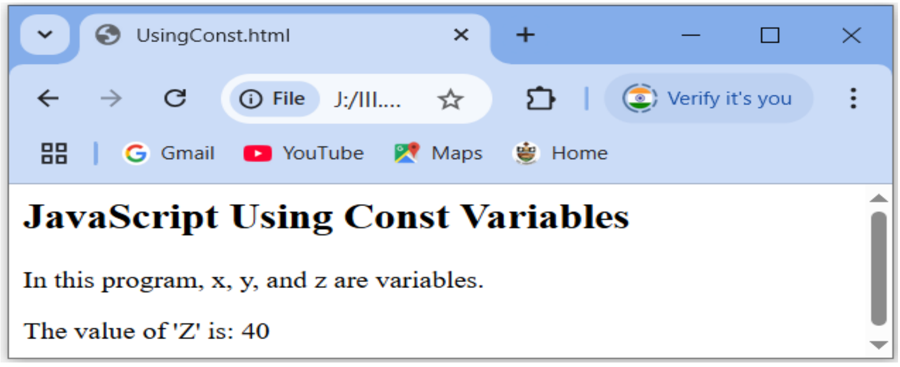
In this program x, y, and z are variables.

The value of 'Z' is: 36

(IV). Using Const Declaration:

```
<html>
<body>
        <h2> JavaScript Using Const Variables </h2>
                <p>In this program, x, y, and z are variables.</p>
        <p id="SMF"></p>
        <script>
                const x = 15;
                const y = 25;
                const z = x + y;
                document.getElementById("SMF").innerHTML = "The value of 'Z' is: " + z;
        </script>
</body>
</html>
```

Output:



JavaScript Using Const Variables

In this program, x, y, and z are variables.

The value of 'Z' is: 40

When to Use var, let, or const?

1. Always declare variables
2. Always use const if the value should not be changed
3. Always use const if the type should not be changed (Arrays and Objects)
4. Only use let if you can't use const
5. Only use var if you MUST support old browsers.

(Q) Explain about JavaScript Statements?

JavaScript statements are programming instructions that a computer executes. A computer program is essentially a list of these "instructions" designed to perform tasks. In a programming language, such instructions are called statements.

Types of Statements:

1. Variable Declarations (var, let, const): In JavaScript, you can declare variables using var, let, or const. let and const are used in modern JavaScript for block-scoped variables, while var is function-scoped and generally considered outdated.

Examples:
```
let name = "Mohammad Fareed";   // Declaration with 'let'
const age = 39;                 // Declaration with 'const' (constant value)
var isActive = true;            // Declaration with 'var' (older version)
```

2. Assignment Statement: An assignment statement is used to assign a value to a variable. You can assign any type of value, including strings, numbers, and objects, using the assignment operator =.

Example:
```
let number = 9491202987;          // Assigning a number
let message = "Hello, Fareed!";   // Assigning a string
```

3. Expression Statements: An expression in JavaScript is any valid unit of code that resolves to a value. When an expression is executed, it can perform an action (e.g., calculation, function call) and return a result. An expression can also be used as a statement.

Example:
```
x = 20 + 10;        // Expression statement
console.log(x);     // Expression statement using a function call
```

4. Control Flow Statements: Control flow statements are used to control the order in which statements are executed in a program. Examples include if, else, switch, while, and for loops.

Example:
```
let number = 40;
if (number > 10)
{
  console.log("Number is greater than 10");
}
```

5. Function Declarations: A function declaration is a statement that defines a function in JavaScript. Functions are reusable blocks of code designed to perform specific tasks.

Example:
```
function wish(myname)
{
  return "Hello, " + myname;
}
console.log(wish("Mohammad Fareed"));
```

6. Return Statement: The return statement is used to exit a function and optionally pass a value back to the calling code.

Example:

```
function add(x, y)
{
  return x + y;
}
let result = add(5, 3);         // output will be 8
```

7. Throw Statement: The throw statement is used to create custom errors in JavaScript. It is often used in conjunction with try...catch to handle errors.

Example:

```
function checkAge(age)
{
  if (age < 18)
       {
         throw new Error("Age must be 18 or older");
       }
}
```

8. Try...Catch Statement: The try...catch statement is used to handle exceptions in JavaScript. The code inside the try block is executed, and if an error occurs, the code inside the catch block will handle the error.

Example:

```
try
  {
       let result = someUndefinedFunction();      // This will throw an error
       console.log(result);
  }
 catch (error)
       {
         console.error("An error occurred:", error.message);
       }
```

9. Break and Continue Statements: The break and continue statements are used within loops. break exits the loop, while continue skips to the next iteration.

Example:

```
for (let i = 0; i < 10; i++)
{
  if (i == 5)
       {
       break;           // Exits the loop when i equals 5
       }
    console.log(i);
}
```

(Q) Explain about JavaScript Comments?

JavaScript Comments: JavaScript comments can be used to explain JavaScript code, and to make it more readable. JavaScript comments have two types:

1. Single Line Comment
2. Multi-line Comment

1. Single Line Comment: Single line comments start with double slashes (//). Any text between // and the end of the line will be ignored by JavaScript (will not be executed). Uses a single-line comment before each code line.

Example: let phone = 9491202987;   // Phone Number: 9491202987

2. Multi-line Comment: Multi-line comments start with /* and end with */. Any text between /* and */ will be ignored by JavaScript. Uses a multi-line comment (a comment block).

Example: let name; /* This name variable
Contain myname */

(Q) Define Keywords in JavaScript?

Keywords are pre-defined or reserved words which has some specific meanings in the language. They should not be used as variable names. JavaScript also provide a number of keywords which helps us to develop browser applications. Some are listed below.

abstract	boolean	break	byte	case
catch	char	class	continue	default
Do	double	else	extends	false
final	finally	float	for	if
implements	import	instanceof	int	interface
long	native	new	null	package
private	protected	public	return	short
static	super	switch	synchronized	this
throw	throws	transient	true	try
void	volatile	while		

VIMP*****(Q) Define a Data Types in JavaScript?**

Data Types: In JavaScript, variables can receive different data types over time. The latest JavaScript standard defines eight data types out of which six data types are Primitive (predefined) and two data types are Non-Primitives (complex).

(I). Primitive Data Types: The predefined data types provided by JavaScript language are known as primitive data types. Primitive data types are also known as in-built data types.

1. **Number:** JavaScript numbers are always stored in double-precision 64-bit binary format IEEE 754. Unlike other programming languages, you don't need int, float, etc to declare different numeric values.
   Example: var x = 20; or var y = 20.5;

2. **String:** A string (or a text string) is a series of characters. Strings are written with quotes, you can use single or double quotes.

   Example: let name = "Shaik Mohammad Fareed";

3. **Boolean:** Represent a logical entity and can have two values: true or false.
   Example: let x = true;

4. **Null:** This type has only one value that is ***null.***

   Example: let x = “NULL”;

5. **Undefined:** A variable without a value has a value undefined. The value is also undefined.

   Example: let car = undefined;

6. **BigInt:** All JavaScript numbers are stored in a 64-bit floating-point format.

   Example: let x = BigInt(“9491202987”);

NOTE: Whole numbers (integers): byte (8-bit), short (16-bit), int (32-bit), long (64-bit).
Real numbers (floating-point): float (32-bit), double (64-bit).
JavaScript numbers are always one type: double (64-bit floating point).

(II). Non-Primitive Data Types: The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.

1. **Objects:** JavaScript objects are written with curly braces {}. Object properties are written as name:value pairs, separated by commas.

   Example: const person = {firstName:"Mohammad", lastName:"Fareed", age:39, eyeColor:"blue"};

   NOTE: The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.

2. **Arrays:** JavaScript arrays are written with square brackets. Array items are separated by commas.

   Example: const cars = ["Ertiga", "Volvo", "BMW"];

   The following code declares (creates) an array called cars, containing three items (car names):

   NOTE: Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

VIMP********* (Q) What Operator? Explain various operators in JavaScript?**

In JavaScript, an operator is a symbol that performs an operation on one or more operands, such as variables or values, and returns a result. Let us take a simple expression (4 + 5) is equal to 9. Here 4 and 5 are called operands, and '+' is called the operator.

JavaScript supports the following types of operators:

1. Arithmetic Operators
2. Comparison Operators
3. Logical (or Relational) Operators
4. Bitwise Operators
5. Assignment Operators
6. String Operators
7. Ternary Operators

(1). JavaScript Arithmetic Operators: The JavaScript arithmetic operators are used to perform mathematical calculations such as addition, multiplication, subtraction, division, etc. on numbers. JavaScript supports the following arithmetic operators.

Example: Assume variable x holds 10 and variable y holds 20, then:

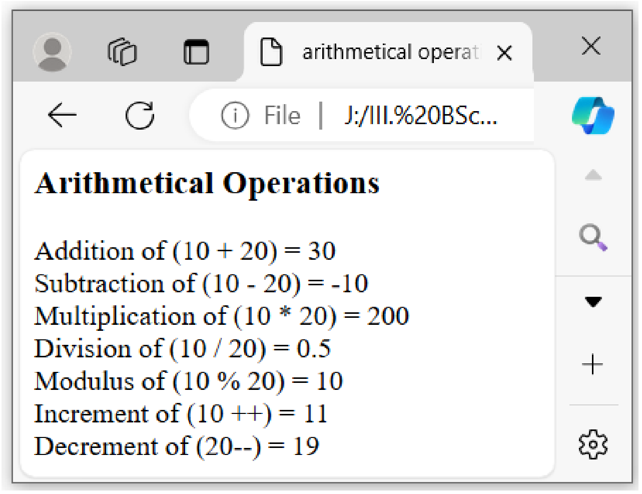
Arithmetic Operators	Description	Example
+ (Addition)	Adds two operands.	(10 + 20) = 30.
- (Subtraction)	Subtracts the second operand from the first.	(10 - 20) = -10.
* (Multiplication)	Multiplies both operands.	(10 * 20) = 200.
/ (Division)	Divides the numerator by the denominator.	(10 / 20) = 2.
% (Modulus)	Outputs the remainder of an integer division.	(10 % 20) = 0
++ (Increment)	Increases an integer value by one.	10++ = 11.
-- (Decrement)	Decreases an integer value by one.	10-- = 9.

Example:

```
<html>
<body>
 <h3> Arithmetical Operations </h3>
   <script>
     const x = 10; y = 20;
     var add = x + y;
     var sub = x - y;
     var mul = x * y;
     var div = x / y;
     var mod = x % y;
     var inc = x+1;
     var dec = y-1;
          document.write("Addition of (10 + 20) = " + add +"</br>");
          document.write("Subtraction of (10 - 20) = " + sub +"</br>");
          document.write("Multiplication of (10 * 20) = " + mul +"</br>");
          document.write("Division of (10 / 20) = " + div +"</br>");
          document.write("Modulus of (10 % 20) = " + mod +"</br>");
```

```
        document.write("Increment of (10 ++) = " + inc +"</br>");
        document.write("Decrement of (20--) = " + dec +"</br>");
    </script>
</body>
</html>
```

Output:



```
Arithmetical Operations

Addition of (10 + 20) = 30
Subtraction of (10 - 20) = -10
Multiplication of (10 * 20) = 200
Division of (10 / 20) = 0.5
Modulus of (10 % 20) = 10
Increment of (10 ++) = 11
Decrement of (20--) = 19
```

(2). JavaScript Comparison Operators: The JavaScript comparison operators compare two values and return a Boolean result (true or false). JavaScript supports the following comparison operators:

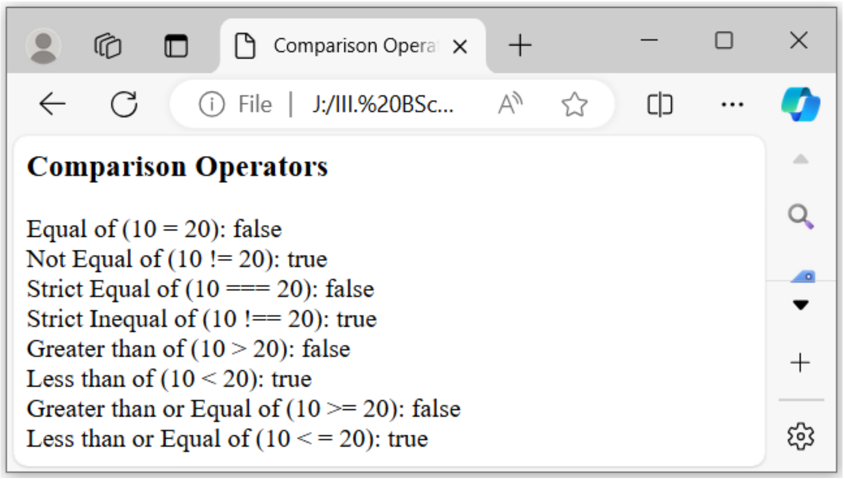
Example: Assume variable x holds 10 and variable y holds 20, then:

Operator	Description	Example
==	Equal	(10 == 20) is false
!=	Not Equal	(10 != 20) is true
===	Strict equality (equal value and equal type)	(10 === 20) is false
!==	Strict inequality (not equal value or not equal type)	(10 !== 20) is true
>	Greater than	(10 > 20) is false
<	Less than	(10 < 20) is true
>=	Greater than or Equal to	(10 >= 20) is false
<=	Less than or Equal to	(10 <= 20) is true

Example:

```
<html>
<body>
 <h3> Comparison Operators </h3>
   <script>
     const x = 10; y = 20;
     var equal = x == y;
     var notequal = x != y;
     var strictequal = x === y;
     var strictinequal = x !== y;
     var greater = x > y;
     var less = x < y;
     var greaterequal = x >= y;
     var lessequal = x <= y;
        document.write("Equal of (10 = 20): " + equal +"</br>");
        document.write("Not Equal of (10 != 20): " + notequal +"</br>");
         document.write("Strict Equal of (10 === 20): " + strictequal +"</br>");
         document.write("Strict Inequal of (10 !== 20): " + strictinequal +"</br>");
         document.write("Greater than of (10 > 20): " + greater +"</br>");
         document.write("Less than of (10 < 20): " + less +"</br>");
         document.write("Greater than or Equal of (10 >= 20): " + greaterequal +"</br>");
         document.write("Less than or Equal of (10 < = 20): " + lessequal +"</br>");
     </script>
</body>
</html>
```

Output:



```
Comparison Operators

Equal of (10 = 20): false
Not Equal of (10 != 20): true
Strict Equal of (10 === 20): false
Strict Inequal of (10 !== 20): true
Greater than of (10 > 20): false
Less than of (10 < 20): true
Greater than or Equal of (10 >= 20): false
Less than or Equal of (10 < = 20): true
```

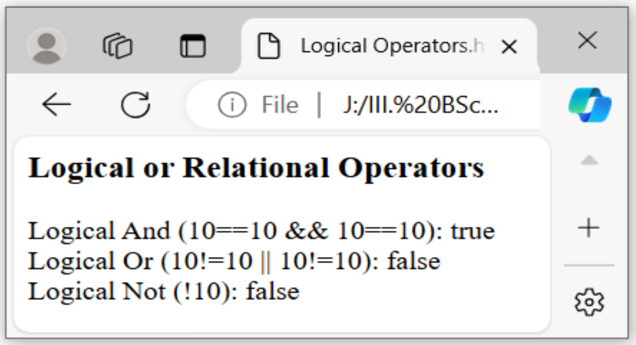
(3). JavaScript Logical or Relational Operators: The logical operators are generally used to perform logical operations on Boolean values. But logical operators can be applied to values of any types not only Boolean. JavaScript supports the following logical operators:

Example: Assume that the value of x is 10 and y is 0.

Logical Operators	Description	Example
&& (Logical AND)	If both the operands are non-zero, then the condition becomes true.	(x && y) is false
|| (Logical OR)	If any of the two operands are non-zero, then the condition becomes true.	(x || y) is true.
! (Logical NOT)	Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false.	!x is false.

Example:

```
<html>
<body>
 <h3> Logical or Relational Operators </h3>
   <script>
      const x = 10; y = 10;
      var logicaland = (x==y && x==y);
      var logicalor = (x!=y || y!=x)
      var logicalnot = !x;
         document.write("Logical And (10==10 && 10==10): " + logicaland +"</br>");
         document.write("Logical Or (10!=10 || 10!=10): " + logicalor +"</br>");
          document.write("Logical Not (!10): " + logicalnot +"</br>");
    </script>
</body>
</html>
```



```
Logical or Relational Operators

Logical And (10==10 && 10==10): true
Logical Or (10!=10 || 10!=10): false
Logical Not (!10): false
```

Output:

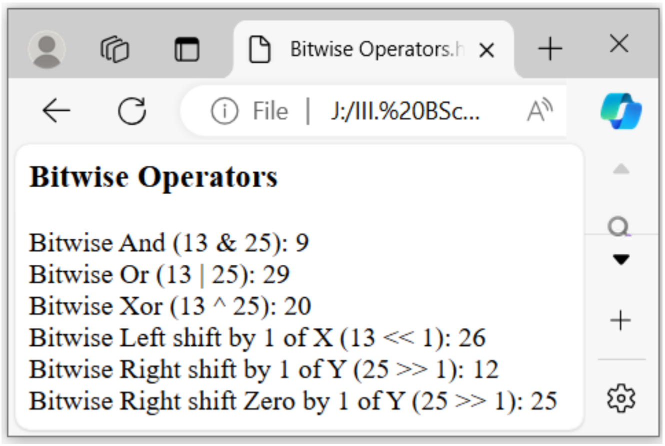
(4). JavaScript Bitwise Operators: The bitwise operators in JavaScript perform operations on the integer values at the binary level. They are used to manipulate each bit of the integer values. Bitwise operators are similar to logical operators but they work on individual bits. There are seven bitwise operators in JavaScript. Following is the list of bitwise operators with description.

Operator	Name	Description
&	Bitwise AND	Returns 1 if both bits are 1, otherwise 0.
|	Bitwise OR	Returns 1 if either bit is 1, otherwise 0.
^	Bitwise XOR	Returns 1 if both bits are different, otherwise 0.
! or ~	Bitwise NOT	Returns 1 if bit is 0, otherwise 0.
<<	Left Shift	Shifts the bits left by pushing zeros in from right and discarding leftmost bits.
>>	Right Shift	Shifts the bits right by pushing copies of leftmost bit in from left and discarding rightmost bits.
>>>	Right Shift with Zero	Shifts the bits right by pushing zeros in from left and discarding rightmost bits.

Example:

```
<html>
<body>
 <h3> Bitwise Operators </h3>
   <script>
      const x = 13; y = 25;
      var bitand = x & y;
      var bitor = x | y;
      var bitxor = x ^ y;
      var leftshift = x << 1;
      var rightshift = y >> 1;
      var rightshiftzero = y >>> 0;
          document.write("Bitwise And (13 & 25): " + bitand +"</br>");
          document.write("Bitwise Or (13 | 25): " + bitor +"</br>");
          document.write("Bitwise Xor (13 ^ 25): " + bitxor +"</br>");
          document.write("Bitwise Left shift by 1 of X (13 << 1): " + leftshift +"</br>");
          document.write("Bitwise Right shift by 1 of Y (25 >> 1): " + rightshift +"</br>");
          document.write("Bitwise Right shift Zero by 1 of Y (25 >> 1): " + rightshiftzero +"</br>");
    </script>
</body>
</html>
```

Output:



```
Bitwise Operators

Bitwise And (13 & 25): 9
Bitwise Or (13 | 25): 29
Bitwise Xor (13 ^ 25): 20
Bitwise Left shift by 1 of X (13 << 1): 26
Bitwise Right shift by 1 of Y (25 >> 1): 12
Bitwise Right shift Zero by 1 of Y (25 >> 1): 25
```

(5). JavaScript Assignment Operators: The assignment operators in JavaScript are used to assign values to the variables. These are binary operators. An assignment operator takes two operands, assigns a value to the left operand based on the value of right operand. The left operand is always a variable and the right operand may be literal, variable or expression.

Example:
```
let x = 10;         // right operand is a literal (value)
let y = x;          // right operand is a variable
let z = x + 10;     // right operand is an expression
```

An assignment operator first evaluates the expression and then assigns the value to the variable (left operand). A simple assignment operator is equal (=) operator. In the JavaScript statement "let x = 10;" the = operator assigns 10 to the variable x.

We can combine a simple assignment operator with other type of operators such as arithmetic, logical, etc. to get compound assignment operators. Some arithmetic assignment operators are +=, -=, *=, /=, etc. The += operator performs addition operation on the operands and assign the result to the left hand operand.

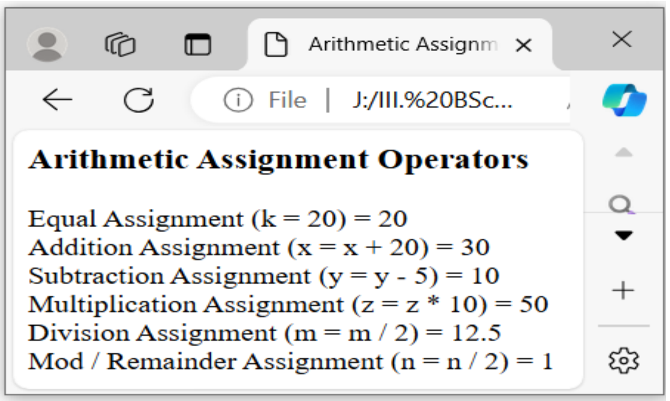
Arithmetic Assignment Operators: An arithmetic assignment operator performs arithmetic operation and assigns the result to a variable. Following is the list of operators with example:

Assignment Operator	Example	Equivalent To
= (Assignment)	a = b	a = b
+= (Addition Assignment)	a += b	a = a + b
-= (Subtraction Assignment)	a -= b	a = a – b
*= (Multiplication Assignment)	a *= b	a = a * b
/= (Division Assignment)	a /= b	a = a / b
%= (Remainder Assignment)	a %= b	a = a % b

Example:

```
<html>
<body>
 <h3> Arithmetic Assignment Operators </h3>
   <script>
        let k = 20;
        let x = 10; x += 20;
        let y = 15; y -= 5;
        let z = 5; z *= 10
        let m = 25; m /= 2;
        let n = 25; n %= 2;
            document.write("Equal Assignment (k = 20) = " + k +"</br>");
            document.write("Addition Assignment (x = x + 20) = " + x +"</br>");
            document.write("Subtraction Assignment (y = y - 5) = " + y +"</br>");
            document.write("Multiplication Assignment (z = z * 10) = " + z +"</br>");
            document.write("Division Assignment (m = m / 2) = " + m +"</br>");
            document.write("Mod / Remainder Assignment (n = n / 2) = " + n +"</br>");
    </script>
</body>
</html>
```

Output:



Arithmetic Assignment Operators

```
Equal Assignment (k = 20) = 20
Addition Assignment (x = x + 20) = 30
Subtraction Assignment (y = y - 5) = 10
Multiplication Assignment (z = z * 10) = 50
Division Assignment (m = m / 2) = 12.5
Mod / Remainder Assignment (n = n / 2) = 1
```

(6). JavaScript String Operators:

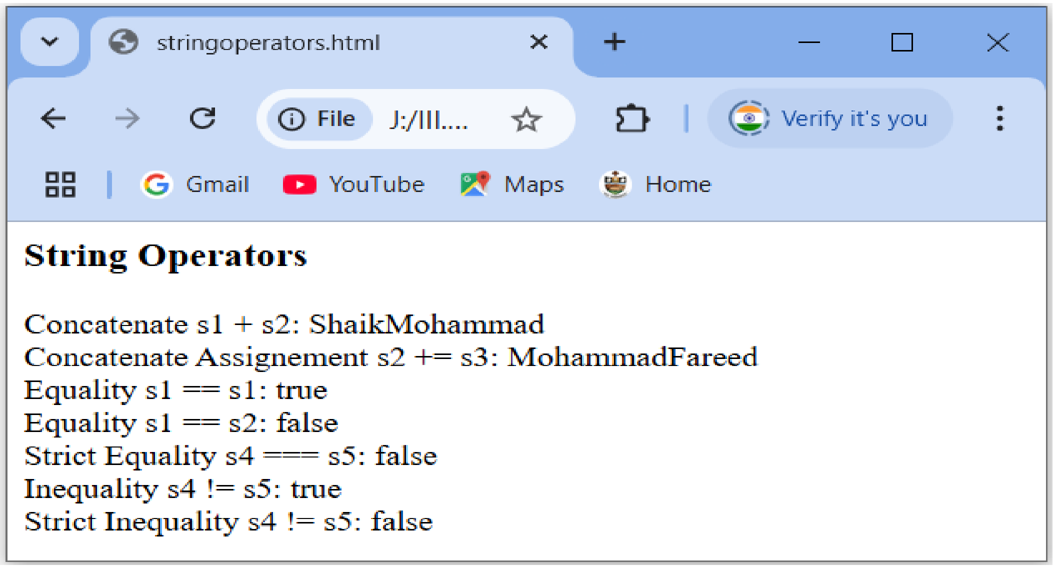
JavaScript String Operators are used to manipulate and perform operations on strings. There are two operators which are used to modify strings in JavaScript. These operators help us to join one string to another string.

String Operator	Explanation	Example
+ (Concatenate)	Combines two strings	s1 + s2
+= (Concatenate Assignment)	One string by adding content of another one to it	s1 += s2
== (Equality)	Checks if two strings are equal (ignoring type)	s1 == s2
=== (Strict Equality)	Checks if two stings are equal in value and type both	s1 === s2
!= (Inequality)	Checks if two strings are not equal (ignoring type)	s1 != s2

Example:

```
<html>
<body>
 <h3> String Operators </h3>
   <script>
         let s1 = "Shaik";
         let s2 = "Mohammad";
         let s3 = "Fareed";
         let s4 = "MCA";
         let s5 = "M.Sc";
                  document.write("Concatenate s1 + s2: " + (s1 + s2) + "</br>");
                  document.write("Concatenate Assignment s2 += s3: " + (s2 += s3) + "</br>");
                  document.write("Equality s1 == s1: " + (s1 == s1) + "</br>");
                  document.write("Equality s1 == s2: " + (s1 == s2) + "</br>");
                  document.write("Strict Equality s4 === s5: " + (s4 === s5) + "</br>");
                  document.write("Inequality s4 != s5: " + (s4 != s5) + "</br>");
                  document.write("Strict Inequality s4 != s5: " + (s4 !== s4) + "</br>");
    </script>
</body>
</html>
```

Output:



String Operators

```
Concatenate s1 + s2: ShaikMohammad
Concatenate Assignement s2 += s3: MohammadFareed
Equality s1 == s1: true
Equality s1 == s2: false
Strict Equality s4 === s5: false
Inequality s4 != s5: true
Strict Inequality s4 != s5: false
```

(7). JavaScript Ternary Operator: The Ternary Operator in JavaScript is a shortcut for writing simple if-else statements. It's also known as the Conditional Operator because it works based on a condition. The ternary operator allows you to quickly decide between two values depending on whether a condition is true or false.

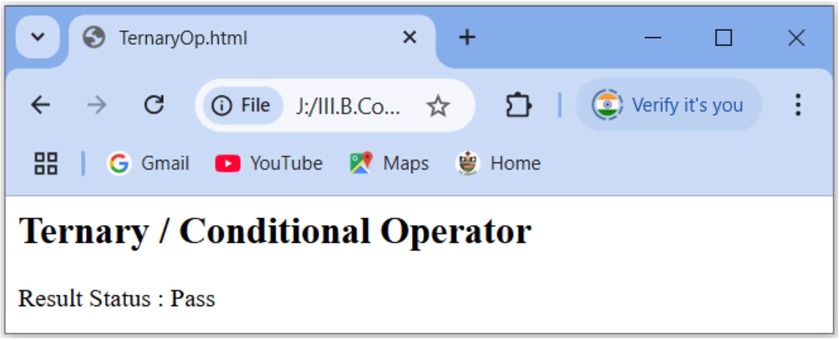
Syntax: condition ? trueExpression : falseExpression;

- **Condition:** A condition that evaluates to true or false.
- **expressionIfTrue:** The value or expression is returned if the condition is true.
- **expressionIfFalse:** The value or expression returned if the condition is false.

Example:

```
<html>
<body>
        <h2> Ternary / Conditional Operator </h2>
        <script>
                let PassMarks = 85;
                let res = (PassMarks > 40) ? "Pass" : "Fail";
                document.write("Result Status : " + res + "</br>");
        </script>
</body>
</html>
```

Output:



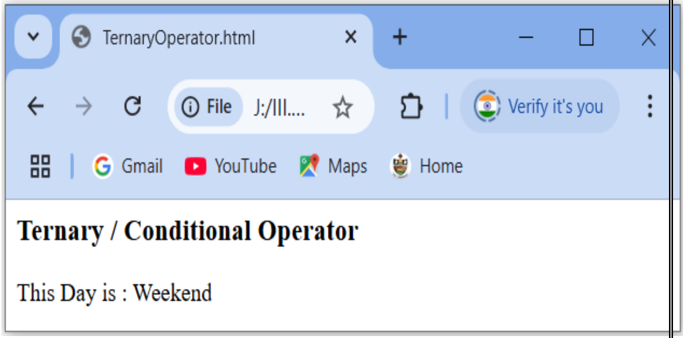
Ternary / Conditional Operator

```
Result Status : Pass
```

NOTE:

- We are checking if PassMarks is greater than 40.
- If the condition is true (PassMarks > 40), it returns "Pass".
- If false, it returns "Fail".

```
<html>
<body>
 <h3> Ternary / Conditional Operator </h3>
   <script>
        let day = 6;
        let greeting = (day === 1) ? 'Start of the week' :
                        (day === 2) ? 'Second day' :
                        (day === 3) ? 'Midweek' :
                        (day === 4) ? 'Almost weekend' :
                        'Weekend';
        document.write("This Day is : " + greeting);
    </script>
</body>
</html>
```



Ternary / Conditional Operator

```
This Day is : Weekend
```

Output:

VIMP*********(Q) Explain about various Conditional/Control Statements in JavaScript?**

JavaScript conditional statements allow you to execute specific blocks of code based on conditions. If the condition is met, a particular block of code will run; otherwise, another block of code will execute based on the condition.

They are 3 types of conditional/control statements in JavaScript:

(I). Branching / Decision / Selection Statements.
(II). Looping / Iterative Statements.
(III). Jumping / Transfer Statements.

(I). Types of Branching / Decision / Selection Conditional Statements:

1. if statement
2. if...else statement
3. Nested if…else statement
4. if...else if...else statement (if…else Ladder)
5. switch statement
6. ternary (conditional) operator

Big NOTE:

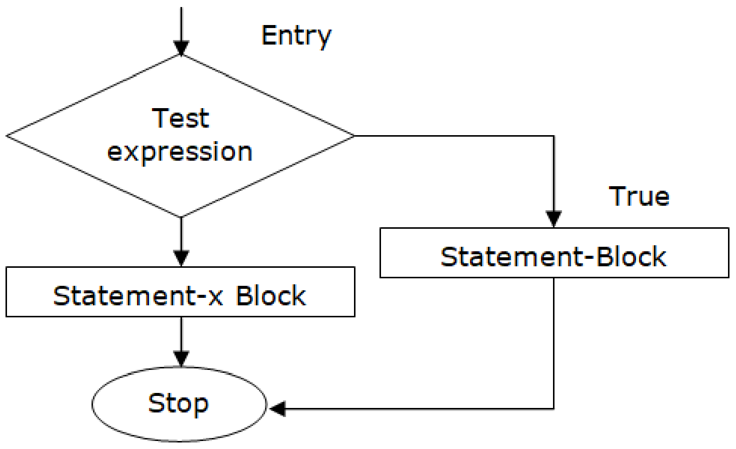
- Use 'if' to specify a block of code to be executed, if a specified condition is true
- Use 'else' to specify a block of code to be executed, if the same condition is false
- Use 'else if' to specify a new condition to test, if the first condition is false
- Use 'switch' to specify many alternative blocks of code to be executed

(1). if statement: The if statement evaluates a condition inside parentheses. If the given condition is "True", the block of code inside the curly braces {} runs. If it's false, it skips that block.

Syntax of if:

```
if (expression / condition)
  {
    Statement (s);        // to be executed if expression is true.
  }
    Statement-x;
```

Flow chart for simple if:

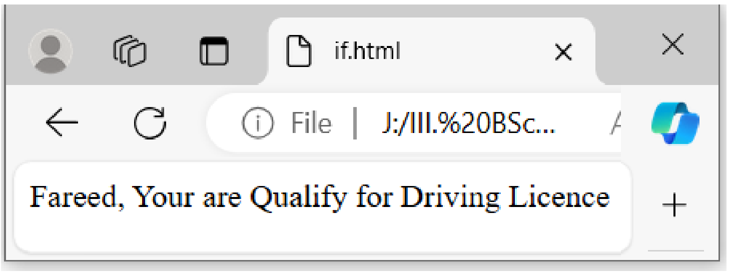


Example:

```
<html>
<body>
  <div id ='smf'> </div>
  <script type = "text/javascript">
    let result;
    let age = 20;
    if( age > 18 )
    {
      result = "Fareed, Your are Qualify for Driving Licence";
    }
        document.getElementById("smf").innerHTML = result;
  </script>
```

```
</body>
</html>
```

Output:



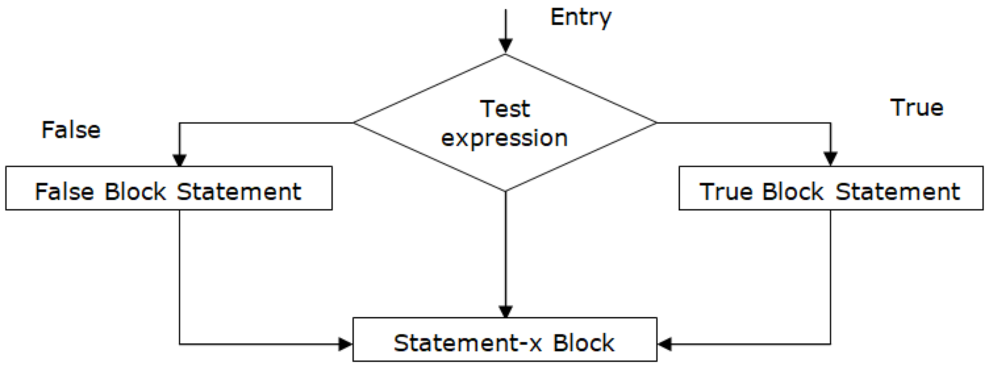
```
Fareed, Your are Qualify for Driving Licence
```

(2). if…else statement: The if…else statement is an extension of the if statement. If the test expression is true, then the true-block statement(s), immediately executed; otherwise, the false-block statements(s) are executed. In the both cases, the control is transferred subsequently to the statement-x.

Syntax of if…else statement:

```
if (expression / condition)
  {
        True–Block Statement(s);
  }
else
    {
        False–Block Statement(s);
    }
Statement-x;
```

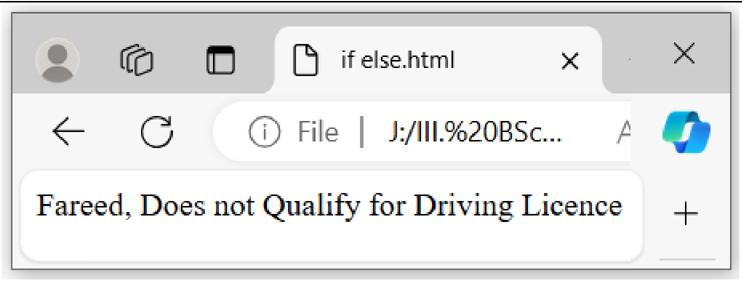
Flow chart for if…else statement:



Example:

```
<html>
<body>
  <div id ='shaik'> </div>
  <script type = "text/javascript">
        let result;
        let age = 15;
        if( age > 18 )
        {
                result = "Fareed, Your are Qualify for Driving Licence";
        }
        else
                {
                        result = "Fareed, Does not Qualify for Driving Licence";
                }
        document.getElementById("shaik").innerHTML = result;
  </script>
  </body>
</html>
```

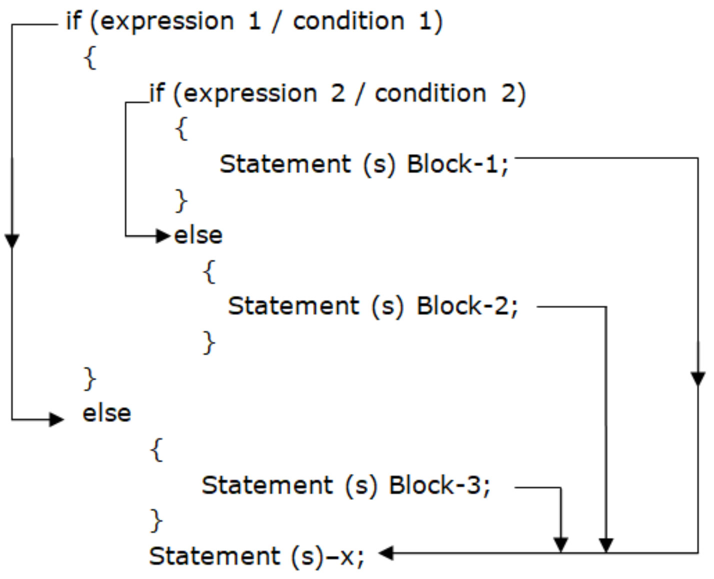
Output:



Fareed, Does not Qualify for Driving Licence

(3). Nested if...else statement: JavaScript allows us to nest if statements within if statements. i.e, we can place an if statement inside another if statement. A nested if is an if statement that is the target of another if or else.

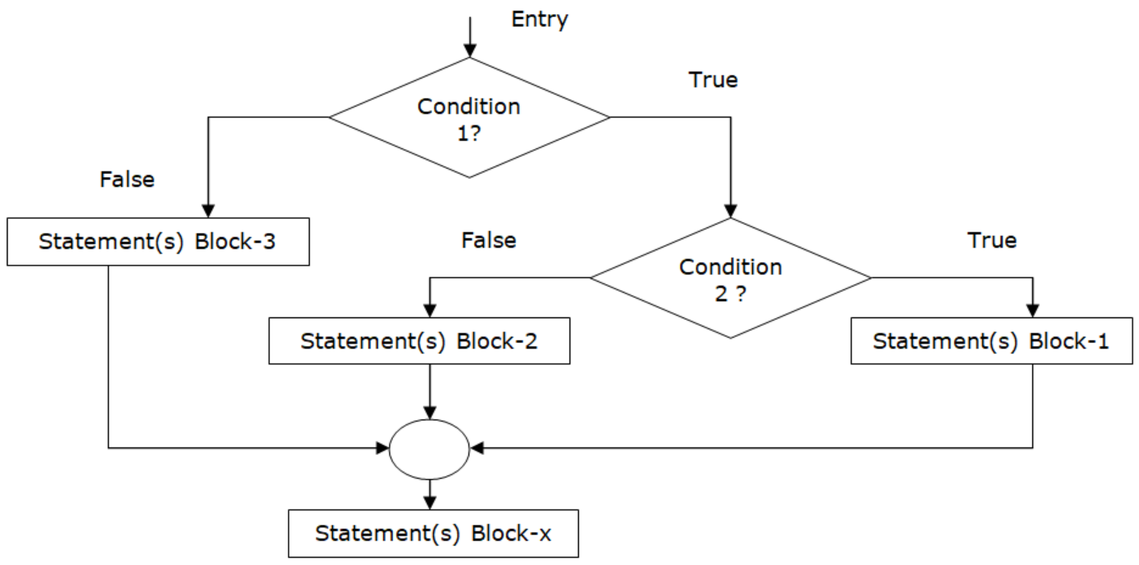
Syntax for Nested if...else statement:



```
if (expression 1 / condition 1)
{
    if (expression 2 / condition 2)
    {
        Statement (s) Block-1;
    }
    else
    {
        Statement (s) Block-2;
    }
}
else
{
    Statement (s) Block-3;
}
Statement (s)–x;
```

- if test condition 1 and Condition 2 are evaluated to "True", then Statement(s) Block-1 will executed.
- if test condition 1 evaluates to "True" and test condition 2 evaluates to "False", then Statement(s) Block-2 will be executed.
- if test condition 1 evaluates to "False", then the Statement(s) Block-3 will be executed.

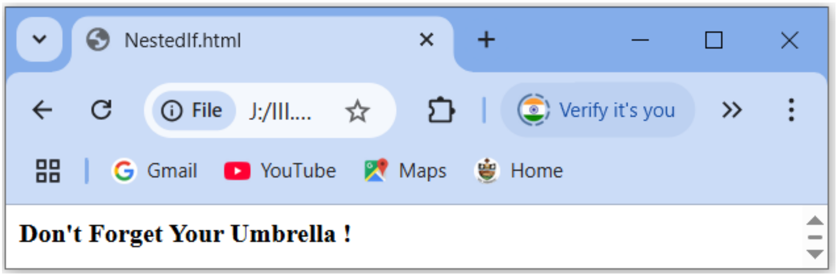
Flow Chart for Nested If...Else:



Example 1:

```
<html>
<body>
        <p id="shaik"></p>
        <script>
                let weather = "rainy";
                let temp = 20;
                if (weather === "sunny")
                 {
                        if (temp > 30)
                        {
                                document.writeln("<h4> It's a Hot Day! </h4>");
                        }
                        else if (temp > 25)
                           {
                                document.writeln("<h4> It's a Warm Day. </h4>");
                           }
                else
                        {
                                document.writeln("<h4> It's a Bit Cool Today. </h4>");
                        }
                }
                else if (weather === "rainy")
                        {
                                document.writeln("<h4> Don't Forget Your Umbrella ! </h4>");
                        }
                        else
                          {
                                document.writeln("<h4> Check the Weather Forecast!. </h4>");
                          };
</script>
</body>
</html>
```

Output:

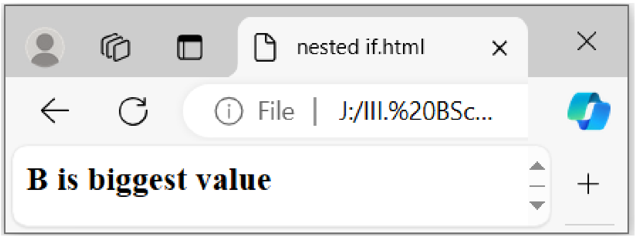


Don't Forget Your Umbrella !

Example 2:

```
<html>
<body>
<script>
        var a = 10;
        var b = 20;
        var c = 15;
        if(a>b)
        {
                if(a>c)
                        document.writeln("<h3> A is biggest value </h3>");
                else
                        document.writeln("<h3> C is biggest value </h3>");
        }
        else
                {
                        if (b>c)
                                document.writeln("<h3> B is biggest value </h3>");
                        else
                        document.writeln("<h3> C is biggest value </h3>");
                }
</script>
</body>
</html>
```

Output:

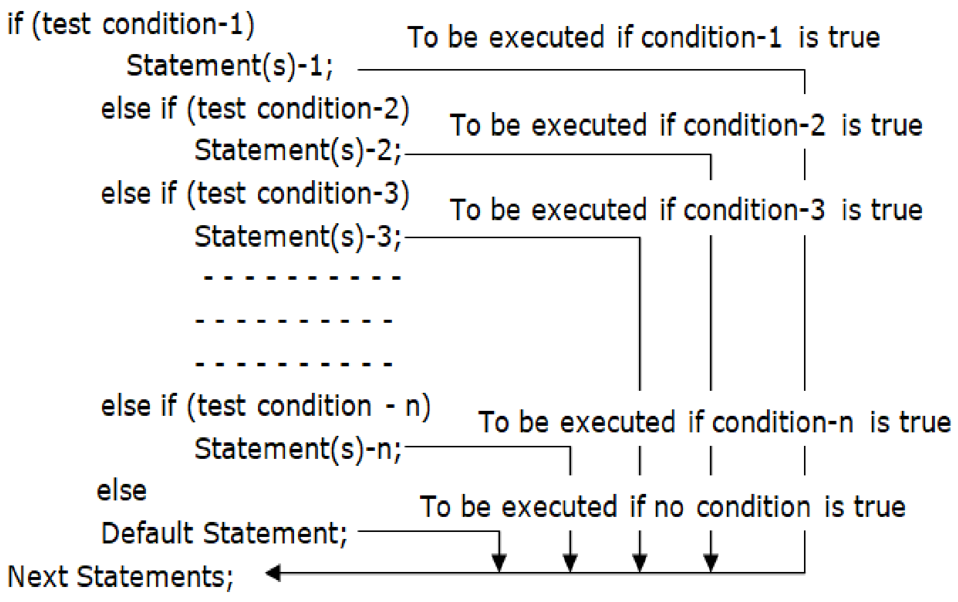


B is biggest value

(4). if…else…if ladder: The if…else…if ladder is used when we compare more than one condition to execute one set of statements among several mutually exclusive statements. The conditions are evaluated from the top to downwards.

- The statement which is associated with the condition that is evaluated to "True" will be executed and all the remaining statements will be skipped.
- Whenever it finds a "True" condition then the execution will be stopped there and control comes out from the ladder.
- When all the 'n' conditions becomes False, then the final 'else' containing the 'default-statement' will be executed.

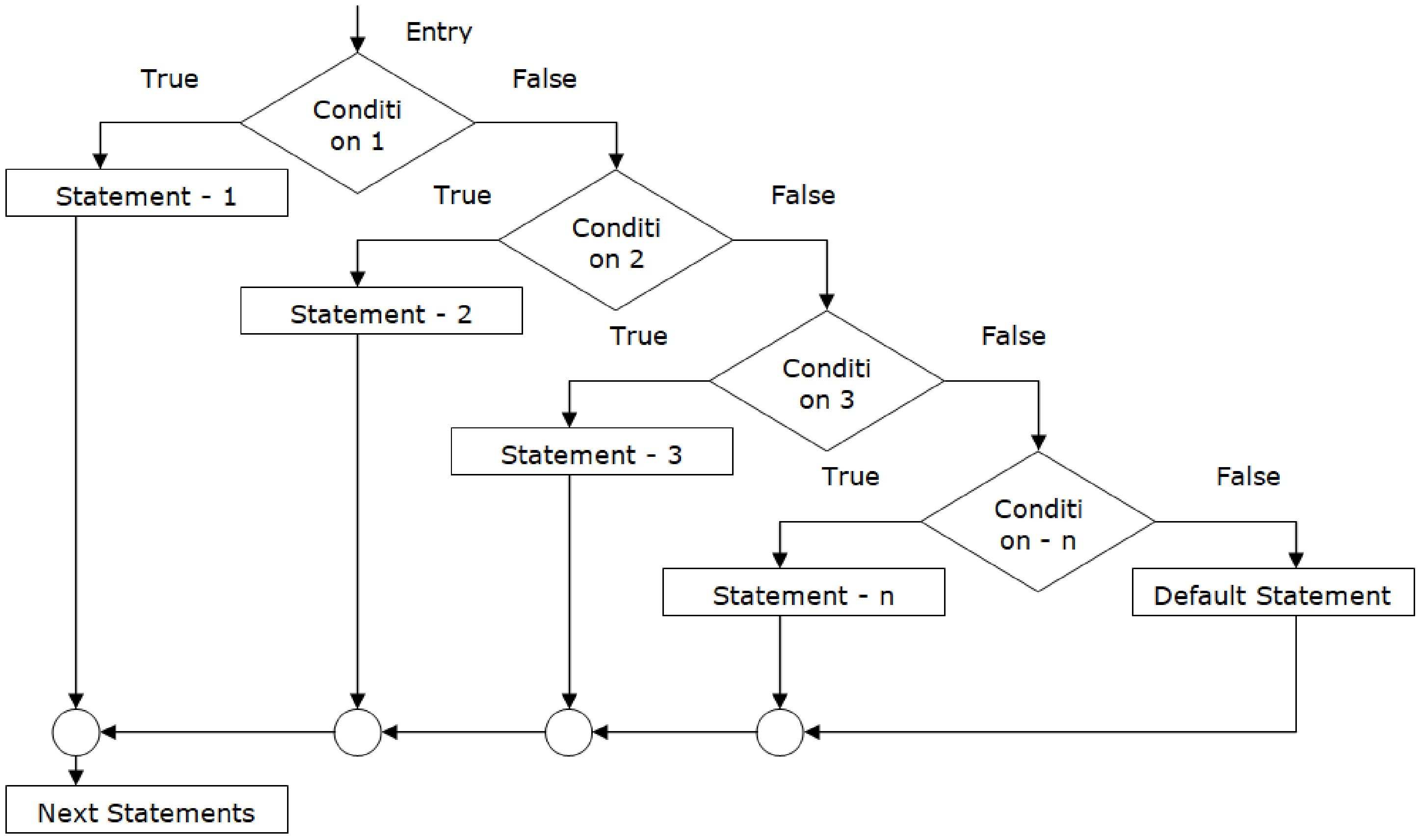
Syntax for if…else…if ladder Statement:



```
if (test condition-1)
        Statement(s)-1;
    else if (test condition-2)
            Statement(s)-2;
    else if (test condition-3)
            Statement(s)-3;
             - - - - - - - - - -
            - - - - - - - - - -
            - - - - - - - - - -
    else if (test condition - n)
            Statement(s)-n;
    else
    Default Statement;
Next Statements;
```

To be executed if condition-1 is true
To be executed if condition-2 is true
To be executed if condition-3 is true
To be executed if condition-n is true
To be executed if no condition is true

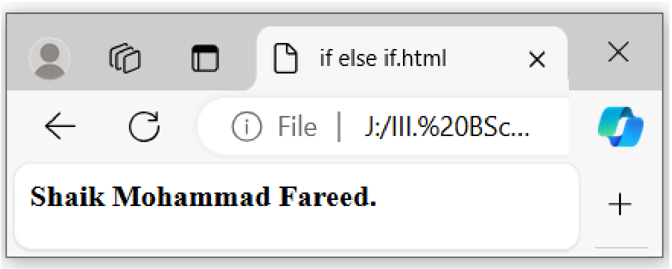
Flow chart for if...else...if ladder statement:



Example:

```
<html>
<body>
  <div id ="shaik"></div>
  <script type="text/javascript">
        const output = document.getElementById("shaik")
    let myname = "smf";
    if (myname == "shaik")
    {
      output.innerHTML="<b>This is Shaik.</b>";
    } else if (myname == "smf")
    {
      output.innerHTML="<b>Shaik Mohammad Fareed.</b>";
    } else if (myname == "abu")
    {
      output.innerHTML="<b>This is Shaik Abu Hanifa.</b>";
    } else
    {
      output.innerHTML="<b>Unknown Name</b>";
    }
   </script>
   </body>
<html>
```

Output:



Shaik Mohammad Fareed.

(5). Switch Statement:

The JavaScript switch statement evaluates an expression and executes a block of code based on matching cases. It provides an alternative to long if-else chains, improving readability and maintainability, especially when handling multiple conditional branches.

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- If there is no match, the default code block is executed.

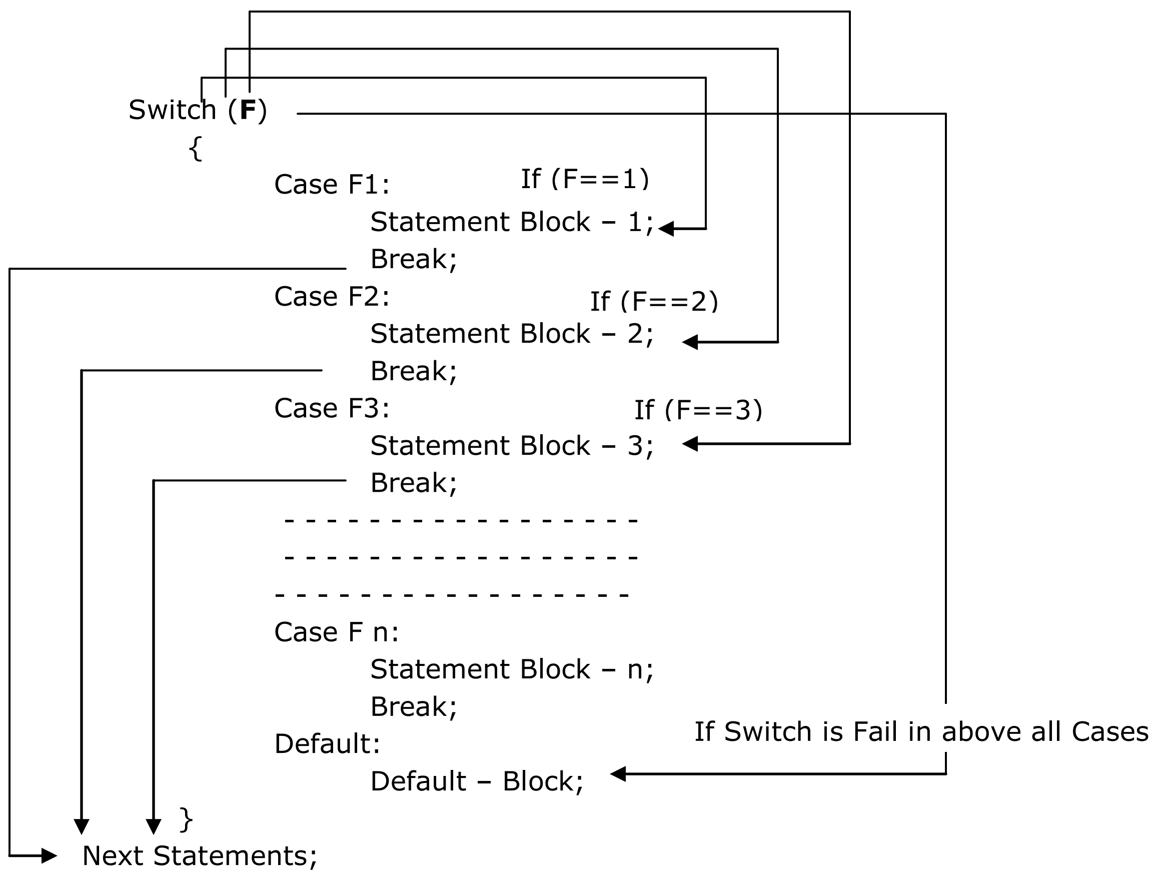
Syntax for Switch Statement:

```
switch (expression)
    {
        case label 1:
                Statement Block-1;
                Break;
        case label 2:
                Statement Block-2;
                Break;
        case label 3:
                Statement Block-3;
                Break;
                - - - - - - - - - - - - - - - - -
                - - - - - - - - - - - - - - - - -
                - - - - - - - - - - - - - - - - -
        case label n:
                Statement Block-n;
                Break;
        default:
                Default-Block;
  }
Next Statements;
```

- **Break:** The statement keyword indicates the end of a particular case. If the 'break' statement were omitted, the interpreter would continue executing each statement in each of the following cases.

- **Default:** The default keyword is used to define the default expression. When any case doesn't match the expression of the switch-case statement, it executes the default code block.

The Control Flow in the Switch statement is shown below.

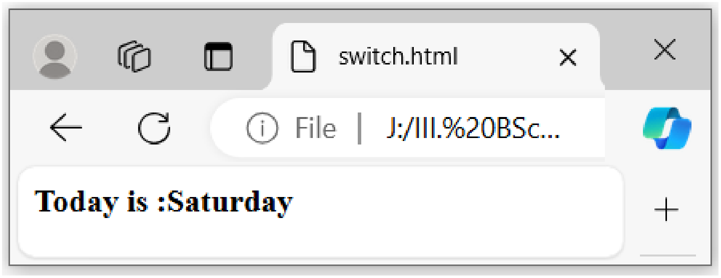


```
Switch (F)
    {
        Case F1:
            Statement Block – 1;
            Break;
        Case F2:
            Statement Block – 2;
            Break;
        Case F3:
            Statement Block – 3;
            Break;
         - - - - - - - - - - - - - - - - -
         - - - - - - - - - - - - - - - - -
        - - - - - - - - - - - - - - - - -
        Case F n:
            Statement Block – n;
            Break;
        Default:
            Default – Block;
    }
Next Statements;
```

Example 1:

```
<html>
<body>
        <p id="shaik"></p>
        <script>
                let day;
                switch (new Date().getDay())
                {
                        case 0:
                                day = "Sunday";
                                break;
                        case 1:
                                day = "Monday";
                                break;
                        case 2:
                                day = "Tuesday";
                                break;
                        case 3:
                                day = "Wednesday";
                                break;
                        case 4:
                                day = "Thursday";
                                break;
                        case 5:
                                day = "Friday";
                        break;
                   case  6:
                        day = "Saturday";
              }
        document.getElementById("shaik").innerHTML = "<b>Today is :<b>" + day;
</script>
</body>
</html>
```

Output:

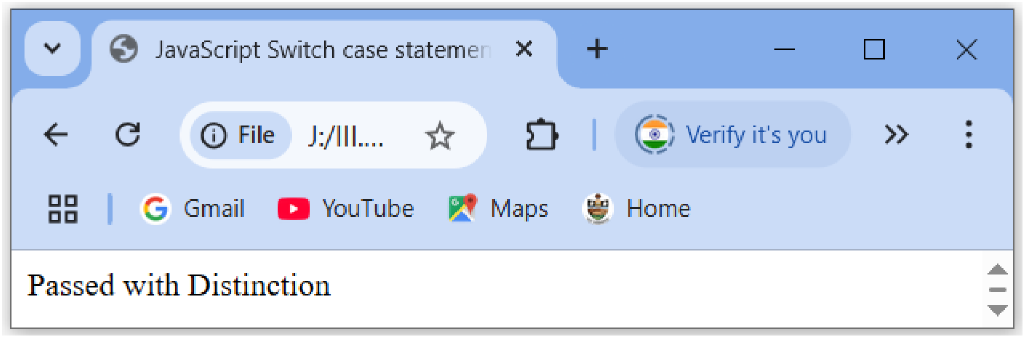


Today is :Saturday

Example 2:

```
<html>
<head>
  <title> JavaScript Switch case statement </title>
</head>
<body>
  <p id = "shaik"> </p>
  <script>
    const output = document.getElementById("shaik");
        let grade = 'A';
     switch (grade)
    {
      case 'A': output.innerHTML += "Passed with Distinction <br />";
               break;
      case 'B': output.innerHTML += "Passed <br />";
               break;
      case 'C': output.innerHTML += "Failed <br />";
               break;
      default: output.innerHTML += "Unknown grade <br />";
    }
  </script>
</body>
</html>
```

Output:



Passed with Distinction

(6). Ternary Operator (?:): The ternary operator is a compact shorthand for an if...else statement. It is called “ternary” because it takes three operands:

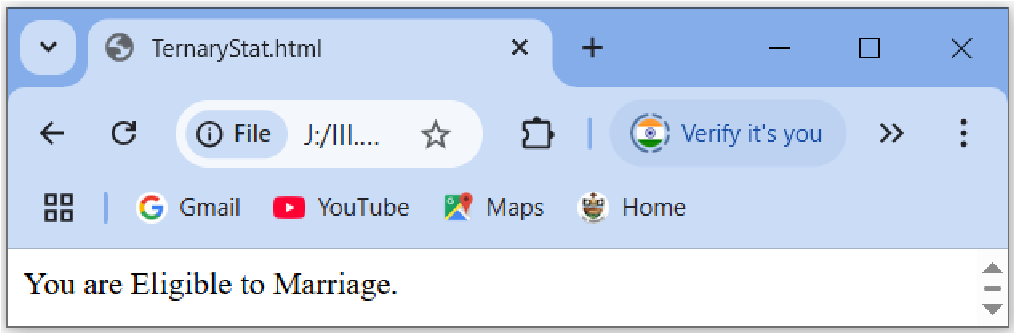
- A condition to test.
- An expression to evaluate if the condition is true.
- An expression to evaluate if the condition is false.

Syntax: condition ? expressionIfTrue : expressionIfFalse;

Example:

```
<html>
<body>
        <p id="shaik"></p>
        <script>
                let age = 39;
                const result = (age >= 18) ? "You are Eligible to Marriage."
                                : "You are Not Eligible to Marriage.";
                document.getElementById("shaik").innerHTML = result;
        </script>
</body>
</html>
```

Output:



```
You are Eligible to Marriage.
```

(II). JavaScript Looping Statements: "The Process of repeatedly executing a block of code as long as a specified condition is true known as Looping". The Statements in the block may be executed any number of times, from one to infinite number. This makes code more concise and efficient.

There are 2 kinds of loops in JavaScript, as given below:

- **Entry-Controlled Loop:** The loop checks whether the looping condition is valid first and enters into the body of the loop to execute the loop statements.

- **Exit-Controlled Loop:** The loop enters into the body and executes the loop statements without checking the condition. After completing the iteration, it checks the condition.

Flow Chart of Entry and Exit-Controlled Looping:

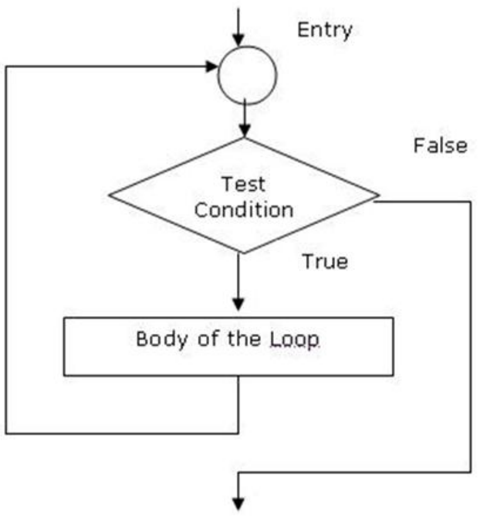


Fig: Entry Control

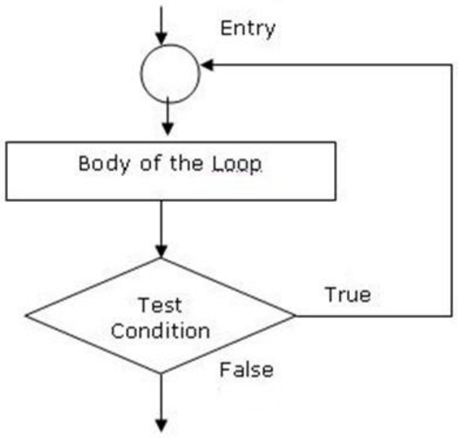


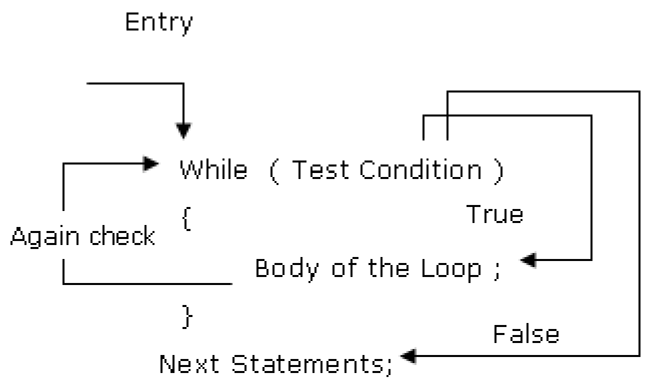
Fig: Exit Control

The Looping statement comes under 3 statements:

(1). while loop statement.
(2). do…while loop statement.
(3). for loop statement.

(1). while Loop Statement: "When the given condition is True, the statements are executed repeatedly. If it is False, the loop will be terminated".

The Control flow in the While Statement is shown below:

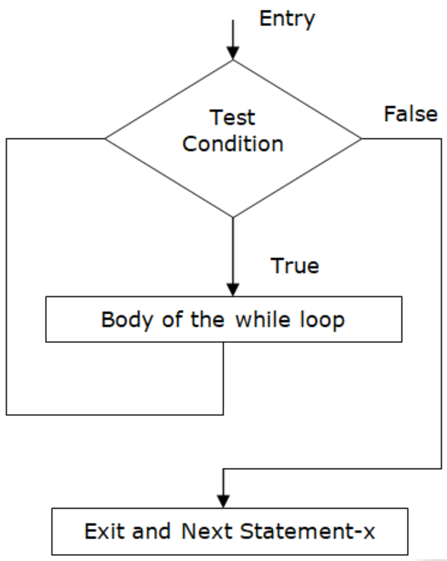


Syntax for while statement:

```
initialization;
while (test condition)
  {
       Body of the loop;
  }
```

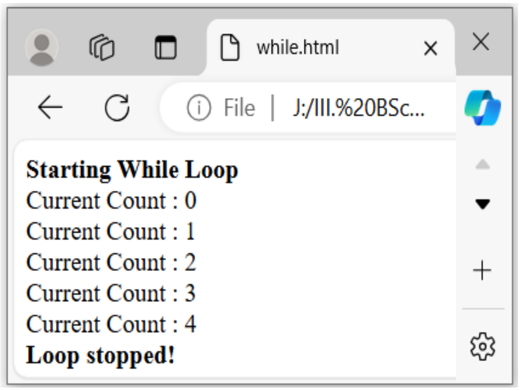
- While Loop is an Entry-Controlled Loop statement as it tests the given condition from the first iteration onwards.
- The Test condition is evaluated and if the condition is "True", then the body of the loop is executed.
- After execution of the body, the tests condition is once again evaluated and if it is 'True', the body is executed once again.
- This process of repeated execution of the body continues until the test condition finally becomes "False" and the control is transferred out of the loop on exit.

Flow Chart for while Statement:



Example:

```
<html>
<body>
   <div id = 'shaik'></div>
   <script type="text/javascript">
      let output = document.getElementById("shaik");
      var count = 0;
      output.innerHTML="<b> Starting While Loop </b> <br>";
      while (count < 5)
      {
         output.innerHTML+="Current Count : " + count + "<br>";
         count++;
      }
      output.innerHTML+=" <b> Loop stopped! </b>";
```



```
Starting While Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop stopped!
```

```
   </script>
</body>
</html>
```

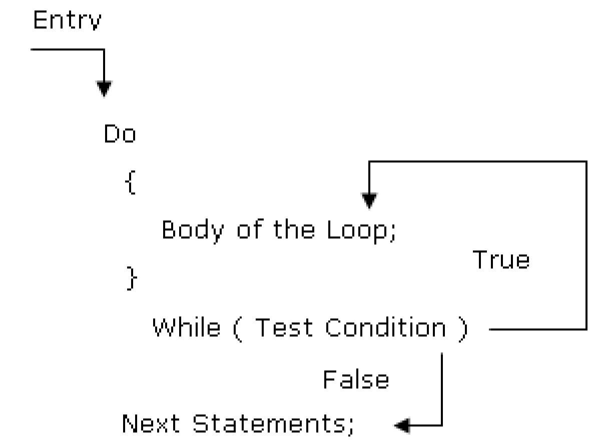
Output:

(2). do…while Loop Statement: The Do…While Loop Statement, executed the body of statement at first then evaluates the condition. If condition is True to allow the body of the loop repeatedly. If it is False, the loop will be terminated.

Syntax for do…while Loop Statement:

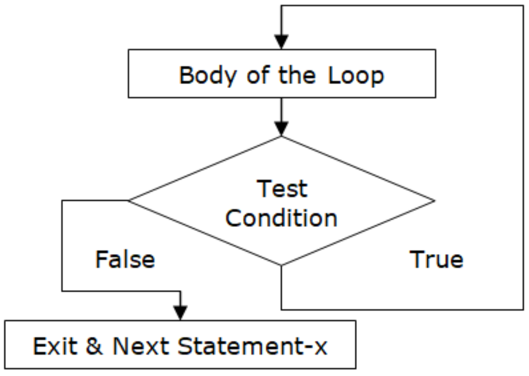
```
do
 {
    Body of the Loop;
 }
  while ( test condition );
        Next Statements;
```

The Control flow in the Do...While Statement: -



- Do…While Loop is an Exit-Condition loop statement as it allows the loop body to be executed for the first time without any condition.
- The Body of statement executed at first time without any test condition, then evaluate the condition, If condition is "True" to evaluate the loop body will be allowed to execute.
- This loop is used to execute a set of statements repeatedly for a finite number of times.
- This process will be continued until the given test condition becomes False, The Loop will be terminated and control goes to the statement that appears immediately after the while statement.

Flow Chart for Do…While Statement:

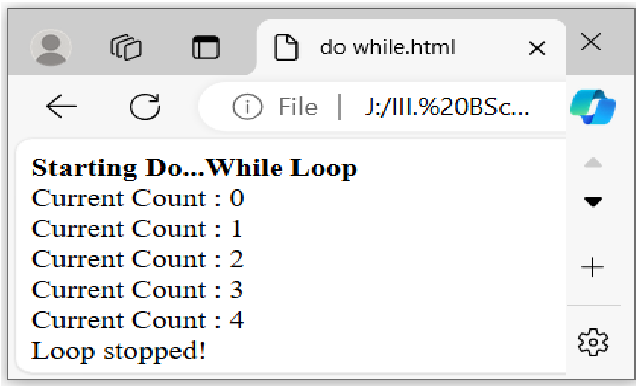


Example:

```
<html>
<body>
   <div id="shaik"></div>
   <script type="text/javascript">
      let output = document.getElementById("shaik");
      var count = 0;
      output.innerHTML += "<b> Starting Do...While Loop </b>" + "<br />";
      do {
         output.innerHTML += "Current Count : " + count + "<br />";
         count++;
      }
      while (count < 5);
      output.innerHTML += "Loop stopped!";
```

```
    </script>
    </body>
</html>
```

Output:



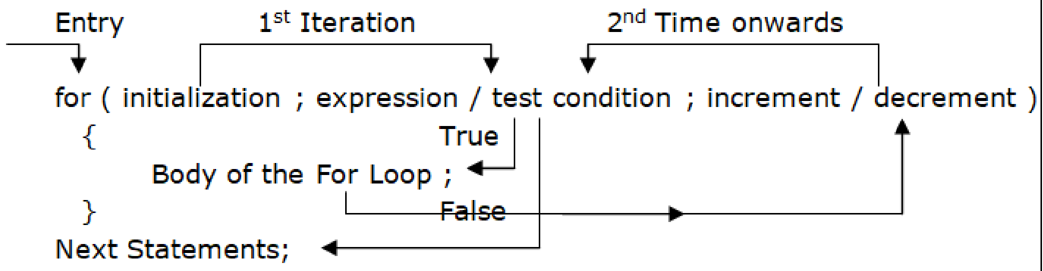
```
Starting Do...While Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop stopped!
```

(3) for Loop Statement: It is a looping statement which repeat again and again till it satisfies the define condition it is also entry control loop.

Syntax of for Loop Statement:

```
for (initialization ; expression / test condition ; increment / decrement )
{
        Body of the For Loop ;
}
```

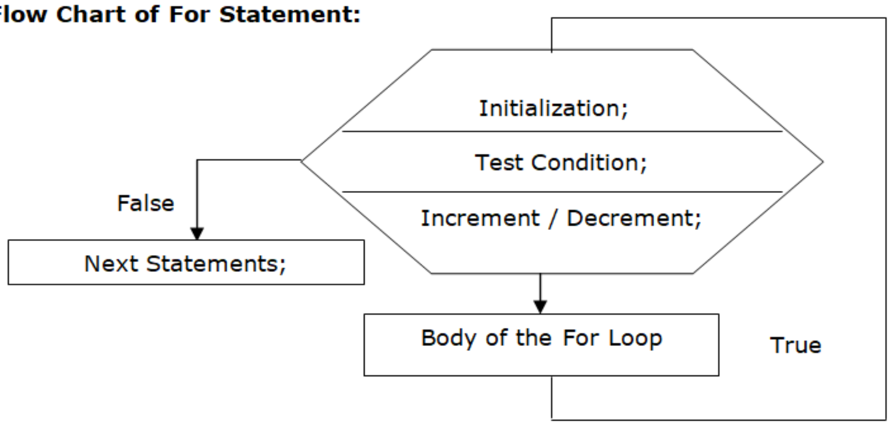
The Control Flow of For Statement:



The JavaScript loops are used to execute the particular block of code repeatedly. The 'for' loop is the most compact form of looping. It includes the following three important parts:

- **Initialization:** The loop initialization expression is where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- **Condition:** The condition expression which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed. Otherwise, the control will come out of the loop.
- **Iteration:** The iteration expression is where you can increase or decrease your counter.

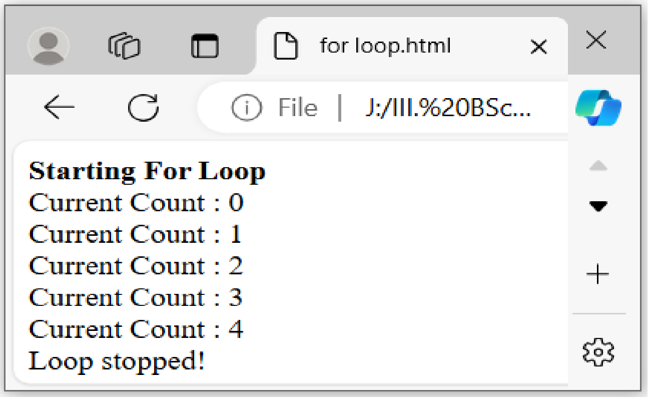
Flow Chart of For Statement:



Example:

```
<html>
<body>
   <p id = "shaik"> </p>
   <script>
      const output = document.getElementById("shaik");
      output.innerHTML = "<b> Starting For Loop </b> <br>";
      let count;
      for (let count = 0; count < 5; count++)
      {
         output.innerHTML += "Current Count : " + count + "<br/>";
      }
      output.innerHTML += "Loop stopped!";
   </script>
</body>
</html>
```

Output:



```
Starting For Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop stopped!
```

(3.a) The for…in Loop: The for…in loop in JavaScript is used to loop through an object's properties. The JavaScript for…in loop is a variant of the for loop. The for loop can't be used to traverse through the object properties. So, the for…in loop is introduced to traverse through all object properties.

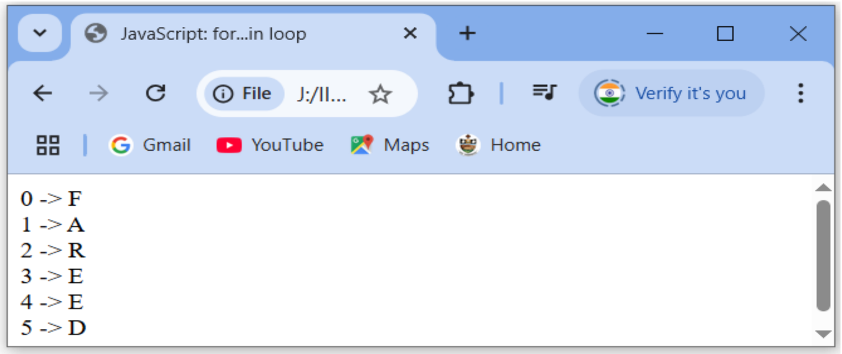
Syntax:
```
for (variableName in object)
 {
      statement or block to execute
 }
```
Parameters:
- **variableName:** It is a property name (key) of the object.
- **in:** It is an 'in' operator in JavaScript.
- **object:** It is the object to traverse.

In each iteration, one property from object is assigned to variableName and this loop continues till all the properties of the object are exhausted.

Example:

```
<html>
<head>
  <title> JavaScript: for...in loop </title>
</head>
<body>
  <p id = "shaik"> </p>
  <script>
    let output = document.getElementById("shaik");
    let str = "FAREED";
    for (key in str)
    {
      output.innerHTML += key + " -> " + str[key] + "<br>";
    }
  </script>
</body>
</html>
```

Output:



```
0 -> F
1 -> A
2 -> R
3 -> E
4 -> E
5 -> D
```

(3.b) JavaScript for...of Loop: The for...of loop in JavaScript is used to traverse elements of the iterable object. In each iteration, it gives an element of the iterable object. Iterable objects include arrays, strings, maps, sets, and generators.

Syntax:
```
for (ele of iterable)
 {
        // loop body
 }
```

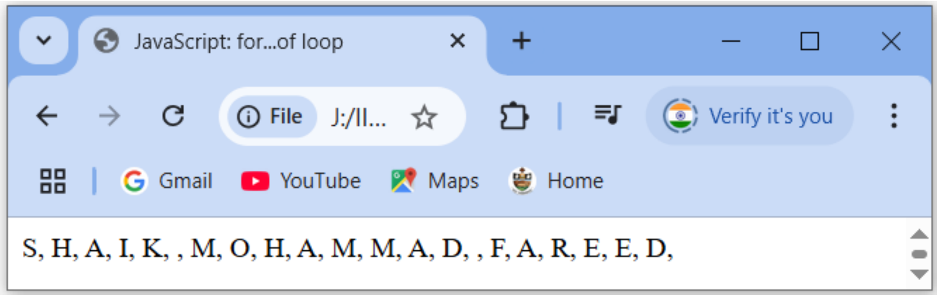
Parameters:

- **ele:** It is a current element of the iterable.
- **of:** It is a JavaScript operator.
- **Iterable:** It is iterable like an object, array, string, etc.

Example:

```
<html>
<head>
   <title> JavaScript: for...of loop </title>
</head>
<body>
   <p id="SMF"> </p>
   <script>
      const output = document.getElementById("SMF");
      let str = "SHAIK MOHAMMAD FAREED";
      for (let char of str)
      {
         output.innerHTML += char + ", ";
      }
   </script>
</body>
</html>
```

Output:



```
S, H, A, I, K, , M, O, H, A, M, M, A, D, , F, A, R, E, E, D,
```

(III). JavaScript Jumping / Transfer Statements:

There are two different controls used to jump from one JavaScript statement to another and make the execution of the programming procedure fast. There are two jumping controls are:

(1). break statement.
(2). continue statement.

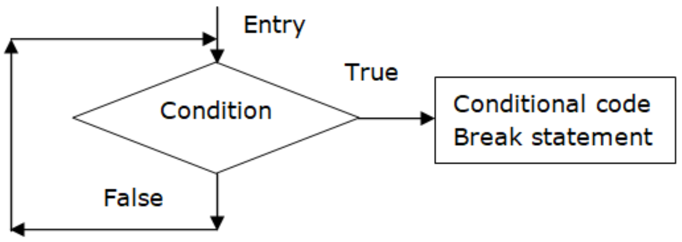
(1). break statement: The break statement in JavaScript terminates the loop or switch case statement. When you use the break statement with the loop, the control flow jumps out of the loop and continue to execute the other code.

The break statement can also be used to jump a labeled statement when used within that labeled statement. It is a useful tool for controlling the flow of execution in your JavaScript code.

Syntax: break; OR break [label];

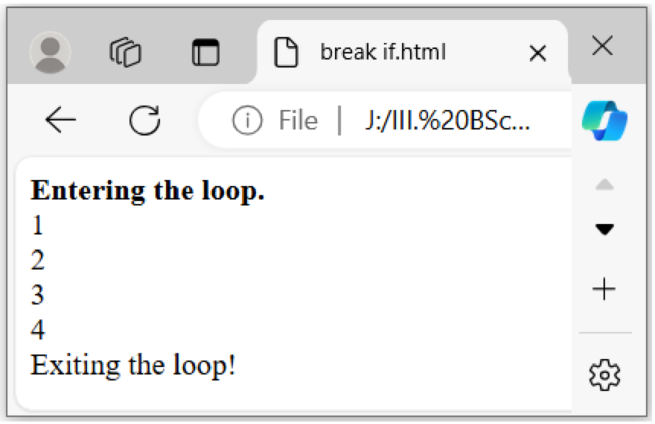
NOTE: The label is optional with a break statement.

Flow chart for break statement:



Example:

```
<html>
<body>
  <p id = "shaik"> </p>
  <script>
    const output = document.getElementById("shaik");
    output.innerHTML += "<b> Entering the loop. </b> <br /> ";
    for (let x = 1; x < 10; x++)
    {
      if (x == 5)
      {
        break;   // breaks out of loop completely
      }
        output.innerHTML += x + "<br />";
    }
    output.innerHTML += "Exiting the loop!<br /> ";
  </script>
</body>
</html>
```



```
Entering the loop.
1
2
3
4
Exiting the loop!
```

Output:

Explanation: In the example, we used the for loop to make iterations. We added the conditional expression in the loop using the 'if' statement. When the value of 'x' is 5, it will 'break' the loop using the break statement. The code prints only 1 to 4 values in the output.

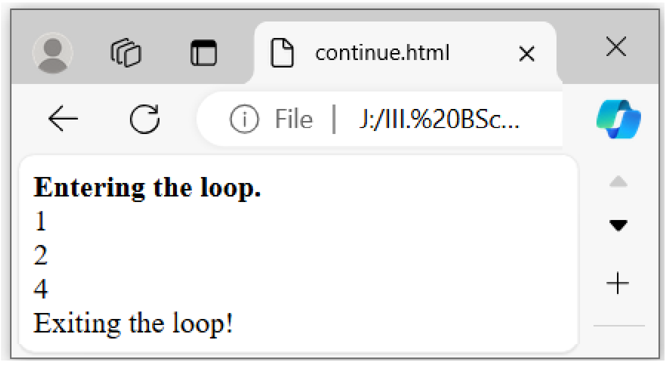
(2). continue statement: The JavaScript continue statement tells the interpreter to immediately start the next iteration of the loop and skip the remaining code block. When a continue statement is encountered, the program flow moves to the loop check expression immediately and if the condition remains true, then it starts the next iteration, otherwise the control comes out of the loop.

Syntax: continue;   OR   continue label;

NOTE: We can use the continue statement inside the loops like for loop, while loop, do…while loop, etc.

Example:

```
<html>
<body>
  <p id = "shaik"> </p>
  <script>
    let output = document.getElementById("shaik");
    output.innerHTML += "<b> Entering the loop. </b> <br /> ";
    for (let x = 1; x < 5; x++)
    {
      if (x == 3)
      {
        continue;   // skip rest of the loop body
      }
      output.innerHTML += x + "<br />";
    }
    output.innerHTML += "Exiting the loop!<br /> ";
  </script>
</body>
</html>
```



```
Entering the loop.
1
2
4
Exiting the loop!
```

Output:

Explanation: The example, uses the continue statement with the for loop. In the loop, when the value of the x is 3, it will execute the continue statement to skip the current iteration and move to the next iteration. In the output, you can see that the loop doesn't print 3.

VVIMP************(Q) What is an Array? Explain about Arrays in JavaScript?**

In JavaScript, an Array is an ordered list of values. Each value is called an Element, and each element has a numeric position in the array, known as its Index. Arrays in JavaScript are zero-indexed, meaning the first element is at index 0, the second at index 1, and so on.

94	91	20	29	87	$\longrightarrow$ Elements
0	1	2	3	4	$\longrightarrow$ Index

Definition of Array: "An Array is a group of contiguous or related data items that share a common name".

(or)

"Array is defined as Collection of homogeneous elements that share a common name".

(or)

"Array is collections of all are Integer or all are Float or all are String / Characters that share a common name".

Key characteristics of JavaScript Arrays are:

- **Elements:** An array is a list of values, known as elements.
- **Ordered:** Array elements are ordered based on their index.
- **Zero indexed:** The first element is at index 0, the second at index 1, and so on.
- **Dynamic size:** Arrays can grow or shrink as elements are added or removed.
- **Heterogeneous:** Arrays can store elements of different data types (numbers, strings, objects and other arrays).

The Difference between Arrays and Objects? When to Use Arrays. When to use Objects?

- In JavaScript, Arrays use numbered indexes.
- In JavaScript, Objects use named indexes.
- JavaScript does not support associative arrays.
- You should use arrays when you want the element names to be numbers.
- You should use objects when you want the element names to be strings (text).

(I). Creating Arrays:

JavaScript Arrays can be constructed in fewer than three different ways. The easiest way is simply to declare a variable and pass it some elements in array formats.

Method 1: General Declaration:

```
const names = ["Shaik", "Mohammad", "Fareed", "MCA", "M.Sc", "IRPM"];
```

That creates an array of six elements, each holding a text string. Notice that the array of elements is surrounded by "Square Brackets". In most programming languages square brackets denote arrays and arrays operations.

NOTE: It is a common to declare arrays with the 'const' keyword.

Method 2: Array Object (new keyword constructor): The Second approach is creating an "Array Object" using the keyword "new" (keyword constructor) and a set of elements to store.

const names = new Array ("Shaik", "Mohammad", "Fareed", "MCA", "M.Sc", "IRPM");

Using this construct, the contents of an array are surrounded by "Parentheses Brackets" because they are parameters to the constructor of the "Array Object".

Method 3: Mixed mode types: In JavaScript Arrays can hold mixed data types as the following:

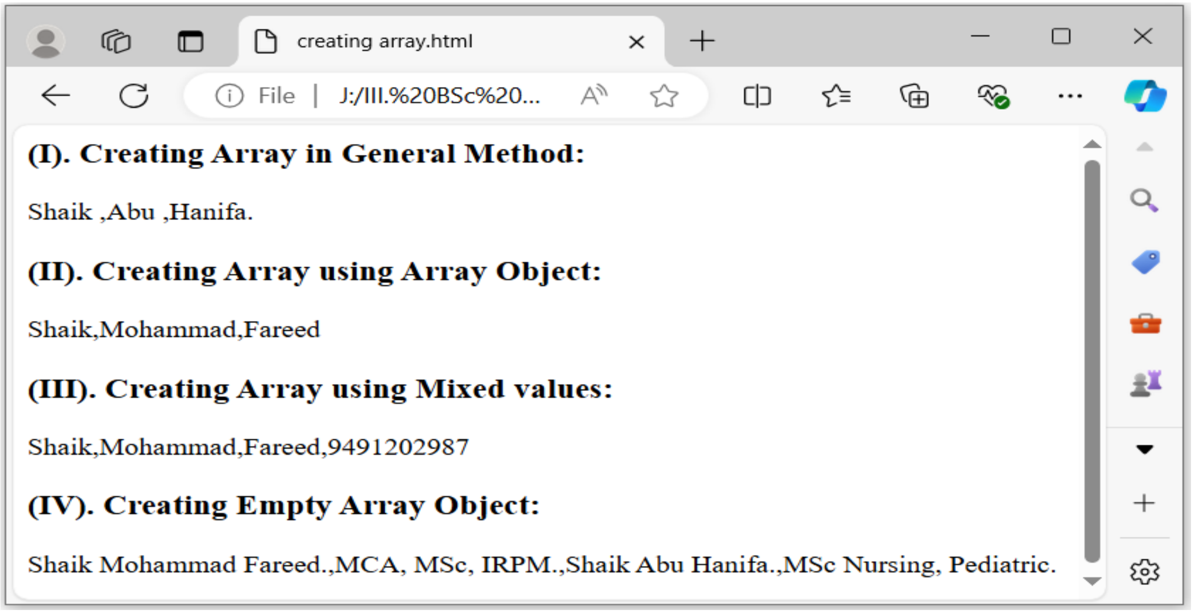
const names = ["Shaik", "Mohammad", "Fareed", "9491202987", "95.9" "MCA", "M.Sc", "IRPM"];
(OR)
const names = new Array("Shaik", "Mohammad", "Fareed", "9491202987", "95.9" "MCA", "M.Sc", "IRPM");

Method 4: Empty Array Object: Finally an empty array objects which has space for number of elements can be created.

const names = new Array(6);

Example:
```
<html>
<body>
<h3>(I). Creating Array in General Method: </h3>
<p id="SAH"></p>
        <script>
                const myname = ["Shaik ", "Abu ", "Hanifa."];
                document.getElementById("SAH").innerHTML = myname;
        </script>

<h3>(II). Creating Array using Array Object: </h3>
<p id="SMF"></p>
        <script>
                const mynames = new Array("Shaik", "Mohammad", "Fareed");
                document.getElementById("SMF").innerHTML = mynames;
        </script>
<h3>(III). Creating Array using Mixed values: </h3>
<p id="ABUSMF"></p>
        <script>
                const names = new Array("Shaik", "Mohammad", "Fareed", 9491202987);
                document.getElementById("ABUSMF").innerHTML =  names;
        </script>

<h3>(IV). Creating Empty Array Object: </h3>
<p id="ISM"></p>
        <script>
                const qualification = [];
                qualification [0] = "Shaik Mohammad Fareed.";
                qualification [1] = "MCA, MSc, IRPM.";
                qualification [2] = "Shaik Abu Hanifa.";
                qualification [3] = "MSc Nursing, Pediatric.";
                document.getElementById("ISM").innerHTML = qualification +"<br>";
        </script>
</body>
</html>
```

Output:



(I). Creating Array in General Method:

Shaik ,Abu ,Hanifa.

(II). Creating Array using Array Object:

Shaik,Mohammad,Fareed

(III). Creating Array using Mixed values:

Shaik,Mohammad,Fareed,9491202987

(IV). Creating Empty Array Object:

Shaik Mohammad Fareed.,MCA, MSc, IRPM.,Shaik Abu Hanifa.,MSc Nursing, Pediatric.

(II). Adding Elements to an Array:

Array elements are accessed by their Index. In JavaScript if we want to add an item to an Array which already full. Then the interpreters simply extend the array and insert the new item into an array.

1. var names = ["Shaik", "Mohammad", "Fareed", "9491202987"];
2. names[4] = "Hanifa";

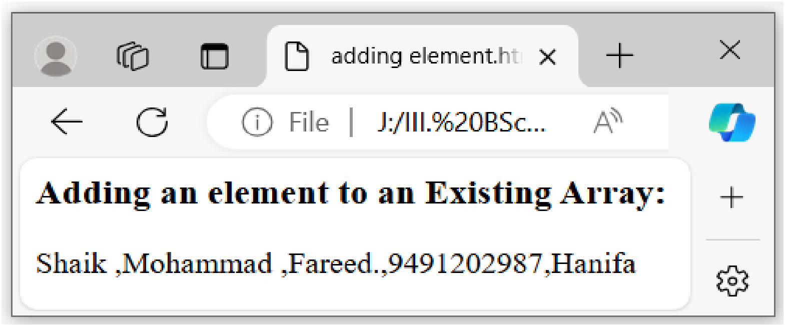
NOTE:

- The above code creates an array of four elements in line one. A new element "Hanifa" is added at position at '4th' in to line one.
- Array Index start with zero (0).
- Array[0] is first element, Array[1] is second element, Array[n-1] is last element in the array.

Example:

```
<html>
<body>
<h3>Adding an element to an Existing Array:</h3>
<p id="abu"></p>
        <script>
                const myname = ["Shaik ", "Mohammad ", "Fareed.", "9491202987"];
                myname[4] = "Hanifa";
                document.getElementById("abu").innerHTML = myname;
</script>
</body>
</html>
```

Output:



Adding an element to an Existing Array:

Shaik ,Mohammad ,Fareed.,9491202987,Hanifa

(III). Accessing Array Members:

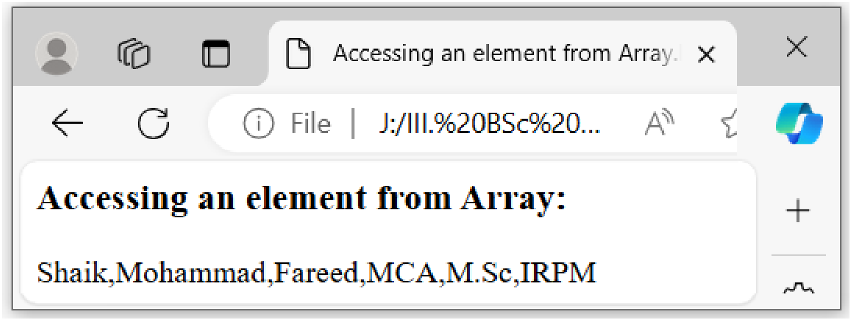
The elements in the array are accessed through their index. The same access method is used to find elements and to change their values.

When accessing array elements you don't want to read beyond its ends. Therefore you need to know how many elements have been stored. This is done through the "Length" attribute. That index numbers run from '0' to (length-1).

Example 1:

```
<html>
<body>
<h3>Accessing an element from Array:</h3>
        <script>
                const names = ["Shaik", "Mohammad", "Fareed", "MCA", "M.Sc", "IRPM"];
                const len = names.length;
                for(const count=0; count<len; count++)
                        {
                                document.writeln(names);
                        }
        </script>
</body>
</html>
```

Output:



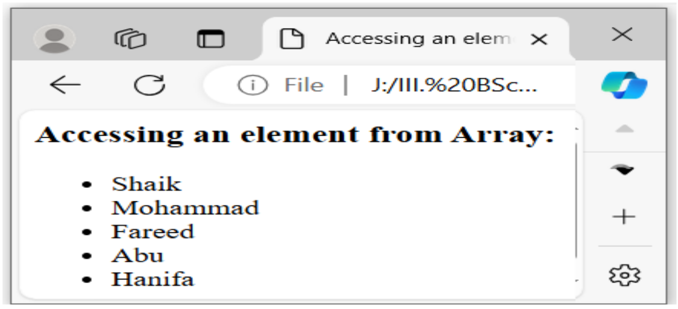
Accessing an element from Array:

Shaik,Mohammad,Fareed,MCA,M.Sc,IRPM

Example 2:

```
<html>
<body>
<h3>Accessing an element from Array:</h3>
        <p id="shaik"></p>
        <script>
                const mynames = ["Shaik", "Mohammad", "Fareed", "Abu", "Hanifa"];
                let leng = mynames.length;
                let namelist = "<ul>";
                for (let i = 0; i < leng; i++)
                        {
                                namelist += "<li>" + mynames[i] + "</li>";
                        }
                namelist += "</ul>";
                document.getElementById("shaik").innerHTML = namelist;
        </script>
</body>
</html>
```

Output:



Accessing an element from Array:

- Shaik
- Mohammad
- Fareed
- Abu
- Hanifa

(IV). Searching an Array: To search an array, simply read each element in turn and compare it with the value.

Example:

```
        var names = [“Shaik”, “Mohammad”, “Fareed”, “MCA”, “M.Sc”, “IRPM”];
        var len=names.length;
        for(var count=0; count<len; count++)
        {
                if(names[count] == “Fareed”);
                  {
                    document.write(data[count] + ", ");
                    break;
                  }
        }
```

The following code in the script this loop through the array, compares each elements with a string, if the two elements are equal a message is printed, to stop the search. Here built-in ‘break’ function which terminates the current loop. You can use ‘break’ with ‘for’ and ‘while’ loop.

Array Properties: Here is a list of each property and their description.

Property	Description
constructor	Returns a reference to the array function that created the object.
index	The property represents the zero-based index of the match in the string
input	This property is only present in arrays created by regular expression matches.
length	Reflects the number of elements in an array.
prototype	The prototype property allows you to add properties and methods to an object.

Array Methods: Here is a list of each method and its description.

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value or 1 if none is found.
join()	Joins all elements of an array into a string.
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value or 1 if none is found.
map()	Creates a new array with the results of calling a provided function on every element in this array.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reduce()	Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.

reduceRight()	Apply a function simultaneously against two values of the array (from right-to-left) as to reduce it to a single value.
reverse()	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
slice()	Extracts a section of an array and returns a new array.
some()	Returns true if at least one element in this array satisfies the provided testing function.
toSource()	Represents the source code of an object
sort()	Sorts the elements of an array.
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representing the array and its elements.
unshift()	Adds one or more elements to the front of an array and returns the new length of the array.

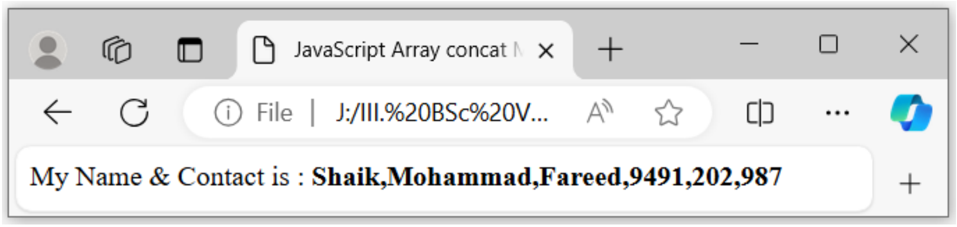
1. JavaScript concat() Method: JavaScript array concat() method returns a new array comprised of this array joined with two or more arrays.

Syntax: arrayname.concat(value 1, value 2, ….., value n);

Example:

```
<html>
<head>
<title>JavaScript Array concat Method</title>
</head>
<body>
<script language="javascript">
  var myname = ["Shaik", "Mohammad", "Fareed"];
  var number = [9491, 202, 987];
  var mydetails = myname.concat(number);
  document.write("My Name & Contact is  : " + "<b>" + mydetails );
</script>
</body>
</html>
```

Output:



```
My Name & Contact is : Shaik,Mohammad,Fareed,9491,202,987
```

2. JavaScript Array indexOf(): The indexOf() method searches an array for an element value and returns its position.

Syntax: arrayname.indexOf(item, start)

- **item:** Required. The item to search for.
- **start:** Optional. Where to start the search. Negative values will start at the given position counting from the end, and search to the end.

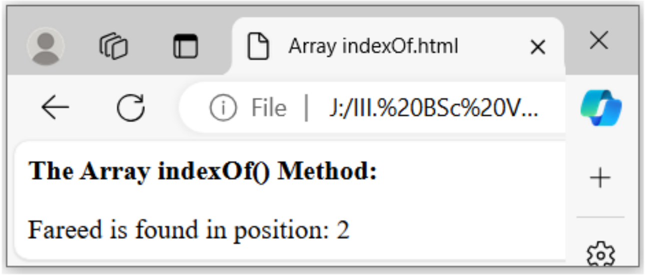
Note:

- Array.indexOf() returns -1 if the item is not found.
- The first item has position 0; the second item has position 1, and so on.
- If the item is present more than once, it returns the position of the first occurrence.

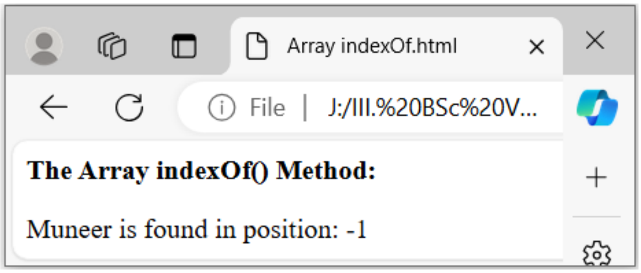
Example:

```
<html>
<body>
<b>The indexOf() Method</b>
<p id="shaik"></p>
<script>
        const mynames = ["Shaik", "Mohammad", "Fareed", "Abu", "Hanifa"];
        let position = mynames.indexOf("Fareed");
        document.getElementById("shaik").innerHTML = "Fareed is found in position: " + position;
</script>
</body>
</html>
```

Output:



```
The Array indexOf() Method:

Fareed is found in position: 2
```



```
The Array indexOf() Method:

Muneer is found in position: -1
```

NOTE: let position = mynames.indexOf("Muneer"); → Output: If it is not found returns -1.

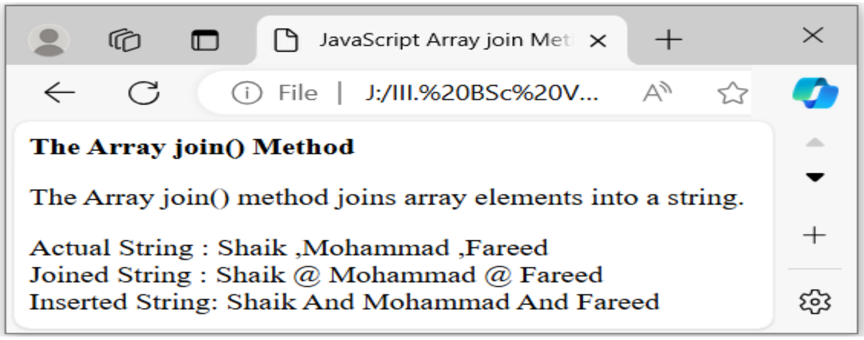
3. JavaScript Array join(): The join() method also joins all array elements into a string. It behave just like 'toString()', but in addition you can specify the separator.

Syntax: arrayname.join("value/element");

Example:

```
<html>
<head>
<title>JavaScript Array join Method</title>
</head>
<b>The Array join() Method</b>
<p>The Array join() method joins array elements into a string.</p>
<body>
        <script type="text/javascript">
                var arr = new Array("Shaik ", "Mohammad ", "Fareed");
                var str = arr.join();
                        document.write("Actual String : " + str );
                var str = arr.join("@ ");
                        document.write("<br />Joined String : " + str );
                var str = arr.join(" And ");
                        document.write("<br />Inserted String: " + str );
        </script>
</body>
</html>
```

Output:



```
The Array join() Method

The Array join() method joins array elements into a string.

Actual String : Shaik ,Mohammad ,Fareed
Joined String : Shaik @ Mohammad @ Fareed
Inserted String: Shaik And Mohammad And Fareed
```

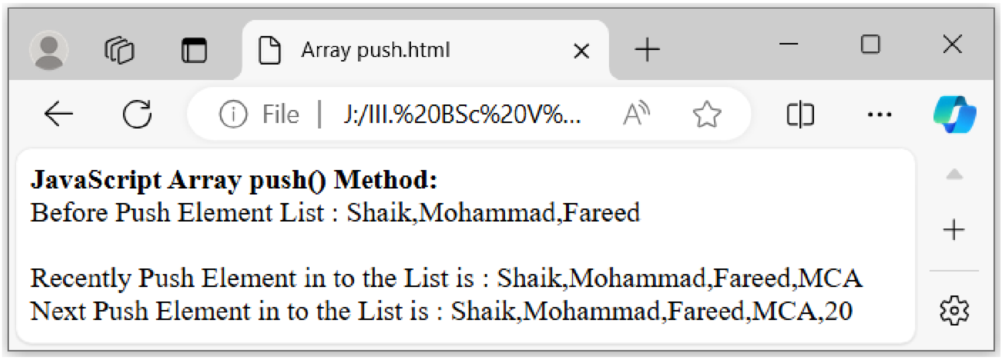
4. JavaScript Array push(): JavaScript array push() method appends (add at the end) the given element(s) and returns the length of the new array.

Syntax: arrayname.push("value/element");

Example:

```
<html>
<body>
<b>JavaScript Array push() Method: </b><br>
<script>
        var myname = new Array("Shaik", "Mohammad", "Fareed");
                document.write("Before Push Element List  : " + myname );
        var length = myname.push("MCA");
                document.write("<br /> <br />Recently Push Element in to the List is   : " + myname );
        length = myname.push(20);
                document.write("<br />Next Push Element in to the List is : " + myname );
</script>
</body>
</html>
```

Output:



```
JavaScript Array push() Method:
Before Push Element List : Shaik,Mohammad,Fareed

Recently Push Element in to the List is : Shaik,Mohammad,Fareed,MCA
Next Push Element in to the List is : Shaik,Mohammad,Fareed,MCA,20
```

5. JavaScript Array pop(): JavaScript array pop() method removes the last element from an array and returns that element.

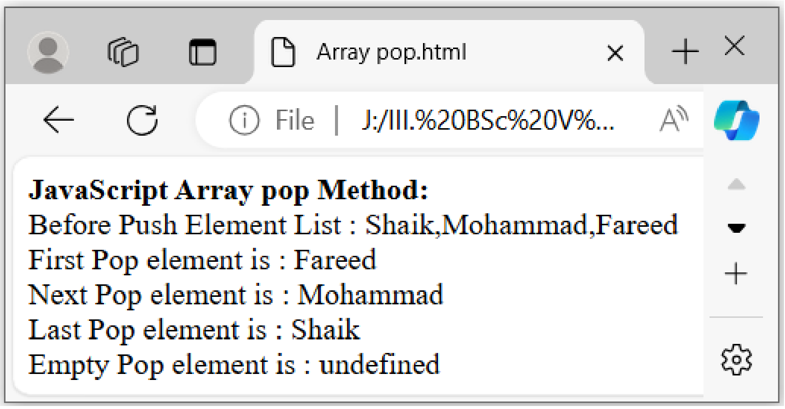
Syntax: arrayname.pop()

Example:

```
<html>
<body>
<b>JavaScript Array pop Method: </b> <br>
<script type="text/javascript">

        var  myname = ["Shaik", "Mohammad", "Fareed"];
        document.write("Before Push Element List  : " + myname + "<br>");

        var element = myname.pop();
        document.write("First Pop element is : " + element );

        var element = myname.pop();
        document.write("<br />Next Pop element is : " + element );

        var element = myname.pop();
        document.write("<br />Last Pop element is : " + element );

        var element = myname.pop();
        document.write("<br />Empty Pop element is : " + element );
```

```
</script>
</body>
</html>
```

Output:



Array pop.html

J:/III.%20BSc%20V%...

```
JavaScript Array pop Method:
Before Push Element List : Shaik,Mohammad,Fareed
First Pop element is : Fareed
Next Pop element is : Mohammad
Last Pop element is : Shaik
Empty Pop element is : undefined
```

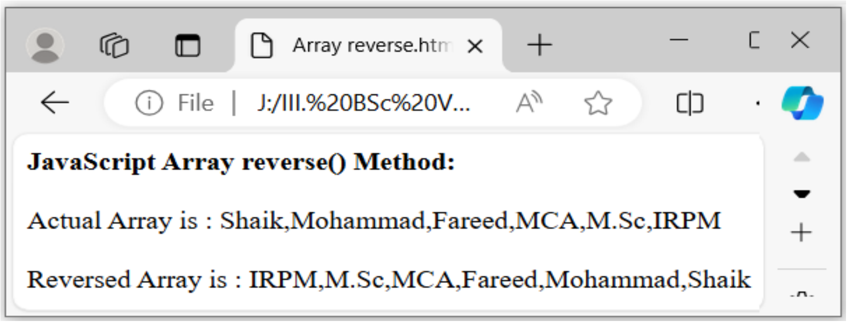
6. JavaScript Array reverse(): JavaScript array reverse() method reverses the element of an array. The first array element becomes the last and the last becomes the first.

Syntax: arrayname.reverse();

Example:

```
<html>
<body>
<b>JavaScript Array reverse() Method: </b> <br>
<script>
        var name = ["Shaik", "Mohammad", "Fareed", "MCA", "M.Sc", "IRPM"];
        document.write("</br>  Actual Array is : " + name);
        var name = ["Shaik", "Mohammad", "Fareed", "MCA", "M.Sc", "IRPM"].reverse();
        document.write("</br> </br> Reversed Array is : " + name);
</script>
</body>
</html>
```

Output:



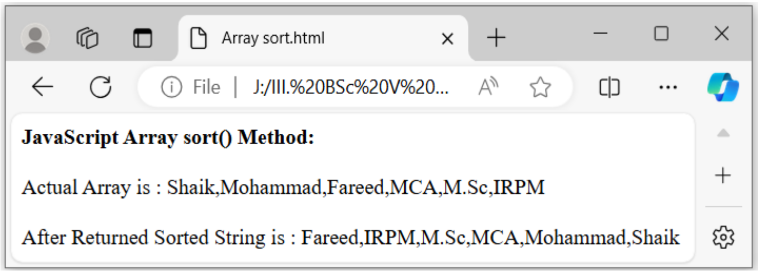
Array reverse.htm

J:/III.%20BSc%20V...

```
JavaScript Array reverse() Method:

Actual Array is : Shaik,Mohammad,Fareed,MCA,M.Sc,IRPM

Reversed Array is : IRPM,M.Sc,MCA,Fareed,Mohammad,Shaik
```

7. JavaScript Array sort(): JavaScript array sort() method sorts the elements of array.

Syntax: arrayname.sort();



Array sort.html

J:/III.%20BSc%20V%20...

```
JavaScript Array sort() Method:

Actual Array is : Shaik,Mohammad,Fareed,MCA,M.Sc,IRPM

After Returned Sorted String is : Fareed,IRPM,M.Sc,MCA,Mohammad,Shaik
```

Example:

```
<html>
<body>
<b>JavaScript Array sort() Method: </b> <br>
<script>
        const name = ["Shaik", "Mohammad", "Fareed", "MCA", "M.Sc", "IRPM"];
        document.write("</br>  Actual Array is : " + name);
        const sortedname = name.sort();
        document.write("</br> </br> After Returned Sorted String is : " + sortedname );
</script>
</body>
</html>
```

Output:

IMP************(Q) Define Functions in JavaScript with an example?**

Definition of Function: Functions in JavaScript are reusable blocks of code designed to perform specific tasks. They allow you to organize, reuse, and modularize code. It can take inputs, perform actions, and return outputs.

(OR)

A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when "something" invokes it (calls it).

Why Functions?

- ❖ Functions can be used multiple times, reducing redundancy.
- ❖ Break down complex problems into manageable pieces.
- ❖ Manage complexity by hiding implementation details.
- ❖ Can call themselves to solve problems recursively.

Types of Functions: Functions are classified into two types:

- Built-in Functions.
- User defined Functions.

Built-in Functions: Those functions already defined are known as Built-in Functions.

User defined Functions: Those functions are defined by users its own.

Function Syntax:

```
function <function name> (parameter1, parameter2, …..)
{
        -----------  ⎫
        -----------  ⎬ Body of the Function
        -----------  ⎭
}
```

Example of Function:

```
function sum(x, y)
{
        return x + y;
}
document.write (sum(6, 9));
```

Output: 15

Explanation: A function definition is sometimes also termed a function declaration or function statement. Below are the rules for creating a function in JavaScript:

- Begin with the keyword "function" Keyword
- A user-defined function name (In the above example, the name is 'sum')
- A list of parameters enclosed within parentheses and separated by commas (In the above example, parameters are 'x' and 'y')
- A list of statements composing the body of the function enclosed within curly braces {} (In the above example, the statement is "return x + y").

Each of the functions can have the following:

1. Function Name.
2. List of Parameters / Arguments.
3. List of Statements.
4. Return type.

Function Syntax: function <function name> (parameter1, parameter2, …..)

```
{
        -----------
        -----------    } Body of the Function
        -----------
}
```

1. **Function Name:** Functions are defined using the 'function' keyword. The Function name can be any combination of digits, letters and underscore but not contain a space. A Function is a block of code and so has to have its curly brackets. A Function Name is a user defined.

2. **List of Parameters:**

```
function fareed (name, age)
{
  document.write (name + "is" + age + "years old.");
}
```

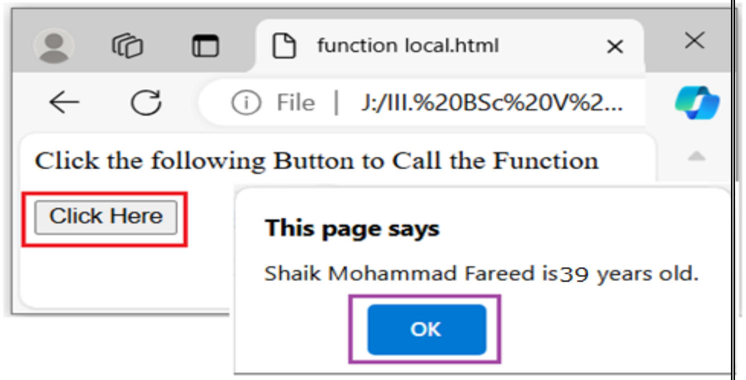
Parameters are input passed to a function. In the above example, fareed (name, age) takes two parameters, 'name' and 'age'. A Function can take parameters by separated by comma.

3. **List of Statements:** The body of the function contains a number of statements, which can execute in a block of function.

4. **Return type:** A function can return a result using the return keyword. This is optional but useful when you want to send data back from the function. The return value is "returned" back to the "caller":

Example 1:

```
<html>
<head>
<script>
        function fareed(name, age)
        {
        alert( name + " is " + age + " years old.");
        }
</script>
</head>
<body>
<p>Click the following Button to Call the Function</p>
<form>
        <input type="button" onclick="fareed('Shaik Mohammad Fareed', 39)" value="Click Here">
</form>
</body>
</html>
```



Click the following Button to Call the Function

This page says

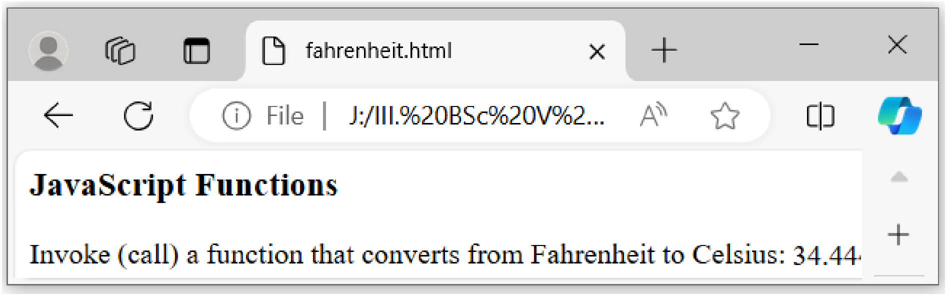
Shaik Mohammad Fareed is39 years old.

Output:

Example 2:

```
<html>
<body>
<h3>JavaScript Functions</h3>
<p>Invoke (call) a function that converts from Fahrenheit to Celsius:</p>
<p id="shaik"></p>
<script>
        function toCelsius(f)
        {
                return (5/9) * (f-32);
        }
        let value = toCelsius(94);
        document.getElementById("shaik").innerHTML = value;
</script>
</body>
</html>
```

Output:



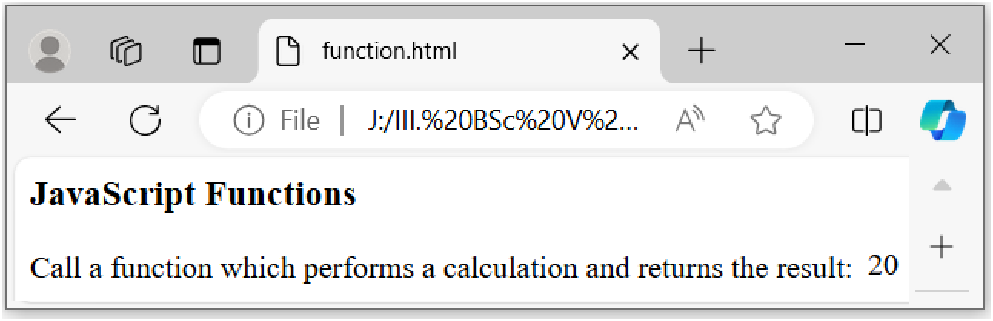
JavaScript Functions

Invoke (call) a function that converts from Fahrenheit to Celsius: 34.44

Example 3:

```
<html>
<body>
<h3>JavaScript Functions</h3>
<p>Call a function which performs a calculation and returns the result: </p>
<p id="shaik"></p>
<script>
        let x = myFunction(5, 4);
        document.getElementById("shaik").innerHTML = x;
        function myFunction(a, b)
        {
                return a * b;
        }
</script>
</body>
</html>
```

Output:



JavaScript Functions

Call a function which performs a calculation and returns the result: 20

Function Invocation:

- The function code you have written will be executed whenever it is called.
- Triggered by an event (e.g., a button click by a user).
- When explicitly called from JavaScript code.
- Automatically executed, such as in self-invoking functions.

Function Expression: It is similar to a function declaration "without the function name". Function expressions can be stored in a variable assignment.

Syntax:
```
let variableName = function (parameter1, parameter2,…)
{
        Set of Statements
}
```
Example:
```
let mul = function (x, y)
{
        return x * y;
};
document.write (mul(4, 5));
```

Output: 20

Scoping Rules: When you declare a variable you might naively except that it can be used anywhere in your program but that is not actually the case. In JavaScript variables can be either Local or Global.

- **Local:** Local variables are declared inside a function. They can only be used by that function.

- **Global:** Global variables are available to all parts of the programs. Such variables are declared outside of any function.

IMP*****(Q) Explain about String Manipulation Functions / String Object with an examples?**

JavaScript strings are used for storing and manipulating text content. They can contain zero or more characters within single or double quotes, like "**Fareed**" or '**Fareed**'.

Syntax: String(object)

String Methods/ String Manipulations Functions/ String Objects: String is a collection of characters. In JavaScript using string objects many string related functionalities can be exposed off. Some commonly used methods of string objects are concatenating two strings, converting the string to upper case or lower case, finding the substring of a given strings and so on.

String Methods	Meanings
chartAt(index)	This function returns the character which is at position index in the string.
chartcodeAt(index)	This function returns the character code (ASCII) of the character at the specify index.
concate(str)	This method concatenates the two stings.
indexOf("search" [, offset])	This function is used to search substring in the main string. If it the substring in main string otherwise it returns -1.
lastIndexOf("search" [, offset])	• The lastIndexOf() method returns the position of the last found occurrence of a specified value in a string. • The string is searched backward, but the index returned is the character position from left to right (starting at 0). • This method return -1 if the value to search for never occurs.
length()	This function returns the length of the string.

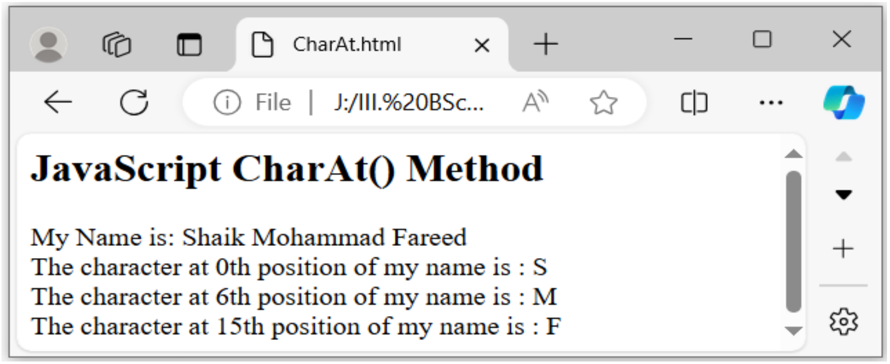
split(separator[, limit])	This function splits a string into an array of substring.
substring(index1[, index2])	Returns the set of characters which starts at index1 and continues up to, but does not include, the character at index2. They following rules • If index 1 is less than 0, it will be treated as 0. • If index 2 is gather than the length of the string, it is treated as length of the string. • If the two index values are equal, an empty string is returned. • If index 2 is missing, all characters up to end of the string. • If index 1 is greater than index 2, a runtime error occurs.
toLowerCase()	Converts all characters in sting to lower cases.
toUpperCase()	Converts all characters in sting to upper cases.

(1). CharAt(): This function returns the character which is at position index (position) in the string. The index of first character is '0', second is '1',… . The index of last character is string length -1.

Syntax: string.charAt(index);

```
<html>
<body>
<h2>JavaScript CharAt() Method</h2>
<script>
        var myname = "Shaik Mohammad Fareed";
        document.write("My Name is: "+ myname + "<br>");
        document.write("The character at 0th position of my name is : "+ myname.charAt(0)+"<br>");
        document.write("The character at 6th position of my name is : "+ myname.charAt(6)+"<br>");
        document.write("The character at 15th position of my name is : "+ myname.charAt(15)+"<br>");
</script>
</body>
</html>
```

Output:



JavaScript CharAt() Method

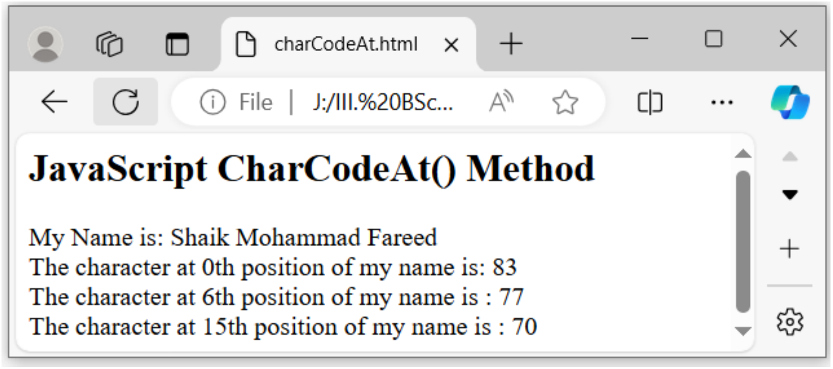
```
My Name is: Shaik Mohammad Fareed
The character at 0th position of my name is : S
The character at 6th position of my name is : M
The character at 15th position of my name is : F
```

(2). charCodeAt(): This function returns the Unicode (ASCII) of the character at a specified index(position) in a string. The index of first character is '0', second is '1',… .The index of last character is string length -1.

Syntax: string.charCodeAt(index);

```
<html>
<body>
<h2>JavaScript CharCodeAt() Method</h2>
<script>
var myname = "Shaik Mohammad Fareed";
document.write("My Name is: "+ myname + "<br>");
document.write("The character at 0th position of my name is: "+ myname.charCodeAt(0)+"<br>");
document.write("The character at 6th position of my name is : "+ myname.charCodeAt(6)+"<br>");
document.write("The character at 15th position of my name is : "+ myname.charCodeAt(15)+"<br>");
</script>
</body>
</html>
```

Output:



```
JavaScript CharCodeAt() Method

My Name is: Shaik Mohammad Fareed
The character at 0th position of my name is: 83
The character at 6th position of my name is : 77
The character at 15th position of my name is : 70
```

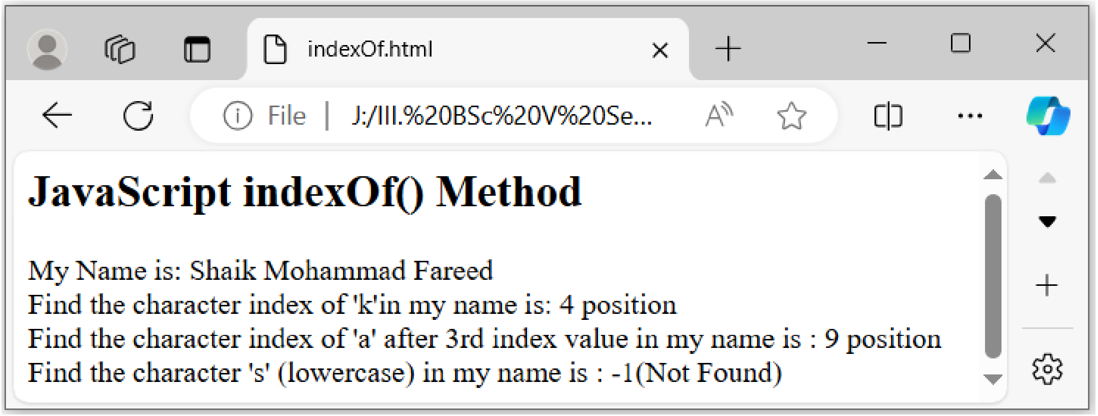
(3). indexOf():

- The string is searched for the string or quoted character in the first parameter.
- If the search is successful, the index of the start of the target string is returned.
- The indices in the original string number from 0 to string length -1.
- If the search is unsuccessful the operation returns -1.
- The indexOf() method is case sensitive.

Syntax: string.indexOf("searchvalue", start);

```
<html>
<body>
<h2>JavaScript indexOf() Method</h2>
<script>
var myname = "Shaik Mohammad Fareed";
document.write("My Name is: "+ myname + "<br>");
document.write("Find the character index of 'k'in my name is: "+ myname.indexOf("k")+" position <br>");
document.write("Find the character index of 'a' after 3rd index value in my name is : "+
                myname.indexOf("a",3) +" position<br>");
document.write("Find the character 's' (lowercase) in my name is : "+ myname.indexOf("s")+"(Not Found)
<br>");
</script>
</body>
</html>
```

Output:



```
JavaScript indexOf() Method

My Name is: Shaik Mohammad Fareed
Find the character index of 'k'in my name is: 4 position
Find the character index of 'a' after 3rd index value in my name is : 9 position
Find the character 's' (lowercase) in my name is : -1(Not Found)
```

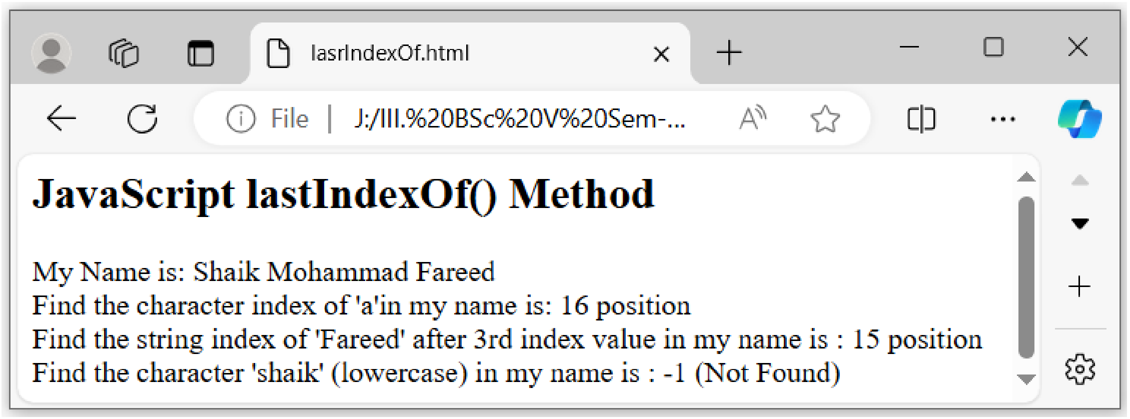
(4). lastIndexOf():

- The lastIndexOf() method returns the index (position) of the last occurrence of a specified value in a string.
- The lastIndexOf() method searches the string backward. (from the end to the beginning)
- The lastIndexOf() method returns the index from the beginning (position 0).
- The lastIndexOf() method returns -1 if the value is not found.
- The lastIndexOf() method is case sensitive.

Syntax: string.lastIndexOf("*searchvalue*", *start*)

```
<html>
<body>
<h2>JavaScript lastIndexOf() Method</h2>
<script>
var myname = "Shaik Mohammad Fareed";
document.write("My Name is: "+ myname + "<br>");
document.write("Find the character index of 'a' in my name is: "+ myname.lastIndexOf("a")+" position
                <br>");
document.write("Find the string index of 'Fareed' after 3rd index value in my name is :
                "+myname.indexOf("Fareed",3) +" position<br>");
document.write("Find the character 'shaik' (lowercase) in my name is : "+ myname.indexOf("shaik")+"
                (Not Found)<br>");
</script>
</body>
</html>
```

Output:



lasrIndexOf.html

J:/III.%20BSc%20V%20Sem-...

JavaScript lastIndexOf() Method

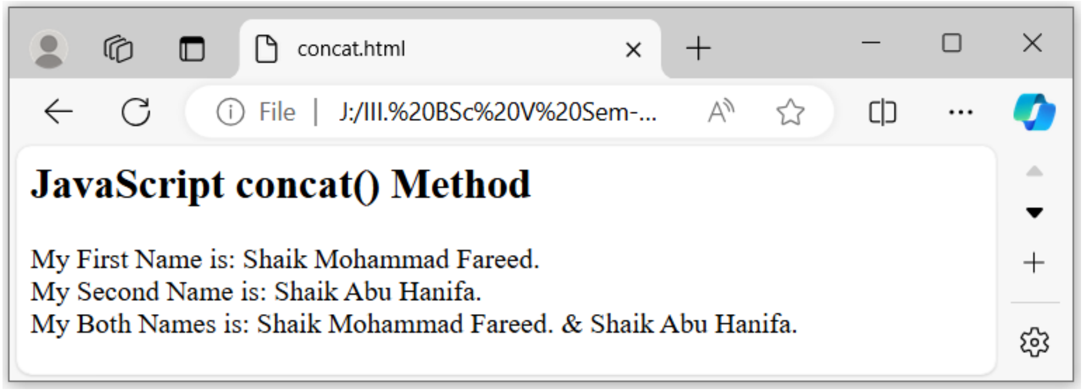
```
My Name is: Shaik Mohammad Fareed
Find the character index of 'a'in my name is: 16 position
Find the string index of 'Fareed' after 3rd index value in my name is : 15 position
Find the character 'shaik' (lowercase) in my name is : -1 (Not Found)
```

(5). concat(): The concat() method joins two or more strings. The concat() method does not change the existing strings. The concat() method returns a new string.

Syntax: string1.concat(string2, string3, ..., stringX)

```
<html>
<body>
<h2>JavaScript concat() Method</h2>
<script>
        var firstname = "Shaik Mohammad Fareed.";
        var secondname = "Shaik Abu Hanifa.";
        document.write("My First Name is: "+ firstname + "<br>");
        document.write("My Second Name is: "+ secondname + "<br>");
        document.write("My Both Names is: "+ firstname.concat(" & ", secondname)+ "<br>");
</script>
</body>
</html>
```

Output:



concat.html

J:/III.%20BSc%20V%20Sem-...

JavaScript concat() Method

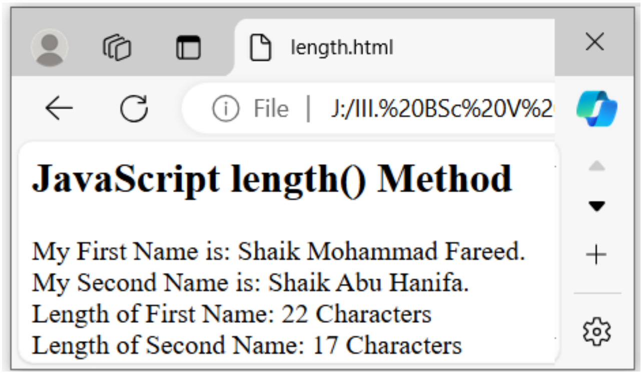
```
My First Name is: Shaik Mohammad Fareed.
My Second Name is: Shaik Abu Hanifa.
My Both Names is: Shaik Mohammad Fareed. & Shaik Abu Hanifa.
```

(6). length(): The length method returns the length of a string. The empty string is 0.

Syntax: string.length;

```
<html>
<body>
<h2>JavaScript length() Method</h2>
<script>
        var firstname = "Shaik Mohammad Fareed.";
        var secondname = "Shaik Abu Hanifa.";
        document.write("My First Name is: "+ firstname + "<br>");
        document.write("My Second Name is: "+ secondname + "<br>");
        document.write("Length of First Name: "+ firstname.length + " Characters <br>");
        document.write("Length of Second Name: "+ secondname.length + " Characters <br>");
</script>
</body>
</html>
```

Output:



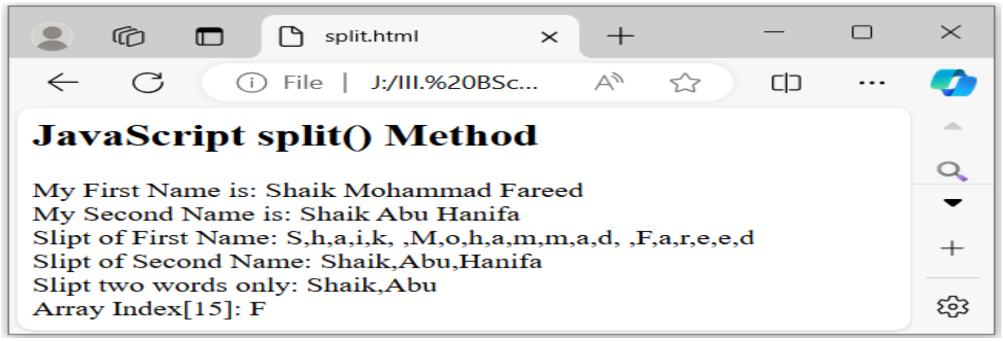
```
JavaScript length() Method

My First Name is: Shaik Mohammad Fareed.
My Second Name is: Shaik Abu Hanifa.
Length of First Name: 22 Characters
Length of Second Name: 17 Characters
```

(7). split(): The split() breaks the string a part whenever it encounters the character passed in as the first parameter the pieces of the string are stored in an array. Split() has an optional second parameter which is an integer value indicating how many of the pieces are to be stored in the array.

Syntax: string.split(“” [, limit]);

```
<html>
<body>
<h2>JavaScript split() Method</h2>
<script>
        var firstname = "Shaik Mohammad Fareed";
        var secondname = "Shaik Abu Hanifa";
        document.write("My First Name is: "+ firstname + "<br>");
        document.write("My Second Name is: "+ secondname + "<br>");
        document.write("Slipt of First Name: "+ firstname.split("") + "<br>");
        document.write("Slipt of Second Name: "+ secondname.split(" ") + "<br>");
        document.write("Slipt two words only: "+ secondname.split(" ", 2) + "<br>");
        document.write("Array Index[15]: "+ firstname[15].split() + "<br>");
</script>
</body>
</html>
```

Output:



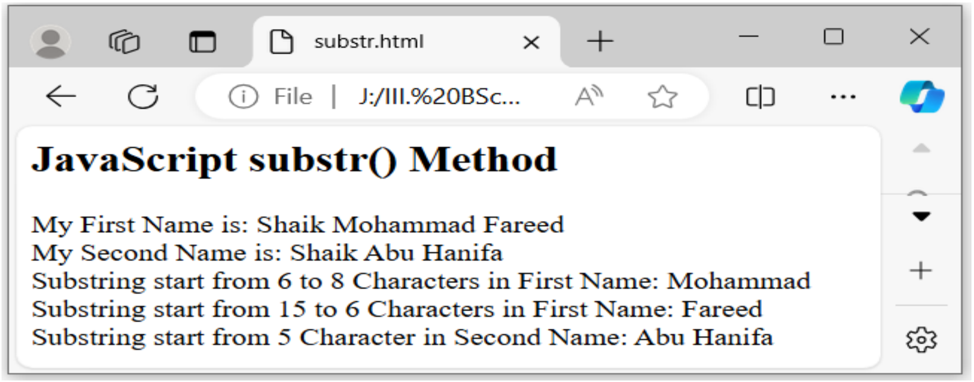
```
JavaScript split() Method

My First Name is: Shaik Mohammad Fareed
My Second Name is: Shaik Abu Hanifa
Slipt of First Name: S,h,a,i,k, ,M,o,h,a,m,m,a,d, ,F,a,r,e,e,d
Slipt of Second Name: Shaik,Abu,Hanifa
Slipt two words only: Shaik,Abu
Array Index[15]: F
```

(8). substr(): This function returns a substring which starts at the character indicated by the index parameter. The substring continues either to the end of the string or for the number of characters indicated by the length parameter. If the index is greater than the length of the string the nothing is returned. If it is negative then it is taken as the offset from the end of the string working backward along its length.

Syntax: substr(index [i, length]);

```
<html>
<body>
<h2>JavaScript substr() Method</h2>
<script>
        var firstname = "Shaik Mohammad Fareed";
        var secondname = "Shaik Abu Hanifa";
        document.write("My First Name is: "+ firstname + "<br>");
        document.write("My Second Name is: "+ secondname + "<br>");
        document.write("Substring start from 6 to 8 Characters  in First Name: "+
                        firstname.substr(6, 8) +"<br>");
        document.write("Substring start from 15 to 6 Characters in First Name: "+
                        firstname.substr(15, 6) + "<br>");
        document.write("Substring start from 5 Character  in Second Name: "+
                        secondname.substring(5) + "<br>");
</script>
</body>
</html>
```

Output:



```
JavaScript substr() Method

My First Name is: Shaik Mohammad Fareed
My Second Name is: Shaik Abu Hanifa
Substring start from 6 to 8 Characters in First Name: Mohammad
Substring start from 15 to 6 Characters in First Name: Fareed
Substring start from 5 Character in Second Name: Abu Hanifa
```

(9). toLowerCase(): Converts all characters in the string to lowercase.

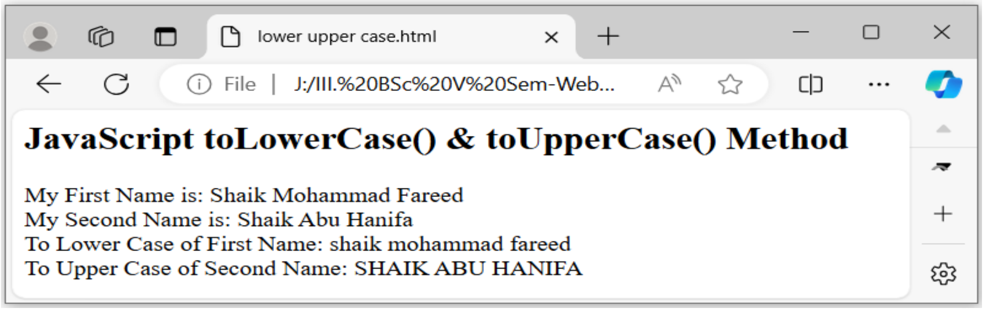
Syntax: string.toLowercase();

(10). toUpperCase(): Converts all characters in the string to uppercase.

Syntax: string.toUppercase();

```
<html>
<body>
<h2>JavaScript toLowerCase() & toUpperCase() Method</h2>
<script>
        var firstname = "Shaik Mohammad Fareed";
        var secondname = "Shaik Abu Hanifa";
        document.write("My First Name is: "+ firstname + "<br>");
        document.write("My Second Name is: "+ secondname + "<br>");
        document.write("To Lower Case of First Name: "+ firstname.toLowerCase() + "<br>");
        document.write("To Upper Case of Second Name: "+ secondname.toUpperCase() + "<br>");
</script>
</body>
</html>
```

Output:



JavaScript toLowerCase() & toUpperCase() Method

```
My First Name is: Shaik Mohammad Fareed
My Second Name is: Shaik Abu Hanifa
To Lower Case of First Name: shaik mohammad fareed
To Upper Case of Second Name: SHAIK ABU HANIFA
```

(Q) Explain about Mathematical Functions / Math Objects in JavaScript?

JavaScript Math object is used to perform mathematical operations on numbers. All the properties of Math are static and unlike other objects, it does not have a constructor.

Mathematical Functions and values are part of a built-in. JavaScript object called Math. All functions and attributes used in complex mathematics must be accessed via this object.

Example: var area = Math.PI.*(r, r); or var result = Math.ceil(area);

1. abs(value): This function returns the absolute value of a number.

Syntax: Math.abs(value);

2. ceil(value): This function returns the rounds a number Rounded Up to the nearest integer value.

Syntax: Math.ceil(value);

3. floor(value): This function returns the rounds a number Rounded Down to the nearest integer value.

Syntax: Math.floor(value);

4. round(value): This function returns the rounds a number to the nearest integer. If 2.49 will be rounded down (2) and 2.50 will be rounded up (3).

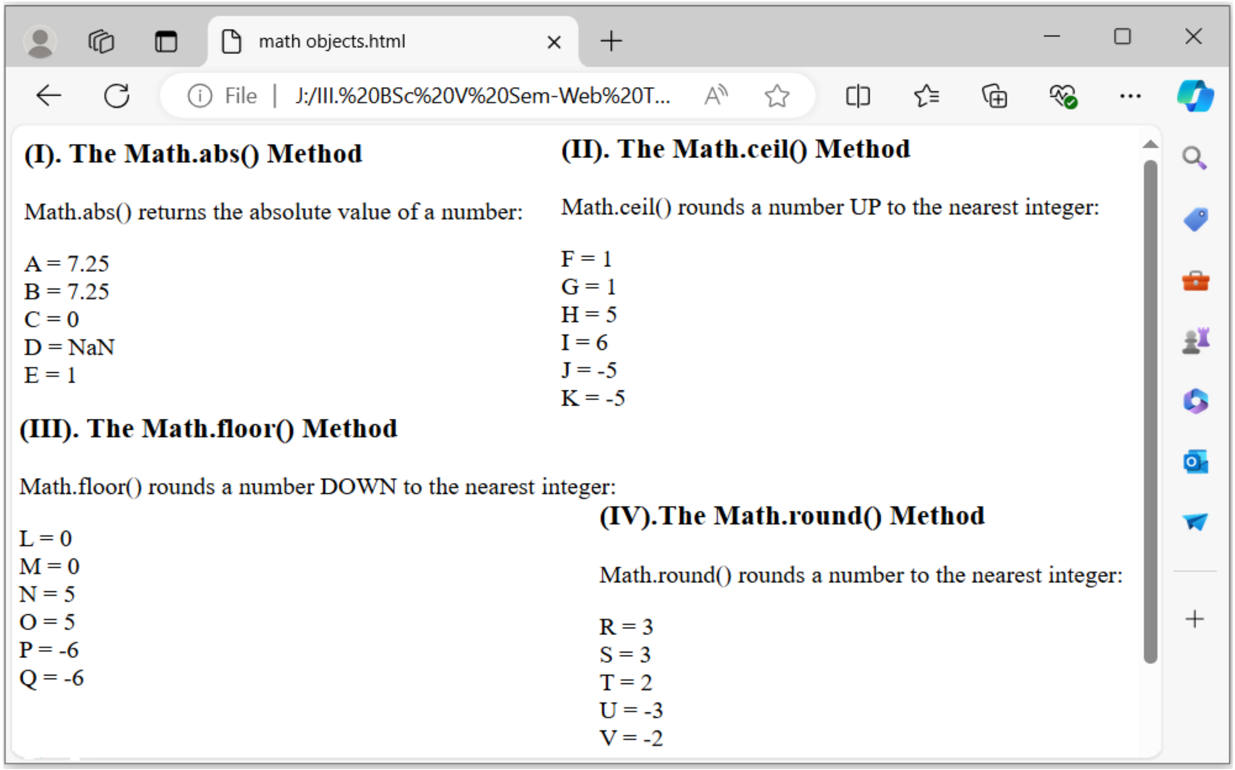
Syntax: Math.round(value)

Example:

```
<html>
<body>
<h3> (I). The Math.abs() Method</h3>
<p>Math.abs() returns the absolute value of a number:</p>
<p id="smf"></p>
<script>
        let a = Math.abs(7.25);
        let b = Math.abs(-7.25);
        let c = Math.abs(null);
        let d = Math.abs("Fareed");
        let e = Math.abs(2-3);
        document.getElementById("smf").innerHTML = "A = " + a + "<br>" + "B = " + b + "<br>" +
                                        "C = " + c + "<br>" + "D = " + d + "<br>" + "E = " + e;
```

```
</script>

<h3> (II). The Math.ceil() Method</h3>
<p>Math.ceil() rounds a number UP to the nearest integer:</p>
<p id="shaik"></p>
<script>
        let f = Math.ceil(0.60);
        let g = Math.ceil(0.40);
        let h = Math.ceil(5);
        let i = Math.ceil(5.1);
        let j = Math.ceil(-5.1);
        let k = Math.ceil(-5.9);
        document.getElementById("shaik").innerHTML =  "F = " + f + "<br>" + "G = " + g + "<br>" +
                        "H = " + h + "<br>" + "I = " + i + "<br>" + "J = " + j + "<br>" + "K = " + k;
</script>

<h3> (III). The Math.floor() Method</h3>
<p>Math.floor() rounds a number DOWN to the nearest integer:</p>
<p id="demo"></p>
<script>
        let l = Math.floor(0.60);
        let m = Math.floor(0.40);
        let n = Math.floor(5);
        let o = Math.floor(5.1);
        let p = Math.floor(-5.1);
        let q = Math.floor(-5.9);
        document.getElementById("demo").innerHTML = "L = " + l + "<br>" + "M = " + m + "<br>" +
                        "N = " + n + "<br>" + "O = " + o + "<br>" + "P = " + p + "<br>" + "Q = " + q;
</script>

<h3> (IV). The Math.round() Method</h3>
<p>Math.round() rounds a number to the nearest integer:</p>
<p id="mca"></p>
<script>
        let r = Math.round(2.60);
        let s = Math.round(2.50);
        let t = Math.round(2.49);
        let u = Math.round(-2.60);
        let v = Math.round(-2.49);
        document.getElementById ("mca").innerHTML = “R = " + r + "<br>" + "S = " + s + "<br>" +
                                        "T = " + t + "<br>" + "U = " + u + "<br>" + "V = " + v;
</script>
</body>
</html>
```

Output:



```
(I). The Math.abs() Method

Math.abs() returns the absolute value of a number:

A = 7.25
B = 7.25
C = 0
D = NaN
E = 1

(II). The Math.ceil() Method

Math.ceil() rounds a number UP to the nearest integer:

F = 1
G = 1
H = 5
I = 6
J = -5
K = -5

(III). The Math.floor() Method

Math.floor() rounds a number DOWN to the nearest integer:

L = 0
M = 0
N = 5
O = 5
P = -6
Q = -6

(IV).The Math.round() Method

Math.round() rounds a number to the nearest integer:

R = 3
S = 3
T = 2
U = -3
V = -2
```

5. sin(value): This function returns the Sine of the given number. The value returned by sin() method ranges between -1 to 1, which represents the Sine of the angle.

Syntax: Math.sin(value);

6. cos(value): This function returns the Cosine of the given number. The value returned by cos() method ranges -1 to 1, which represents the cosine of the angle.

Syntax: Math.cos(num);

7. tag(value): This function method returns the tangent of the given number that represents the tangent of an angle.

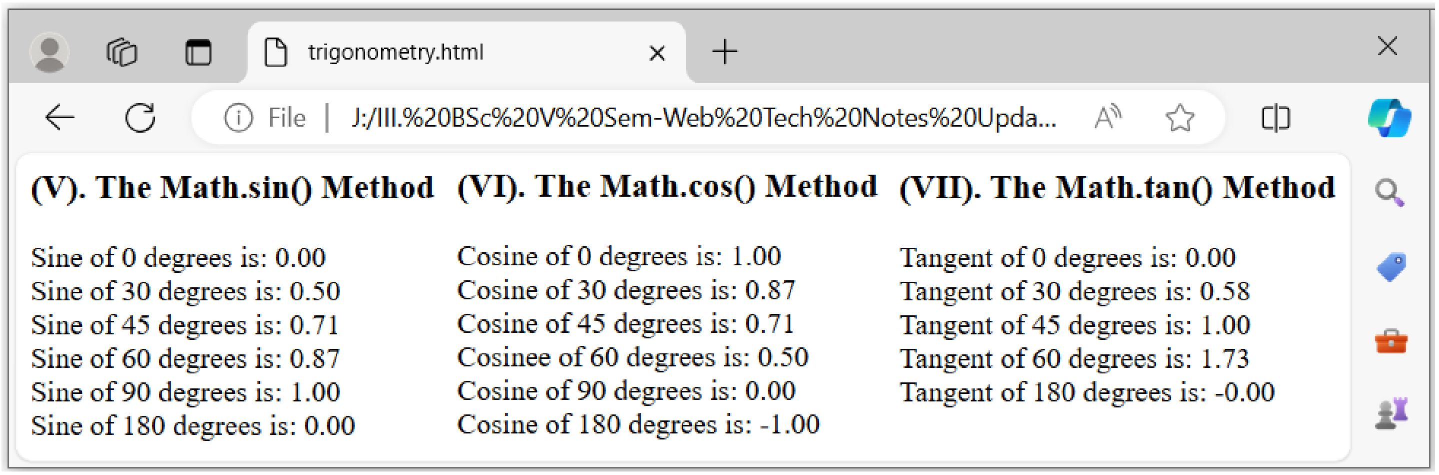
Syntax: Math.tan(value);

Example:

```
<html>
<body>
<h3> (V). The Math.sin() Method</h3>
<p id="shaik"></p>
<script>
document.getElementById("shaik").innerHTML="Sine of 0 degrees is: "+Math.sin(0).toFixed(2) +"<br>" +
                                "Sine of 30 degrees is: " + Math.sin(Math.PI/6).toFixed(2) + "<br>" +
                                "Sine of 45 degrees is: " + Math.sin(Math.PI/4).toFixed(2) + "<br>" +
                                "Sine of 60 degrees is: " + Math.sin(Math.PI/3).toFixed(2) +"<br>" +
                                "Sine of 90 degrees is: " + Math.sin(Math.PI/2).toFixed(2) +"<br>" +
                                "Sine of 180 degrees is: " + Math.sin(Math.PI).toFixed(2);
</script>
```

```
<h3> (VI). The Math.cos() Method</h3>
<p id="smf"></p>
<script>
document.getElementById("smf").innerHTML="Cosine of 0 degrees is: "+Math.cos(0).toFixed(2)+"<br>" +
                              "Cosine of 30 degrees is: " + Math.cos(Math.PI/6).toFixed(2) + "<br>" +
                              "Cosine of 45 degrees is: " + Math.cos(Math.PI/4).toFixed(2) + "<br>" +
                              "Cosine of 60 degrees is: " + Math.cos(Math.PI/3).toFixed(2) +"<br>" +
                              "Cosine of 90 degrees is: " + Math.cos(Math.PI/2).toFixed(2) +"<br>" +
                              "Cosine of 180 degrees is: " + Math.cos(Math.PI).toFixed(2);
</script>

<h3>(VII). The Math.tan() Method</h3>
<p id="abu"></p>
<script>
document.getElementById ("abu").innerHTML="Tangent 0 degrees is:"+Math.tan(0).toFixed(2)+"<br>" +
                              "Tangent of 30 degrees is: " + Math.tan(Math.PI/6).toFixed(2) + "<br>" +
                              "Tangent of 45 degrees is: " + Math.tan(Math.PI/4).toFixed(2) + "<br>" +
                              "Tangent of 60 degrees is: " + Math.tan(Math.PI/3).toFixed(2) +"<br>" +
                              "Tangent of 180 degrees is: " + Math.tan(Math.PI).toFixed(2);
</script>
</body> </html>
```

Output:



```
(V). The Math.sin() Method      (VI). The Math.cos() Method      (VII). The Math.tan() Method

Sine of 0 degrees is: 0.00      Cosine of 0 degrees is: 1.00     Tangent of 0 degrees is: 0.00
Sine of 30 degrees is: 0.50     Cosine of 30 degrees is: 0.87    Tangent of 30 degrees is: 0.58
Sine of 45 degrees is: 0.71     Cosine of 45 degrees is: 0.71    Tangent of 45 degrees is: 1.00
Sine of 60 degrees is: 0.87     Cosinee of 60 degrees is: 0.50   Tangent of 60 degrees is: 1.73
Sine of 90 degrees is: 1.00     Cosine of 90 degrees is: 0.00    Tangent of 180 degrees is: -0.00
Sine of 180 degrees is: 0.00    Cosine of 180 degrees is: -1.00
```

8. max(value1, value2): This is function returns the larger of its arguments.

Syntax: Math.max(value1, value2, value3,……);

9. min(value1, value2): This is function returns the smaller of its arguments.

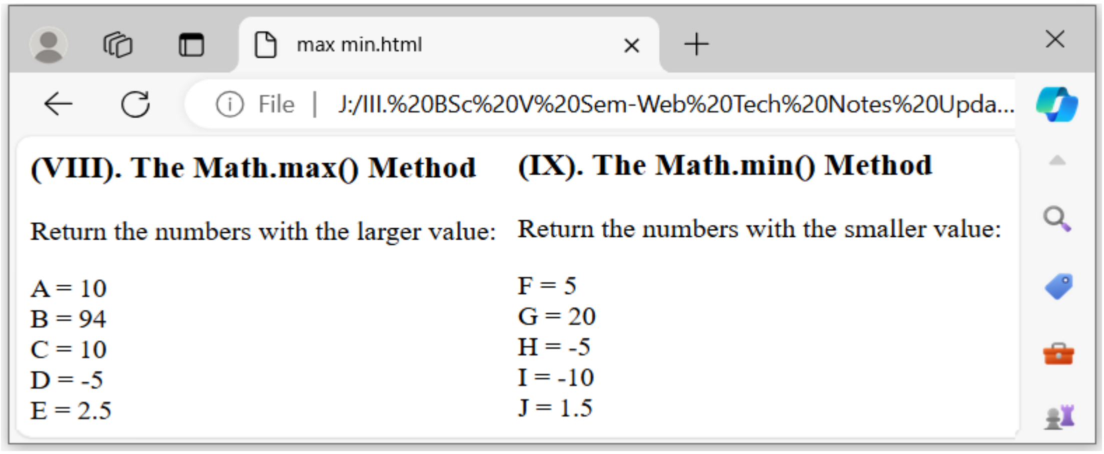
Syntax: Math.min(value1, value2, value3,……);

Example:

```
<html>
<body>
<h3> (VIII). The Math.max() Method</h3>
<p>Return the numbers with the larger value:</p>
<p id="shaik"></p>
```

```
<script>
        let a = Math.max(5, 10);
        let b = Math.max(94, 91, 20, 29, 87);
        let c = Math.max(-5, 10);
        let d = Math.max(-5, -10);
        let e = Math.max(1.5, 2.5);
        document.getElementById("shaik").innerHTML = "A = " + a + "<br>" + "B = " + b + "<br>" +
                                          "C = " + c + "<br>" + "D = " + d + "<br>" + "E = " + e;
</script>

<h3> (IX). The Math.min() Method</h3>
<p>Return the numbers with the smaller value:</p>
<p id="abu"></p>
<script>
        let f = Math.min(5, 10);
        let g = Math.min(94, 91, 20, 29, 87);
        let h = Math.min(-5, 10);
        let i = Math.min(-5, -10);
        let j = Math.min(1.5, 2.5);
        document.getElementById("abu").innerHTML ="F = " + f + "<br>" + "G = " + g + "<br>" +
                                          "H = " + h + "<br>" + "I = " + i + "<br>" + "J = " + j;
</script>                                                                                    </body>
</html>
```

Output:



```
(VIII). The Math.max() Method              (IX). The Math.min() Method

Return the numbers with the larger value:  Return the numbers with the smaller value:

A = 10                                     F = 5
B = 94                                     G = 20
C = 10                                     H = -5
D = -5                                     I = -10
E = 2.5                                    J = 1.5
```

10. log(value): This function returns the natural logarithms (base E) of its argument.

Syntax: Math.log(value);

11. pow(value, power): This function returns the value of x to the power of y (x^y)

Syntax: Math.pow(value, power);

12. sqrt(value): This function returns the result of square root of the value.

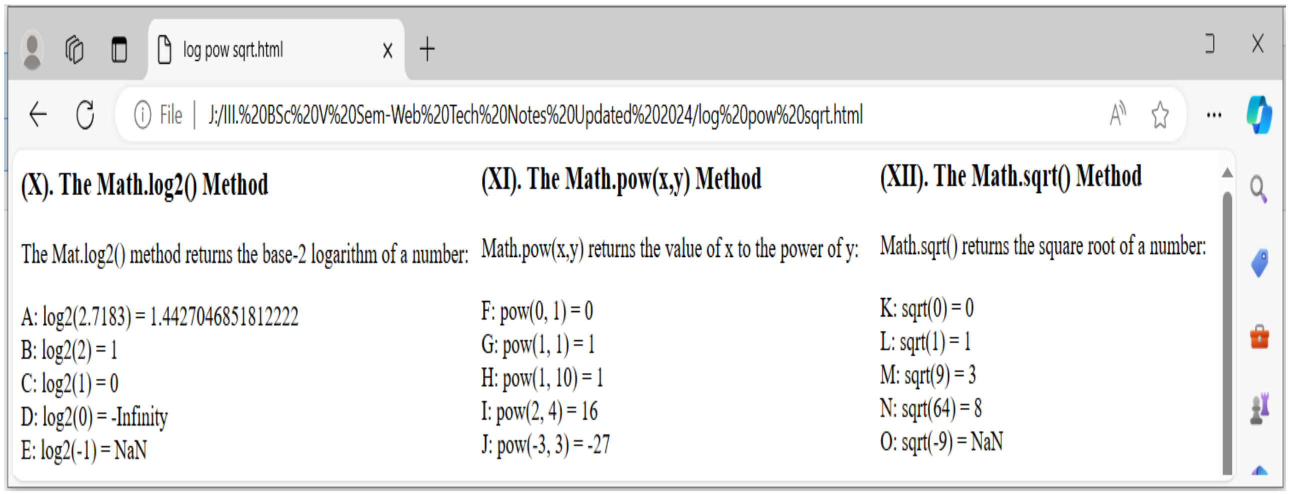
Syntax: Math.sqrt(value);

Example:

```
<html>
<body>
```

```
<h3> (X). The Math.log2() Method</h3>
<p>The Mat.log2() method returns the base-2 logarithm of a number:</p>
<p id="shaik"></p>
<script>
        let a = Math.log2(2.7183);
        let b = Math.log2(2);
        let c = Math.log2(1);
        let d = Math.log2(0);
        let e = Math.log2(-1);
document.getElementById("shaik").innerHTML = "A: log2(2.7183) = " + a + "<br>" + "B: log2(2) = " + b
+ "<br>" +"C: log2(1) = " + c + "<br>" + "D: log2(0) = " + d + "<br>" + "E: log2(-1) = " + e;
</script>

<h3> (XI). The Math.pow(x,y) Method</h3>
<p>Math.pow(x,y) returns the value of x to the power of y:</p>
<p id="abu"></p>
<script>
        let f = Math.pow(0, 1);
        let g = Math.pow(1, 1);
        let h = Math.pow(1, 10);
        let i = Math.pow(2, 4);
        let j = Math.pow(-3, 3);
document.getElementById("abu").innerHTML = "F: pow(0, 1) = " + f + "<br>" + "G: pow(1, 1) = " + g +
"<br>" +"H: pow(1, 10) = " + h + "<br>" + "I: pow(2, 4) = " + i + "<br>" + "J: pow(-3, 3) = " + j;
</script>

<h3> (XII). The Math.sqrt() Method </h3>
<p>Math.sqrt() returns the square root of a number:</p>
<p id="smf"></p>
<script>
        let k = Math.sqrt(0);
        let l = Math.sqrt(1);
        let m = Math.sqrt(9);
        let n = Math.sqrt(64);
        let o = Math.sqrt(-9);
document.getElementById("smf").innerHTML = "K: sqrt(0) = " + k + "<br>" + "L: sqrt(1) = " + l + "<br>"
+"M: sqrt(9) = " + m + "<br>" + "N: sqrt(64) = " + n + "<br>" + "O: sqrt(-9) = " + o ;
</script>
</body>
</html>
```

Output:


log pow sqrt.html
File | J:/III.%20BSc%20V%20Sem-Web%20Tech%20Notes%20Updated%202024/log%20pow%20sqrt.html

(X). The Math.log2() Method

The Mat.log2() method returns the base-2 logarithm of a number:

```
A: log2(2.7183) = 1.4427046851812222
B: log2(2) = 1
C: log2(1) = 0
D: log2(0) = -Infinity
E: log2(-1) = NaN
```

(XI). The Math.pow(x,y) Method

Math.pow(x,y) returns the value of x to the power of y:

```
F: pow(0, 1) = 0
G: pow(1, 1) = 1
H: pow(1, 10) = 1
I: pow(2, 4) = 16
J: pow(-3, 3) = -27
```

(XII). The Math.sqrt() Method

Math.sqrt() returns the square root of a number:

```
K: sqrt(0) = 0
L: sqrt(1) = 1
M: sqrt(9) = 3
N: sqrt(64) = 8
O: sqrt(-9) = NaN
```

UNIT – II:
Chapter -1: HTML DOM (Document Object Model)

(Q) Explain about DOM in JavaScript?

Definition of DOM: "The W3C (World Wide Web Consortium) Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document".

JavaScript HTML DOM: With the HTML DOM, JavaScript can access and change all the elements of an HTML document. When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page. The DOM defines a standard for accessing documents:

HTML DOM: The HTML DOM is a standard object model and programming interface for HTML. It defines:

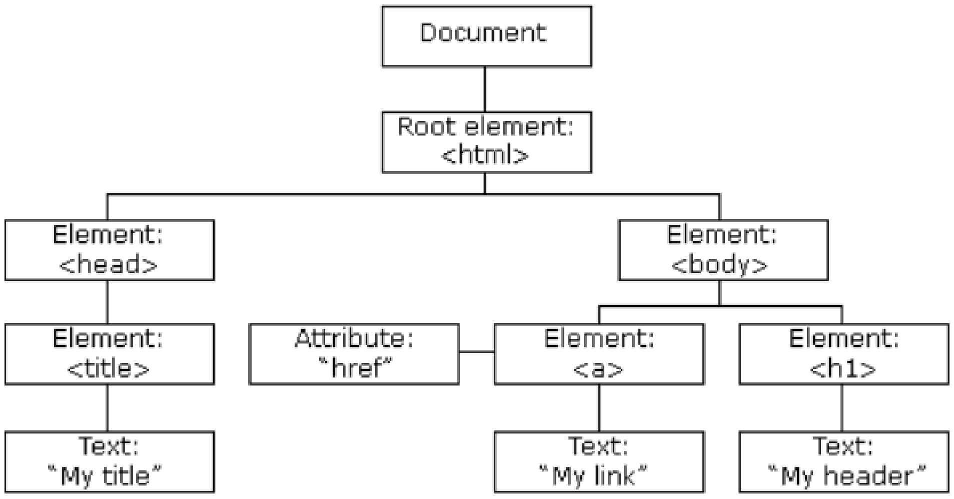
- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements
- **In other words:** The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

NOTE: Many times DHTML is confused with being a language like HTML but it is not. It must be kept in mind that it is an interface or browsers enhancement feature which makes it possible to access the object model through JavaScript language and hence make the webpage more interactive.

The HTML DOM (Document Object Model) is a programming interface that represents the structure of a web page in a way that programming languages like JavaScript can understand and manipulate.

The HTML DOM (Document Object Model): When a web page is loaded, the browser creates a DOM (Document Object Model) of the page. The HTML DOM model is constructed as a Tree of Objects:

The HTML DOM Tree of Objects:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page

- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

(Q) Explain about the JavaScript Document Object?

The JavaScript document object represents your web page. If you want to access any element in an HTML page, you always start with accessing the document object. Below are some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements:

Method	**Description**
document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name
document.querySelectorAll()	Find elements by CSS selectors

Changing HTML Elements:

Property	**Description**
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element
Method	**Description**
element.setAttribute(attribute, value)	Change the attribute value of an HTML element

Adding and Deleting Elements:

Method	**Description**
document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element
document.appendChild(element)	Add an HTML element
document.replaceChild(new, old)	Replace an HTML element
document.write(text)	Write into the HTML output stream

(Q) Explain how to Finding HTML Elements using JavaScript?

JavaScript, you want to manipulate HTML elements. To do so, you have to find the elements first. There are several ways to do this:

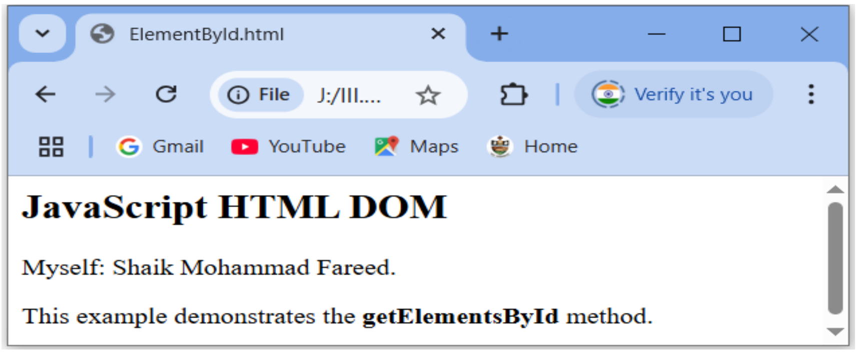
Method	**Description**
document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name
document.querySelectorAll()	Find elements by CSS selectors

(1). Finding HTML Element by Id: The easiest way to find an HTML element in the DOM, is by using the element id. This example finds the element with id="SMF":

Example:

```
<html>
<body>
        <h2>JavaScript HTML DOM</h2>
        <p id="SMF"> Shaik Mohammad Fareed.</p>
        <p>This example demonstrates the <b>getElementsById</b> method.</p>
        <p id="SMF"></p>
        <script>
                const element = document.getElementById("SMF");
                document.getElementById("SMF").innerHTML = "Myself: " + element.innerHTML;
        </script>
</body>
</html>
```

Output:



JavaScript HTML DOM

Myself: Shaik Mohammad Fareed.

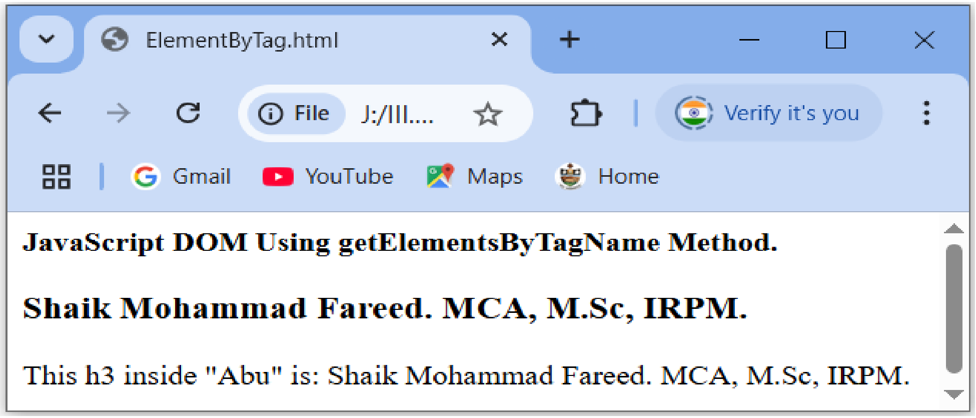
This example demonstrates the **getElementsById** method.

NOTE: If the element is found, the method will return the element as an object (in element).
If the element is not found, element will contain null.

(2). Finding HTML Elements by Tag Name: This example finds <h3> element by tag name:

```
<html>
<body>
<head> <b>JavaScript DOM Using getElementsByTagName Method. </b></head>
        <div id="Abu">
                <h3>Shaik Mohammad Fareed. MCA, M.Sc, IRPM.</h3>
        </div>
        <p id="SMF"></p>
        <script>
                const x = document.getElementById("Abu");
                const y = x.getElementsByTagName("h3");
                document.getElementById("SMF").innerHTML = 'This h3 inside "Abu" is: ' + y[0].innerHTML;
        </script>
</body>
</html>
```

Output:



JavaScript DOM Using getElementsByTagName Method.

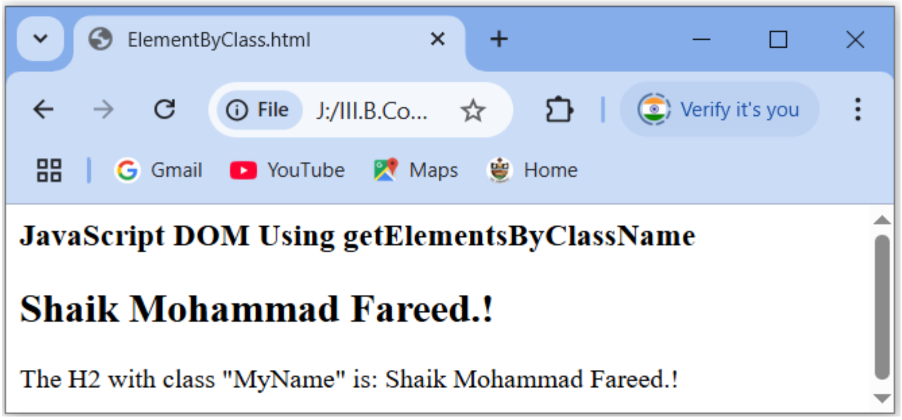
Shaik Mohammad Fareed. MCA, M.Sc, IRPM.

This h3 inside "Abu" is: Shaik Mohammad Fareed. MCA, M.Sc, IRPM.

(3). Finding HTML Elements by Class Name: If you want to find all HTML elements with the same class name, use getElementsByClassName(). This example returns a element with class="MyName".

```
<html>
<body>
 <h3>JavaScript DOM Using getElementsByClassName </h3>
 <h2 class ="MyName">Shaik Mohammad Fareed.!</h2>
 <p id="SMF"></p>
        <script>
        const x = document.getElementsByClassName("MyName");
        document.getElementById("SMF").innerHTML = 'The H2 with class "MyName" is: ' + x[0].innerHTML;
        </script>
</body>
</html>
```

Output:

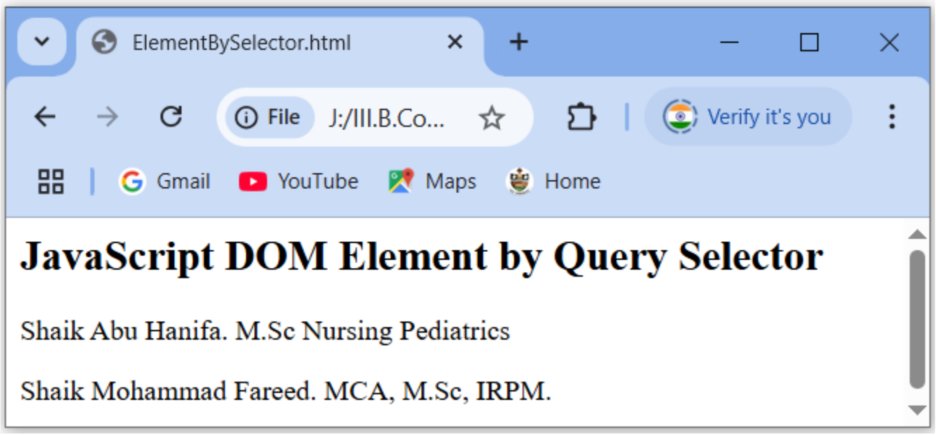


```
JavaScript DOM Using getElementsByClassName

Shaik Mohammad Fareed.!

The H2 with class "MyName" is: Shaik Mohammad Fareed.!
```

(4). Finding HTML Elements by CSS Selectors: If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method. This example returns a list of all <p> elements with class="SAB".

```
<html>
<body>
        <h2>JavaScript DOM Element by Query Selector </h2>
        <p class="SAB">Shaik Abu Hanifa. M.Sc Nursing Pediatrics</p>
        <p class="SAB">Shaik Mohammad Fareed. MCA, M.Sc, IRPM.</p>
        <p id="SMF"></p>
        <script>
                const x = document.querySelectorAll("p.SAB");
        </script>
</body>
</html>
```

Output:



```
JavaScript DOM Element by Query Selector

Shaik Abu Hanifa. M.Sc Nursing Pediatrics

Shaik Mohammad Fareed. MCA, M.Sc, IRPM.
```

(Q) Explain how to Changing HTML Elements using JavaScript?

To change HTML elements, you can use JavaScript to manipulate the Document Object Model (DOM). This involves identifying the element, then modifying its content, attributes, or even its tag name. Methods like innerHTML, textContent, and setAttribute are commonly used for these changes.

Property	**Description**
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element
Method	**Description**
element.setAttribute(attribute, value)	Change the attribute value of an HTML element

1. Changing Element Content:

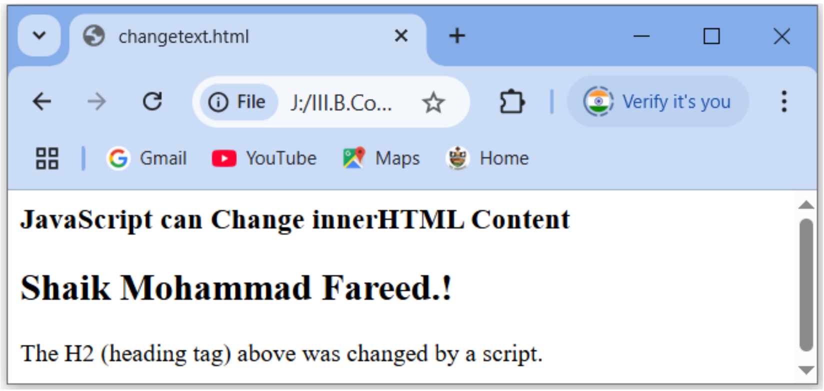
(a). innerHTML: This property allows you to replace the entire HTML content of an element, including tags and other elements.

Syntax: document.getElementById("myElement").innerHTML = "<b>New content with HTML tags!</b>";

Example: This example changes the content of a <h2> element.

```
<html>
<body>
	<h3>JavaScript can Change innerHTML Content</h3>
	<h2 id="SMF">Shaik Abu Hanifa!</h2>
	<script>
		document.getElementById("SMF").innerHTML = "Shaik Mohammad Fareed.!";
	</script>
	<p>The H2 (heading tag) above was changed by a script.</p>
</body>
</html>
```

Output:



JavaScript can Change innerHTML Content

Shaik Mohammad Fareed.!

The H2 (heading tag) above was changed by a script.

Example explained:

- The HTML document above contains a <h2> element with id="SMF"
- We use the HTML DOM to get the element with id="SMF"
- A JavaScript changes the content (innerHTML) of that element to "New text!" (Shaik Mohammad Fareed).

(b). textContent: This property changes only the text content of an element, stripping out any HTML tags.

Syntax: document.getElementById("myElement").textContent = "New text content";

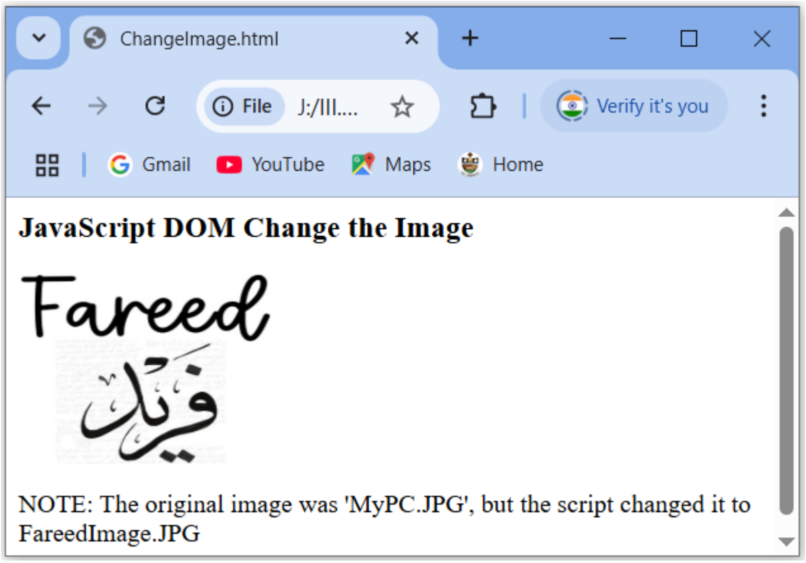
2. Changing the Value of an Attribute: To change the value of an HTML attribute, use this syntax:

Syntax: document.getElementById(id).attribute = new value

Example: This example changes the value of the src attribute of an <img> element:

```
<html>
<body>
<h3>JavaScript DOM Change the Image</h3>
        <img id="MyName" src="J:\III.B.Com V Sem Web Application Development Tools w.e.f 2025
          Major\FareedImage.JPG"" width="160" height="120">
        <script>
                document.getElementById("PC").src = "J:\III.B.Com V Sem Web Application Development
                Tools w.e.f 2025 Major\MyPC.JPG";
        </script>
<p>NOTE: The original image was 'MyPC.JPG', but the script changed it to FareedImage.JPG</p>
</body>
</html>
```

Output:



JavaScript DOM Change the Image

NOTE: The original image was 'MyPC.JPG', but the script changed it to FareedImage.JPG

setAttribute(): Use this method to modify the value of an element's attributes.

Syntax: document.getElementById("myImage").setAttribute("src", "newImage.jpg");

3. Changing Element Styles:

style property: You can directly modify CSS properties of an element using the style property.

Syntax: document.getElementById("myElement").style.color = "red";
document.getElementById("myElement").style.fontSize = "20px";

(Q) Explain how to Adding or Removing HTML Elements using JavaScript?

Adding and Deleting Elements:

Method	Description
document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element
document.appendChild(element)	Add an HTML element
document.replaceChild(new, old)	Replace an HTML element
document.write(text)	Write into the HTML output stream

1. Adding or Removing Elements:

createElement(): Create a new HTML element.

Syntax:
```
const newElement = document.createElement("div");
newElement.textContent = "This is a new div";
```

2. appendChild(): Add the new element as a child to an existing element.

Syntax:
```
document.getElementById("parent").appendChild(newElement);
```

3. removeChild(): Remove an existing element.

Syntax:
```
const elementToRemove = document.getElementById("elementToRemove");
elementToRemove.parentNode.removeChild(elementToRemove);
```

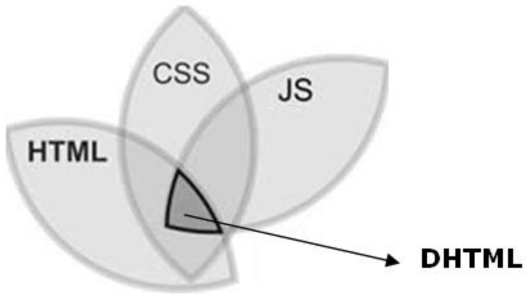
4. Changing the Tag Name:

outerHTML: While you can't directly change the tagName, you can replace the entire element with a new one using outerHTML.

Syntax:
```
const element = document.getElementById("myElement");
element.outerHTML = "<p>This is now a paragraph element</p>";
```

Chapter -2: DHTML with JavaScript

Introduction of DHTML: DHTML is generally Dynamic Hypertext Markup Language (DHTML). It is a new way of looking at and controls the standard HTML codes and commands. DHTML is a collection of technologies used to create interactive and animated website. DHTML use more control over the HTML elements. DHTML is an incorporate a client side scripting such as a JavaScript.



Additional Feature of DHTML:

HTML	DHTML
HTML is a standard language for web page design	DHTML is a collection of various Technologies
Slowly working for client side.	Fast working for client side.
HTML is a Static in nature.	DHTML is Dynamically in nature.
HTML pages do not require any processing from browsers.	DHTML require processing file browser which changes its look and feel.
No need for Database Connectivity.	DHTML may require Database Connectivity.
HTML cannot have any server side code.	DHTML may contain server side code.

Why Use DHTML: DHTML makes a webpage dynamic but JavaScript also does, the question arises that what different does DHTML do? So the answer is that DHTML has the ability to change a webpages look, content and style once the document has loaded on our demand without changing or deleting everything already existing on the browser's webpage. DHTML can change the content of a webpage on demand without the browser having to erase everything else, i.e. being able to alter changes on a webpage even after the document has completely loaded.

Key Features:

- Tags and their properties can be changed using DHTML.
- It is used for real-time positioning.
- Dynamic fonts can be generated using DHTML.
- It is also used for data binding.
- It makes a webpage dynamic and be used to create animations, games, applications along with providing new ways of navigating through websites.
- The functionality of a webpage is enhanced due to the usage of low-bandwidth effect by DHTML.
- DHTML also facilitates the use of methods, events, properties, and codes.

Advantages:

- Sizes of the files are compact in compared to other interactional media like Flash or Shockwave, and it downloads faster.
- It is supported by big browser manufacturers like Microsoft and Netscape.
- Highly flexible and easy to make changes.
- Viewer requires no extra plug-ins for browsing through the webpage that uses DHTML; they do not need any extra requirements or special software to view it.
- User time is saved by sending less number of requests to the server. As it is possible to modify and replace elements even after a page is loaded, it is not required to create separate pages for changing styles which in turn saves time in building pages and also reduces the number of requests that are sent to the server.

- It has more advanced functionality than a static HTML. It is capable of holding more content on the web page at the same time.

Disadvantages:

- It is not supported by all the browsers. It is supported only by recent browsers such as Netscape 6, IE 5.5, and Opera 5 like browsers.
- Learning of DHTML requires a lot of pre-requisites languages such as HTML, CSS, JS, etc should be known to the designer before starting with DHTML which is a long and time-consuming in itself.
- Implementation of different browsers is different. So if it worked in one browser, it might not necessarily work the same way in another browser.
- Even after being great with functionality, DHTML requires a few tools and utilities that are some expensive. For example, the DHTML text editor, Dreamweaver. Along with it the improvement cost of transferring from HTML to DHTML makes cost rise much higher.

Difference between HTML and DHTML:

HTML (Hypertext Markup language)	DHTML (Dynamic Hypertext Markup language)
HTML is simply a markup language.	DHTML is not a language, but it is a set of technologies of web development.
It is used for developing and creating web pages.	It is used for creating and designing the animated and interactive web sites or pages.
This markup language creates static web pages.	This concept creates dynamic web pages.
HTML sites are slow upon client-side technologies	It may contain the code of server-side scripting. DHTML sites are comparatively faster.
The files of HTML are stored with the .html or .htm extension in a system.	The files of DHTML are stored with the .dhtm extension in a system.
A simple page which is created by a user without using the scripts or styles called as an HTML page.	A page which is created by a user using the HTML, CSS, DOM, and JavaScript technologies called a DHTML page.
This markup language does not need database connectivity.	This concept needs database connectivity because it interacts with users.

VIMP*************(Q) Discuss about Data Validation with an Example?**

Data Validation is a process that is used to compare a body of data to the requirement in a set of document acceptance criteria is known as 'Data Validation'. It checks the data is sensible before it is processed. It determined to what extent analytical and other forms of data are reliable, accurate and usable in various context. Validation is done to ensure that programs and processes operate on correct and accurate data.

In general when the user enters data, the script performs the validation and an error occurs because user entering wrong data like entering a space or character other than a digit or a letter into a user name.

JavaScript provides a way to validate forms data o the client computer before sending it to the web server. Form Validation generally performs two functions.

1. Basic Validation. (Verification)
2. Data Format Validation. (Validation)

1. Basic Validation: First of all the form must be checked to make sure data was entered into each from field that required it. This would need just loop through each field in the form and check for Data.

2. Data Format Validation: Second the data that is entered must be checked for correct form and value. This would need to be put more logic to text correctness of data.

Data Validation: Data validation is the process of ensuring that user input is clean, correct, and useful. Most often, the purpose of data validation is to ensure correct user input.

Typical validation tasks are:

- The user filled in all required fields?
- The user entered a valid date?
- The user entered text in a numeric field?

Types of Validation Methods: Validation can be defined by many different methods, and deployed in many different ways.

- Client side validation is performed by a web browser, before input is sent to a web server.
- Server side validation is performed by a web server, after input has been sent to the server.

HTML Constraint Validation: HTML constraint validation is based on:

- Constraint validation HTML Input Attributes
- Constraint validation CSS Pseudo Selectors
- Constraint validation DOM Properties and Methods

Constraint Validation HTML Input Attributes:

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

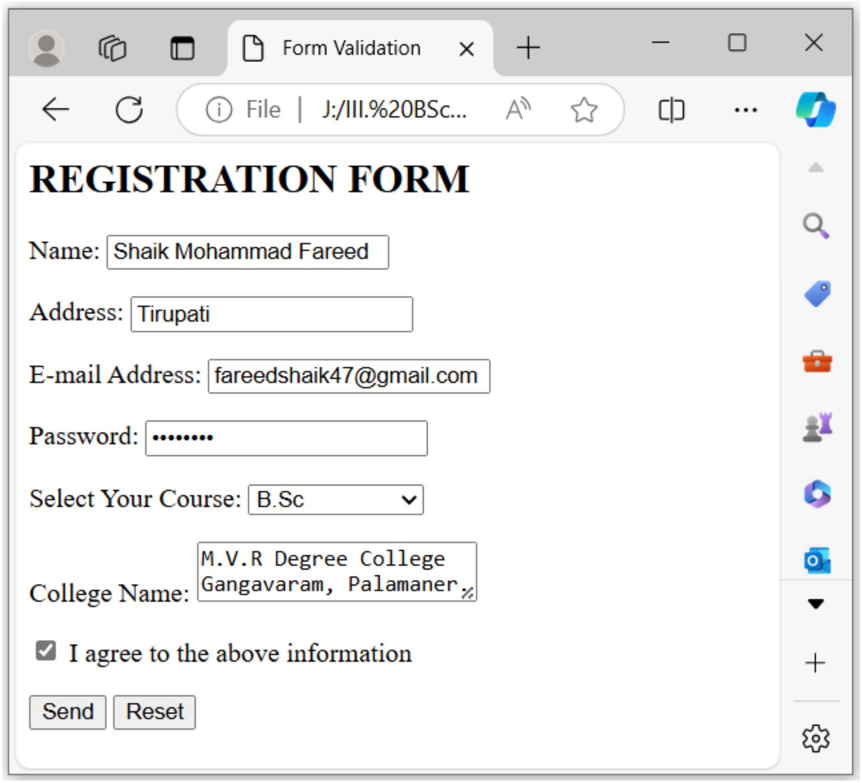
Constraint Validation CSS Pseudo Selectors:

Selector	Description
:disabled	Selects input elements with the "disabled" attribute specified
:invalid	Selects input elements with invalid values
:optional	Selects input elements with no "required" attribute specified
:required	Selects input elements with the "required" attribute specified
:valid	Selects input elements with valid values

Example for Validate the Form elements in JavaScript?

```
<html>
<head>
  <title> Form Validation </title>
  </head>
<body>
  <h2> REGISTRATION FORM </h2>
  <form name="RegForm" onsubmit="return validateForm()">
    <p>
      <label for="name">Name:</label>
      <input type="text" id="name" name="Name" placeholder="Enter your full name" />
     </p>
    <p>
      <label for="address">Address:</label>
      <input type="text" id="address" name="Address" placeholder="Enter your address" />
    </p>
    <p>
      <label for="email">E-mail Address:</label>
      <input type="text" id="email" name="EMail" placeholder="Enter your email" />
    </p>
    <p>
      <label for="password">Password:</label>
      <input type="password" id="password" name="Password" />
    </p>
    <p> <label for="subject"> Select Your Course: </label>
      <select id="subject" name="Subject">
        <option value=""> Select Course </option>
        <option value="B.Sc">B.Sc </option>
        <option value="BBA">BBA </option>
        <option value="BCA">BCA </option>
        <option value="B.Com">B.Com </option>
      </select>
    </p>
    <p>
      <label for="comment"> College Name: </label>
      <textarea id="comment" name="Comment"></textarea>
    </p>
    <p>
      <input type="checkbox" id="agree" name="Agree" />
      <label for="agree"> I agree to the above information </label>
    </p>
    <p>
      <input type="submit" value="Send" name="Submit" />
      <input type="reset" value="Reset" name="Reset" />
    </p>
  </form>
  <script src="script.js"></script></body>
</html>
```

Output:



REGISTRATION FORM

Name: Shaik Mohammad Fareed

Address: Tirupati

E-mail Address: fareedshaik47@gmail.com

Password: ••••••••

Select Your Course: B.Sc

College Name: M.V.R Degree College Gangavaram, Palamaner

☑ I agree to the above information

Send Reset

(Q) Explain the Regular Expressions in JavaScript?

A Regular Expression is a sequence of characters that forms a search pattern. The JavaScript RegExp class represents regular expressions. Regular Expressions are used to perform pattern-matching and "Search-and-Replace" functions on text.

Syntax: var text RegExp(pattern, modifiers); (or) var text =/pattern/modifiers;

- **Pattern:** A string that specifies the pattern of the regular expression or another regular expression.
- **Modifiers:** Modifiers specify if a search should be global, case-sensitive.

Using String Methods: In JavaScript, regular expressions are often used with the two string methods: search() and replace().

- The search() method uses an expression to search for a match, and returns the position of the match.
- The replace() method returns a modified string where the pattern is replaced.

Several modifiers are available that can make your work with regexps much easier, like case sensitivity, searching in multiple lines etc.

Modifier	Description
i	Perform case-insensitive matching.
m	Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary
g	Perform a global match that is, find all matches rather than stopping after the first match.

Bracket: Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Expression	Description
[...]	Any one character between the brackets.
[^...]	Any one character not between the brackets.
[0-9]	It matches any decimal digit from 0 through 9.
[a-z]	It matches any character from lowercase a through lowercase z.
[A-Z]	It matches any character from uppercase A through uppercase Z.
[a-Z]	It matches any character from lowercase a through uppercase Z.

The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from b through v.

Character: Following examples will clear your concepts about matching characters.

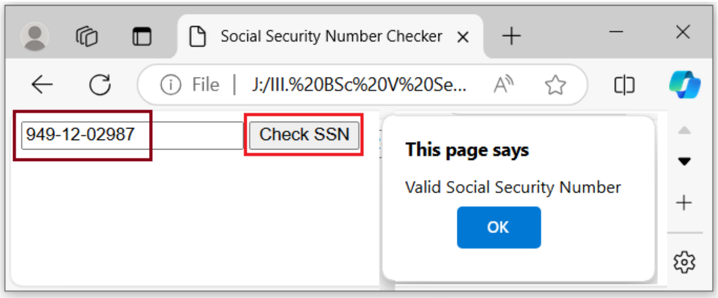
Expression	Description
[^a-z A-Z]	It matches any string not containing any of the characters ranging from a through z and A through Z.
p.p	It matches any string containing p, followed by any character, in turn followed by another p.
^.{2}$	It matches any string containing exactly two characters.
<b>(.*)</b>	It matches any string enclosed within <b> and </b>.
p(hp)*	It matches any string containing a p followed by zero or more instances of the sequence hp.

Example of Regular Expression in JavaScript?

```
<html>
<head>
<title> Social Security Number Checker </title>
<script>
        var reg_ssn = /^[0-9]{3}[\-]?[0-9]{2}[\-]?[0-9]{5}$/;
        function checkSsn(ssn)
        {
                if(reg_ssn.test(ssn))
                {
                        alert("Valid Social Security Number");
                }
        else
        {
                alert("Invalid Social Security Number");
        }
        }
</script>
</head>
<body>
<form onsubmit="return false;">
        <input type="text" name="ssn" size="20">
        <input type="button" value="Check SSN" onclick="checkSsn(this.form.ssn.value);">
```

```
</form>
</body>
</html>
```

Output:

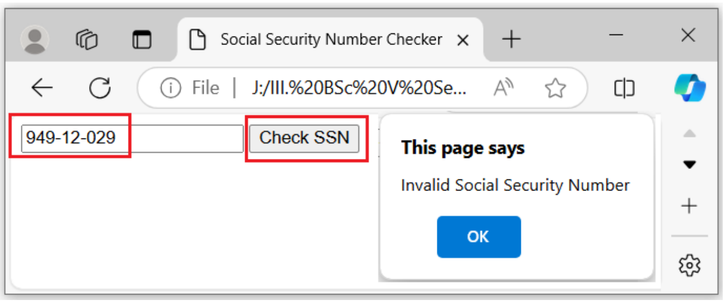


949-12-02987 Check SSN

This page says

Valid Social Security Number

OK



949-12-029 Check SSN

This page says

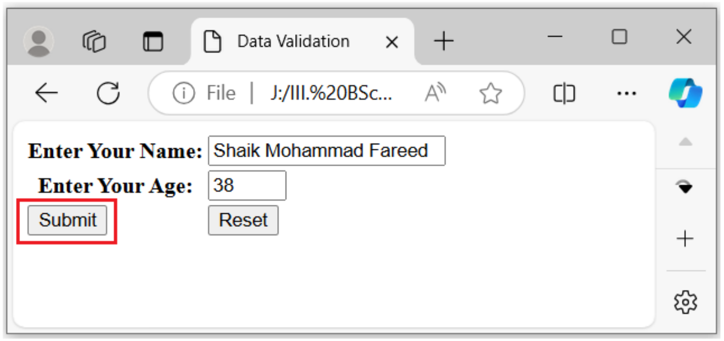
Invalid Social Security Number

OK

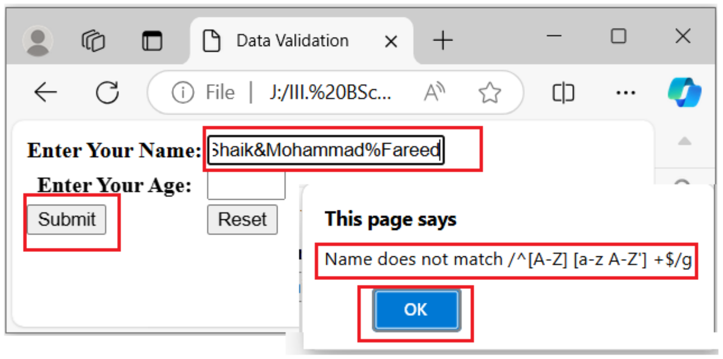
Example for validate the form if any Regular Expressions in JavaScript?

```
<html>
<head>
<title>Data Validation </title>
</head>
<body>
<form method ="post" action="mailto:fareedshaik47@gmail.com" onSubmit="return Validate()">
<table boder ="0">
<tr>
        <th> Enter Your Name: </th>
        <td> <input type ="text" length="24"> </td>
</tr>
<tr>
        <th> Enter Your Age: </th>
        <td> <input type ="text" size="3" maxlength="3"> </td>
</tr>
<tr>
<td> <input type="submit" value="Submit"> </td>
<td> <input type="reset" value="Reset"> </td>
</tr>
</table>
</form>

<script>
function Validate()
{
        var valid = false;
        var name=document.forms[0].elements[0].value;
        var age=document.forms[0].elements[1].value;
        name_re =new RegExp("^[A-Z] [a-z A-Z'] +$", "g");
        age_re =new RegExp("^[\\d]+$",  "g");

if(name.match(name_re))
{
        if(age.match(age_re))  //Only validate the age if the name is OK.
        {
                valid=true;  // Name and age both valid.
        }
        else
                {
                        alert("Age does not match " + agr_re);
                }
}
else
        {
                alert("Name does not match " + name_re);
        }
        return valid;
}
</script>
</body>
</html>
```

Output:



Enter Your Name: Shaik Mohammad Fareed
Enter Your Age: 38



Enter Your Name: Shaik&Mohammad%Fareed
Enter Your Age:

This page says
Name does not match /^[A-Z] [a-z A-Z'] +$/g

(Q) Explain the Exception Handling in JavaScript with examples?

Exception Handling: An Exception is a condition that is caused by a run-time error in a program. Whenever the JavaScript compiler encounters a run-time error, it creates an exception object of the relevant class and throws it out of the method in which it is encountered.

If method throws an exception object, it must be handled properly by catching that exception object. Even through the program raises exception; if we want to continue the execution of the program then we should try to catch the exception object thrown by the error condition. This process is called as "Exception Handling". The Exception Handling code basically consists of two segments.

- To detect errors and throw exception.
- Catch the exception and take appropriate action.

This mechanism of handling exceptions includes the following tasks:

1. Find the problem (***Hit*** or ***try*** the Exception).
2. Inform that an error has occurred (***Throw*** the Exception).
3. Receive the error information (***Catch*** the Exception).
4. Take appropriate action (***Handle*** the Exception).

Process of Exception Handling: In JavaScript Exception Handling mechanism is implemented by using those keywords.

1. try.
2. throw.
3. catch.
4. finally.

Try, Throw, and...Catch...Finally:

- The try statement defines a code block to run (to try).
- The throw statement defines a custom error.
- The catch statement defines a code block to handle any error.
- The finally statement defines a code block to run regardless of the result.

(I). JavaScript try and catch:

- The try statement allows you to define a block of code to be tested for errors while it is being executed.
- The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

Syntax: The JavaScript statements try and catch come in pairs:

```
try
{
        Block of code to try
}
catch(err)
        {
                Block of code to handle errors
        }
```

(II). JavaScript throws Statement: When an error occurs, JavaScript will normally stop and generate an error message.

- The throw statement allows you to create a custom error.
- Technically you can **throw an exception (throw an error)**.
- JavaScript will actually create an **Error object** with two properties: **name** and **message**.
- The exception can be a JavaScript String, a Number, a Boolean or an Object:

Example:
```
throw "Too big value";        // throw a text
throw 500;                    // throw a number
```

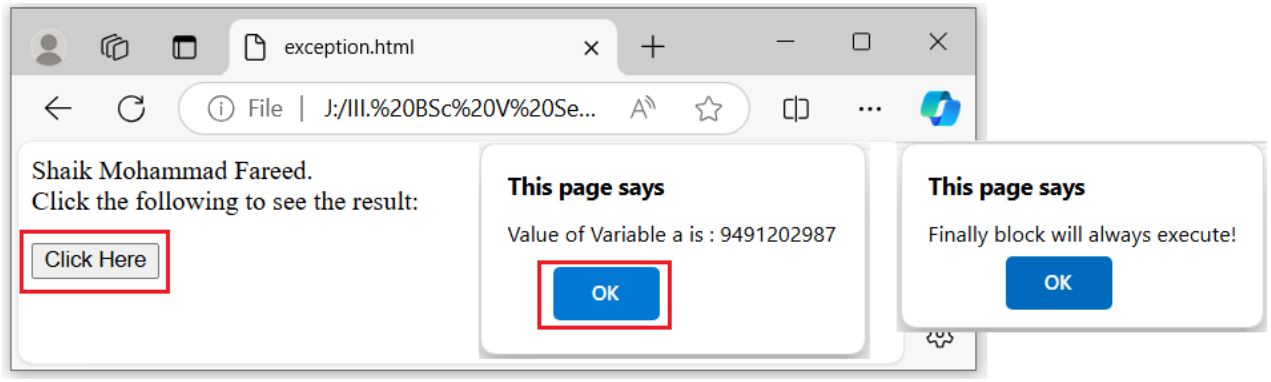
(III). The finally Statement: The finally statement lets you execute code, after try and catch, regardless of the result:

Syntax:
```
try
{
        Block of code to try
}
 catch(err)
{
        Block of code to handle errors
}
finally
{
        Block of code to be executed regardless of the try / catch result
}
```

Example of Try...Catch...Finally Exception in JavaScript?

```
<html>
<head>
<script>
function myFunc()
{
   var mynumber = 9491202987;

   try
        {
         alert("Value of Variable a is : " + mynumber );
        }
catch (shaik)
        {
        alert("Error: " + shaik.description );
        }
finally
 {
      alert("Finally block will always execute!" );
   }
}
</script>
</head>
<body>
        <p>Shaik Mohammad Fareed. <br> Click the following to see the result:</p>
<form>
        <input type="button" value="Click Here" onclick="myFunc();" />
</form>
</body>
</html>
```

Output:



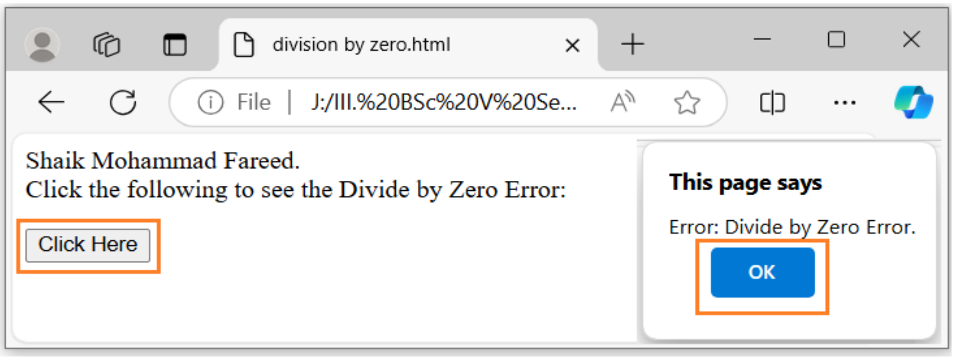
Shaik Mohammad Fareed.
Click the following to see the result:

This page says
Value of Variable a is : 9491202987

This page says
Finally block will always execute!

Example of Division by Zero Error by using Throw Exception in JavaScript?

```
<html>
<head>
<script>
function myFunc()
{
  var a = 100;
  var b = 0;
try
{
    if (b == 0)
    {
      throw("Divide by zero error.");
    }
else
       {
      var c = a / b;
    }
  }
catch (shaik)
 {
    alert("Error: " + shaik );
  }
}
</script>
</head>
<body>
<p> Shaik Mohammad Fareed.  <br> Click the following to see the Divide by Zero Error:</p>
<form>
<input type="button" value="Click Here" onclick="myFunc();" />
</form>
</body>
</html>
```



Shaik Mohammad Fareed.
Click the following to see the Divide by Zero Error:

This page says

Error: Divide by Zero Error.

Output:

(Q) Explain the Messages and Conformations Boxes in DHTML?

JavaScript provide three built-in windows types that can be used from application code. These are useful when you need information from visitors to your site. For instance you may need then to click a confirmation button before submitting information to your database.

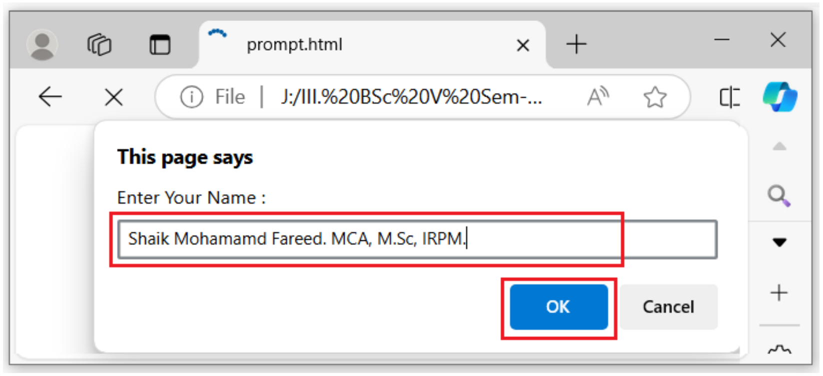
1. Prompt().
2. Confirm().
3. Alert().

1. Prompt(): The prompt commans displays a simple window that contains a prompt and a textfiled in which the user can enter data. The method has two parameters: a text string to be used as the prompt ans a string to use as the default value.

Example:

```
<html>
<body>
<script>
        prompt("Enter Your Name : ", "");
</script>
</body>
</html>
```

Output:



This page says

Enter Your Name :

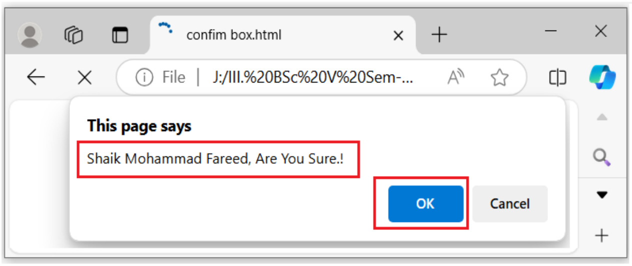
Shaik Mohamamd Fareed. MCA, M.Sc, IRPM.

2. Confirm(): The confirm command is used to display confirmation message, such as submitting form data, or possible as the user tries to follow a link that leaves your site for another. The confirmation window has two buttons 'OK' and 'Cancel'. Selecting Cancel will abort any pending action, while OK will let the action proceed.

Example:

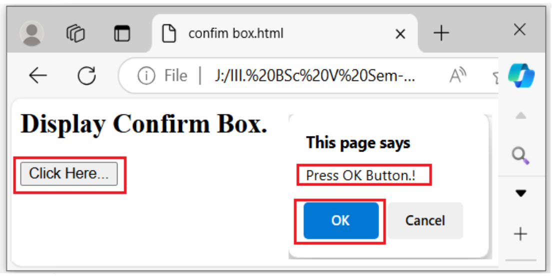
```
<html>
<body>
<h2>Display Confirm Box.</h2>
<button onclick="myFunction()">Click Here...</button>
<script>
confirm("Shaik Mohammad Fareed, Are You Sure.! ");
function myFunction()
{
   confirm("Press OK Button.!");
}
</script>
</body>
</html>
```

Output:



This page says

Shaik Mohammad Fareed, Are You Sure.!



Display Confirm Box.

This page says

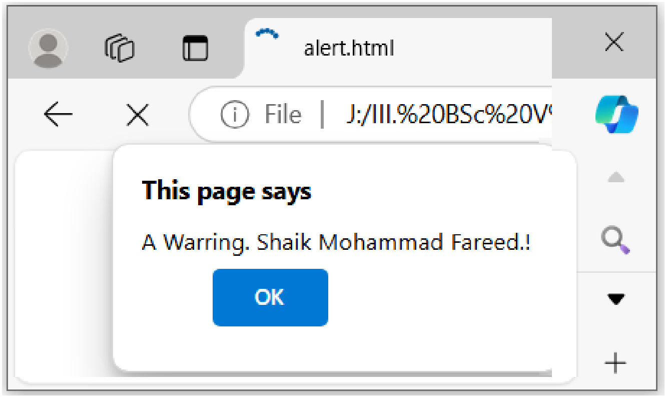
Press OK Button.!

3. Alert(): The alert command is very useful to display the text string and an OK button. This may be use as a warning or to provide a message as visitors.

Example: To display the Alert Box:

```
<html>
<body>
<script type"text/javascript">
alert("A Warring. Shaik Mohammad Fareed.!");
</script>
</body>
</html>
```

Output:



This page says

A Warring. Shaik Mohammad Fareed.!

UNIT – III
Chapter 1: The Building Blocks of PHP

Introduction to PHP:

The PHP stands for "Hypertext Preprocessor" (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor". PHP Syntax is C-Like.
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is zippy in its execution, especially when compiled as an Apache module on the UNIX side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

Common uses of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, and modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

***** Features / Characteristics of PHP:**

- **Performance:** Script written in PHP executes much faster than those scripts written in other language.
- **Open source software:** PHP source code is free available on the web; you can developed all the version of PHP according to your requirement without paying any cost.
- **Platform Independent:** PHP are available for Windows, LINUX & UNIX operating system. A PHP application developed in one operating system can be easily executed in other operating system also.
- **Compatibility & Embedded:** PHP is compatible with almost all local servers used today like Apache, IIS etc. PHP code can be easily embedded within HTML tags and script.

PHP - Environment Setup: In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

- **Web Server**: PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here − https://httpd.apache.org/download.cgi

- **Database**: PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here https://www.mysql.com/downloads/

- **PHP Parser**: In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.

******* (Q) Explain the download, installation and configuring of PHP 8.0.0 for Windows 8/10?**

(I) Download PHP 8.0.0:

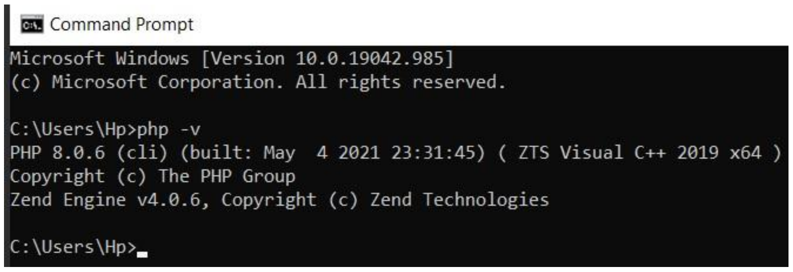
Step 1: Open any web browser, preferable Google Chrome.
Step 2: Here type "Php" in search bar.
Step 3: Here officially website is visible "php.net" click on it.
Step 4: We need go to "Download" section / tab in php.net webpage and click on "Windows Download".
Step 5: Select "Zip [25.36B]" under "VSI6 X64 Thread Safe" menu.
Step 6: After clicking download will begin as we can see left below of status bar.
Step 7: After complete download, then open: Right-Click → Show in folder, it will under download folder and select 'php software' right-click on 'Extract Here' and here it is folder will be visible, just rename it as 'php-8.0.0-15-May-2021'.

(II) Installation PHP 8.0.0:

Step 8: Now it Setup / Install, so select php software folder and right-click than copy, let's goto 'C' drive and past in "Program Files" and give access permission, so click 'Continue'.
Step 9: Now here php software folder, then double-click for open and just copy this path:

Path: "C:\Program Files\php-8.0.0-15-May-2021"

Step 10: Go for System Environment variables, click and type "Environment Variables" on search bar and click on 'Environment Variables' button and choose 'System Variables' and double-click on 'Path' link and click 'New' button and Paste the path which is copied earlier, and click 'Ok' again click 'Ok' and finally click 'Ok' buttons.
Step 11: Now successfully setup / installed for PHP software in our local machine.
Step 12: Now goto start and verify installation: Start → Run → type "cmd". Now opened command window and write a command as "php –v". If the PHP successfully installed then it will be display all details.



```
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Hp>php -v
PHP 8.0.6 (cli) (built: May  4 2021 23:31:45) ( ZTS Visual C++ 2019 x64 )
Copyright (c) The PHP Group
Zend Engine v4.0.6, Copyright (c) Zend Technologies

C:\Users\Hp>
```

(III) Configuring XAMPP Server:

XAMPP:	**X –** Windows Operating System
	A – Apache Tomcat Server
	M – MySQL Database
	P – Php Software
	P – Prams / Programs

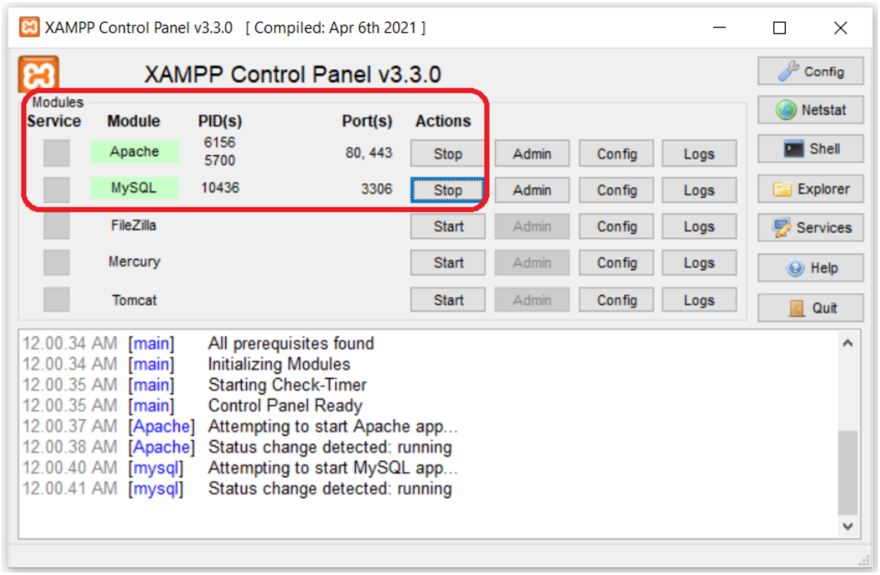
Step 13: Download 'XAMPP Server', then goto Google Chrome, type "XAMPP Server for Windows" in search bar and click on 'download XAMPP for Windows" link and select '7.08/php 7.0.8' and click 'Download' button, it will download in download folder in our system.
Step 14: After download and install XAMPP Server. Go for XAMPP folder which is in 'C' drive, now under this select 'htdocs' (htdocs means: Hypertext Documents). Now create our own folder "FareedPhp".

Install Location: C:\xampp\
Program Location: C:\xampp\htdocs\FareedPhp

Step 15: Now write a PHP program, so we can open any text editor like 'Notepad'. Type a program here.
Step 16: Next 'Save' the program in 'Program Location' must be FileName.php within double quotes.

Save Location: File → Save → C:\xampp\htdocs\FareedPhp\"Welcome.php"

Step 17: Now start XAMPP Server, so just goto search and type 'XAMPP Control Panel'

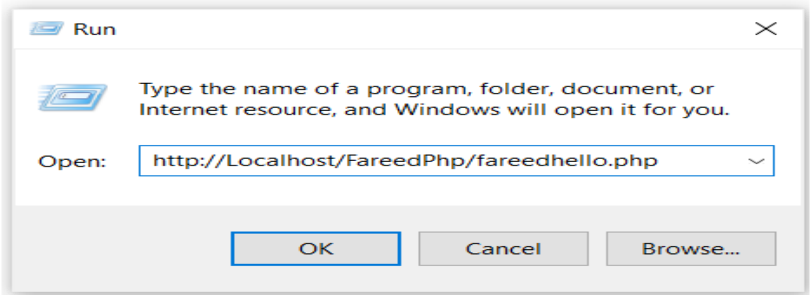


```
12.00.34 AM  [main]     All prerequisites found
12.00.34 AM  [main]     Initializing Modules
12.00.35 AM  [main]     Starting Check-Timer
12.00.35 AM  [main]     Control Panel Ready
12.00.37 AM  [Apache]   Attempting to start Apache app...
12.00.38 AM  [Apache]   Status change detected: running
12.00.40 AM  [mysql]    Attempting to start MySQL app...
12.00.41 AM  [mysql]    Status change detected: running
```

Step 18: Click on "Start" button for Apache Module and Click on "Start" button for MySQL Module.
Step 19: Now check it will install completely or not? Go to web browser type 'localhost' in address bar, it can install correctly it will open XAMPP welcome page. That means everything will be perfectly installed.

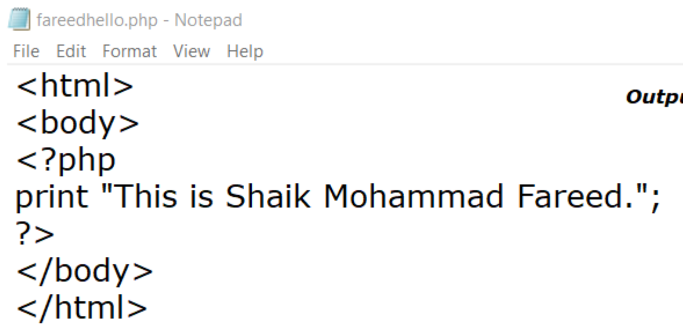
(IV) Run / Executed PHP Program:

Step 20: Open Run prompt and type "http://Localhost/FareedPhp/fareedhello.php" and click 'Ok'.



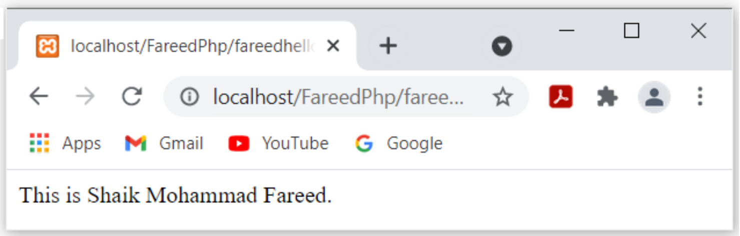
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: http://Localhost/FareedPhp/fareedhello.php



```
<html>
<body>
<?php
print "This is Shaik Mohammad Fareed.";
?>
</body>
</html>
```

Output



```
This is Shaik Mohammad Fareed.
```

Save: C:\xampp\htdocs\FareedPhp\"fareedhello.php"

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension "php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource.
- PHP is easy to learn and runs efficiently on the server side

PHP Syntax: A PHP script is executed on the server, and the plain HTML result is sent back to the browser. PHP script can be placed anywhere in the document.

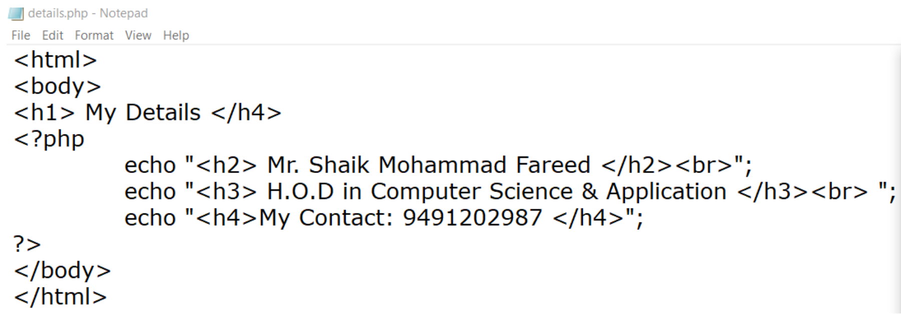
Syntax: A PHP script starts with **<?php** and ends with **?>**

Example:

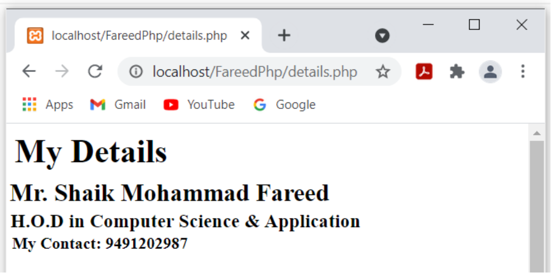
```
        <?php
                // PHP code goes here
        ?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Example:



```
<html>
<body>
<h1> My Details </h4>
<?php
        echo "<h2> Mr. Shaik Mohammad Fareed </h2><br>";
        echo "<h3> H.O.D in Computer Science & Application </h3><br> ";
        echo "<h4>My Contact: 9491202987 </h4>";
?>
</body>
</html>
```


localhost/FareedPhp/details.php
localhost/FareedPhp/details.php
Apps
Gmail
YouTube
Google

```
My Details
Mr. Shaik Mohammad Fareed
H.O.D in Computer Science & Application
My Contact: 9491202987
```

Save: C:\xampp\htdocs\FareedPhp\“details.php”

******* (Q) What is Variables? How to definition and assignment the variables in Php?**

Variables: Variables are fundamental to all programming languages. They are data items that represent a memory storage location in the computer. We can store any value in a variable of its respective datatype in PHP. Variables are containing that hold data such as number and strings.

Rules for PHP variables:

- In PHP a variable is declared using dollar sign ($) followed be variable name.
- Syntax of declaring a variable in PHP: $variable_name = value;
- A variable name must start with a letter or the underscore (_) character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters, underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)
- Variables are assigned with the ‘=’ operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.

Example: $n = 5; $x= 44.5;

Valid Names:

Valid Variable Names	Invalid Variable Names
$name1	$10names
$price_tag	Price.tag
$_abc	$name#last
$Abc_123	A-123
$A23	$5

Declaring and Initialization Variables: Variables are normally declared before they are used. PHP variables can be declared in a script, come from an HTML form. Variables names are explicitly preceded by '$'. You can assign a value to the variable (or initialize a variable) when you declare it, but is not mandatory.

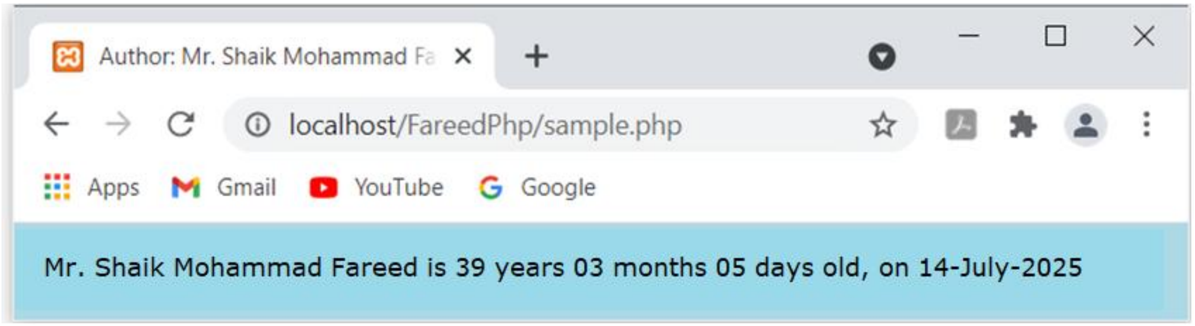
Example:

- $variable_name = Value; // Initialized.
- $variable_name; // Un-initialized, value is null.
- $first_name = "Shaik Mohammad Fareed"; // To declare a variable and value.

Example: Write a Php program to display the basic details of using variables?

```
<html>
<title> Author: Mr. Shaik Mohammad Fareed </title>
<body bgcolor = "lightblue">
<font face ="bookman old style" size = "+2">
        <?php
                $name = "Mr. Shaik Mohammad Fareed";
                $age = "39 years 03 months 05 days";
                $now = date("d/m/y");
                        echo "$name is $age years old, on $now<br />";
        ?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"sample.php"



```
Mr. Shaik Mohammad Fareed is 39 years 03 months 05 days old, on 14-July-2025
```

PHP Scope of Variables: In PHP, variables can be declared anywhere in the script. The scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be four types:

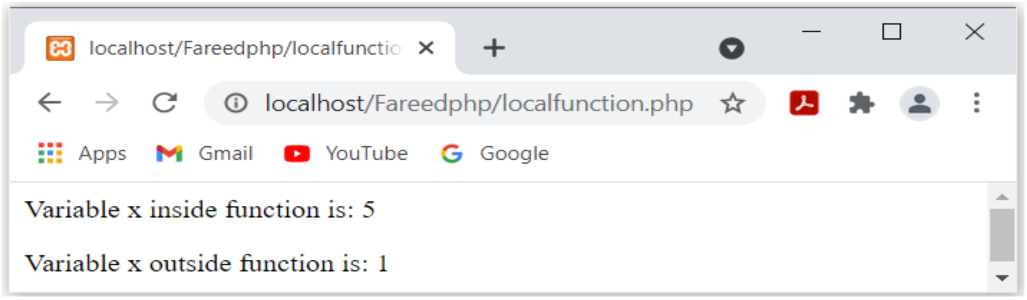
- Local variables
- Functional parameters
- Global variables
- Static variables

(1) Local variables: A variable declared within a function has a local variable and can only be accessed within that function:

Example:

```
<html>
<body>
        <?php
        $x = 1;
                function myTest()
                {
                        $x = 5; // local scope
                        echo "<p>Variable x inside function is: $x</p>";
                }
                myTest();
                echo "<p>Variable x outside function is: $x</p>";
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"localfunction.php"


localhost/Fareedphp/localfunctio
localhost/Fareedphp/localfunction.php
Apps
Gmail
YouTube
Google

```
Variable x inside function is: 5

Variable x outside function is: 1
```

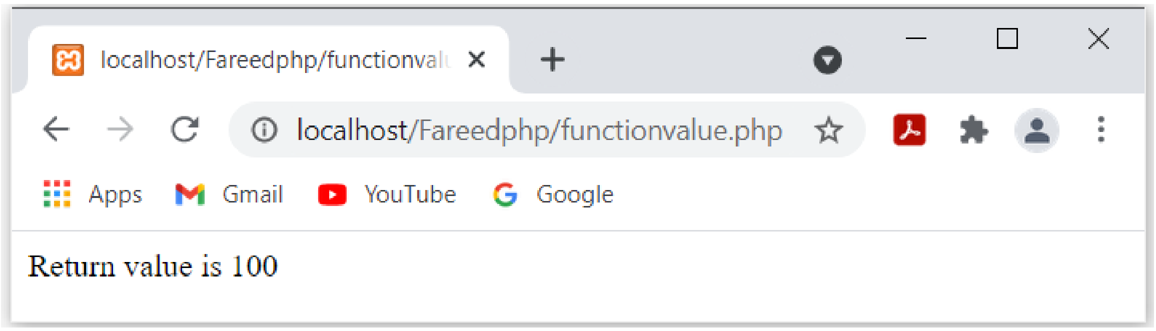
(2) Functional parameters: Function parameters are declared after the function name and inside parentheses.

Example:

```
        <html>
        <body>
        <?php
                 function multiply ($value)  // multiply a value by 10 and return it to the caller
                {
                $value = $value * 10;
                return $value;
                }
        $retval = multiply (10);
        print "Return value is $retval\n";
        ?>
        </body> </html>
```

Save: C:\xampp\htdocs\FareedPhp\"functionvalue.php"

Output:


localhost/Fareedphp/functionvalu
localhost/Fareedphp/functionvalue.php
Apps
Gmail
YouTube
Google

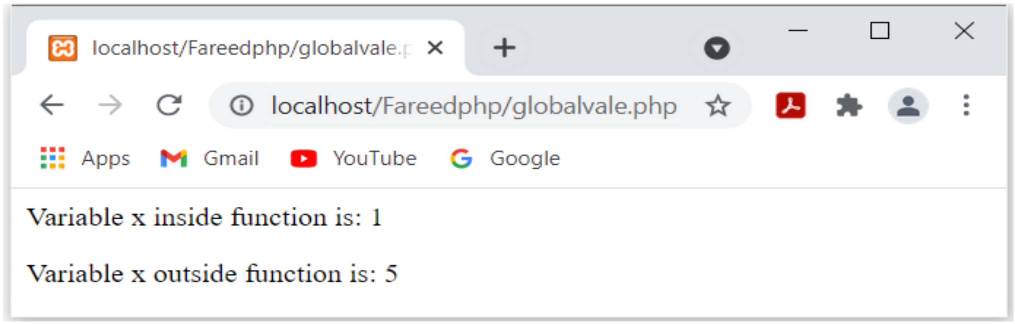
```
Return value is 100
```

(3) Global variable: A variable declared outside a function has a global variable and can only be accessed outside a function:

Example:

```
<html>
<body>
        <?php
                $x = 5;                     // global value
                function myTest()
                {
                        $x = 1;             // local value
                        echo "<p>Variable x inside function is: $x</p>";
                }
                myTest();
                echo "<p>Variable x outside function is: $x</p>";
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"globalvale.php"


localhost/Fareedphp/globalvale.p
localhost/Fareedphp/globalvale.php
Apps
Gmail
YouTube
Google

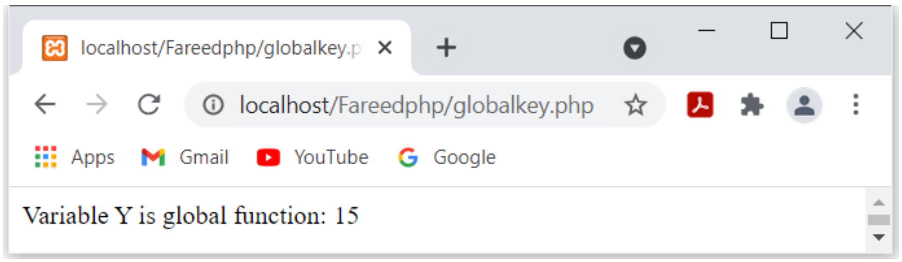
```
Variable x inside function is: 1

Variable x outside function is: 5
```

PHP The global Keyword: The global keyword is used to access a global variable from within a function. To do this, use the global keyword before the variables (inside the function):

Example:

```
<html>
<body>
        <?php
                $x = 5;
                $y = 10;
                function myTest()
                {
                        global $x, $y;
                        $y = $x + $y;
                }
                myTest();                                       // run function
                echo "<p>Variable Y is global function: $y</p>";   // output the new value for variable $y
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"globalkey.php"


localhost/Fareedphp/globalkey.p
localhost/Fareedphp/globalkey.php
Apps
Gmail
YouTube
Google

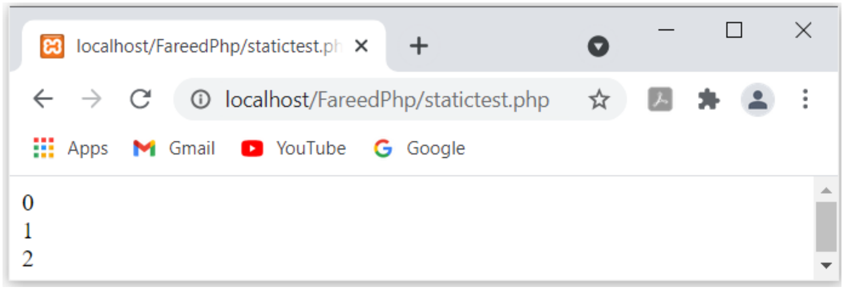
```
Variable Y is global function: 15
```

(4) Static variables: The variables declare as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again. You can declare a variable to be static simply by placing the keyword 'static' in front of the variable name.

Example:

```
<html>
<body>
        <?php
                function myTest()
                {
                        static $x = 0;
                        echo $x;
                        $x++;
                }
                myTest();
                        echo "<br>";
                 myTest();
                        echo "<br>";
                myTest();
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"statictest.php"


localhost/FareedPhp/statictest.ph
localhost/FareedPhp/statictest.php
Apps
Gmail
YouTube
Google

```
0
1
2
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called. The variable is still local to the function.

(Q) Explain difference between PHP echo and PHP print statement?

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. Echo can take multiple parameters (although such usage is rare) while print can take one argument. Echo is marginally faster than print.

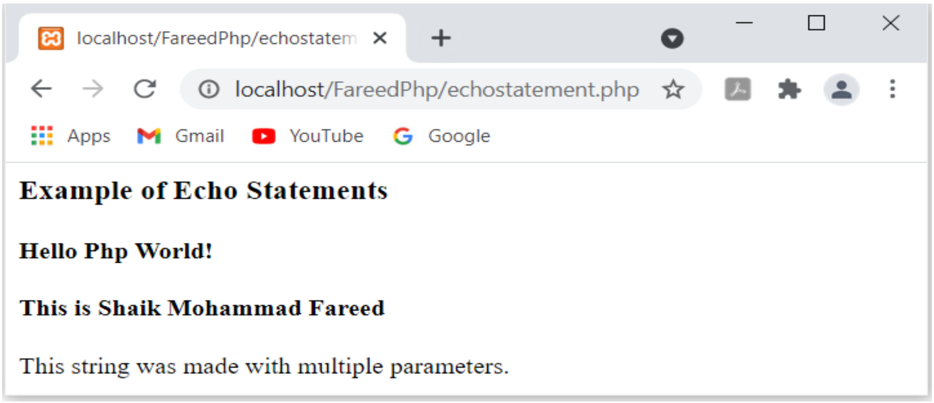
PHP echo Statement: The echo statement can be used with or without parentheses: echo or echo().

Example:

```
<html>
<body>
        <?php
                echo "<h3>Example of Echo Statements</h3>";
                echo "<h4> Hello Php World! </h4>";
                echo "<h4> This is Shaik Mohammad Fareed </h4>";
                echo " This ", "string ", "was ", "made ", "with multiple parameters.";
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"echostatement.php"

Output:



localhost/FareedPhp/echostatem
localhost/FareedPhp/echostatement.php
Apps
Gmail
YouTube
Google

```
Example of Echo Statements

Hello Php World!

This is Shaik Mohammad Fareed

This string was made with multiple parameters.
```

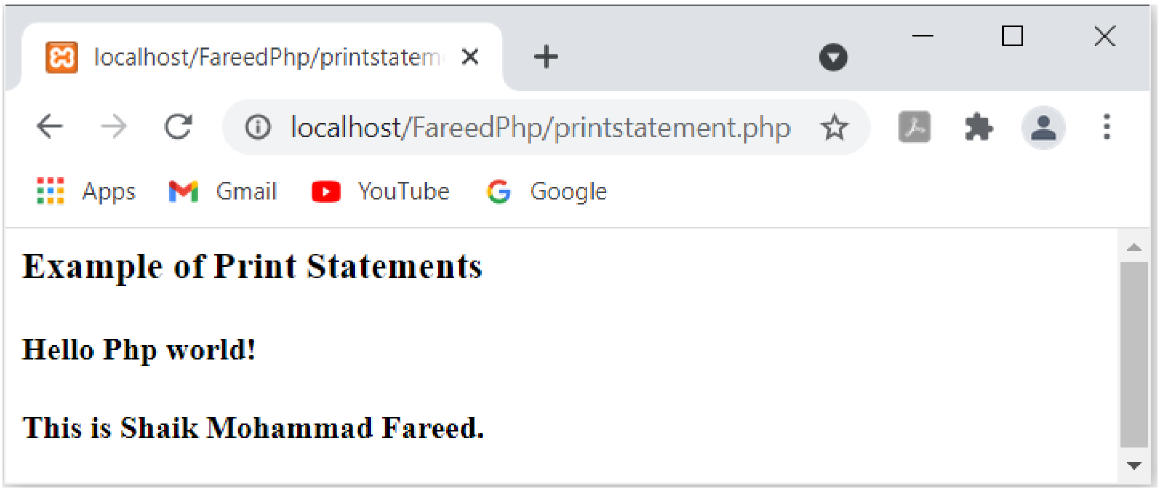
PHP print Statement: The print statement can be used with or without parentheses: print or print()

Example:

```
<html>
<body>
        <?php
                print "<h3>Example of Print Statements</h3>";
                print "<h4> Hello Php world! </h4>";
                print ("<h4> This is Shaik Mohammad Fareed.</h4>");
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"printstatement.php"

Output:



localhost/FareedPhp/printstatem
localhost/FareedPhp/printstatement.php
Apps
Gmail
YouTube
Google

```
Example of Print Statements

Hello Php world!

This is Shaik Mohammad Fareed.
```

VIMP***** (Q) Define PHP Datatypes with examples?**

Datatype: Php datatypes are used to hold different types of data or values. Php supports eight primitive datatypes that can be categorized further in three types:

1. Scalar Datatype (pre-defined)	: Integer, Double / Float, String and Boolean.
2. Compound Datatype (user-defined)	: Array and Object.
3. Special Datatype	: Resource and NULL.

PHP has a total of eight data types which we use to construct our variables:

- **Integer** : Integers are whole numbers, without a decimal point, like 9491202987.
- **Double/Float**: Double or floating-point numbers like 94912.02987
- **String** : String is sequences of characters, like 'PHP supports string operations'.
- **Boolean** : Boolean have only two possible values either true or false.
- **Array** : Array is named and indexed collections of other values.
- **Object** : Object are instances of programmer-defined classes, which can package up both other kinds of values and functions that is specific to the class.
- **Resource** : A Resource is a special variable, holding a reference to an external resource such as a database objects or file handler.
- **NULL** : Null is a special type that only has one value: NULL.

(1) Integer Datatype: An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

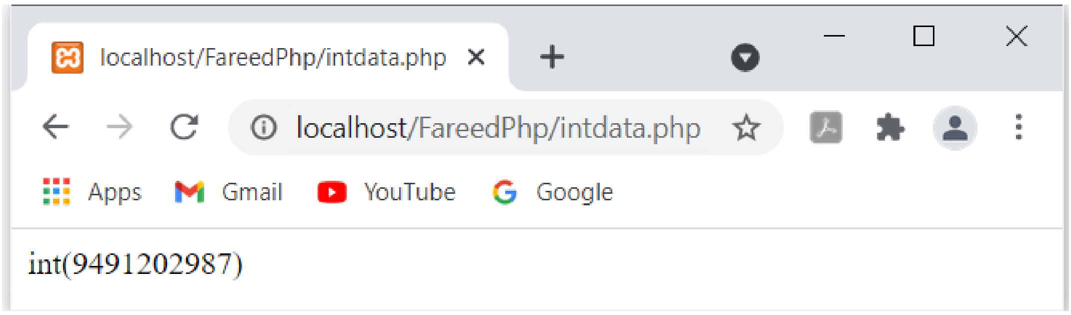
- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

Example:

```
<html>
<body>
        <?php
                $x = 9491202987;
                var_dump($x);
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"intdata.php"

Output:


localhost/FareedPhp/intdata.php
localhost/FareedPhp/intdata.php
Apps
Gmail
YouTube
Google

```
int(9491202987)
```

NOTE: In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

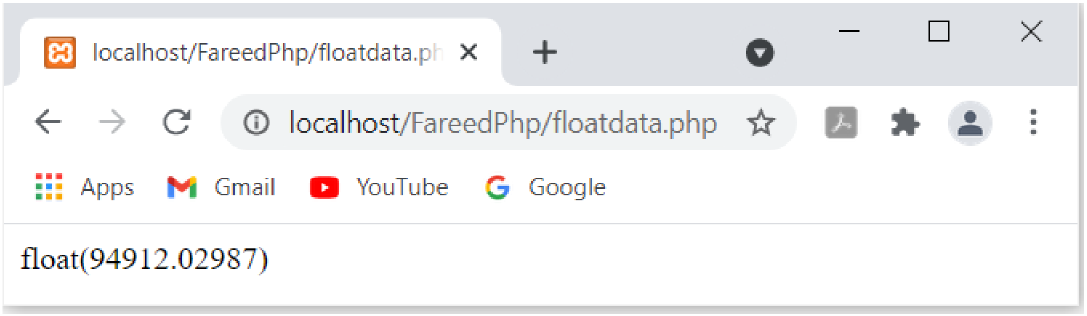
(2) Floating-point number: Floating-point numbers, also called double or real or factional numbers. They must contain a decimal point or an exponent specified. In the following example $x is a float. The PHP var_dump() function returns the data type and value:

Example:

```
<html>
<body>
        <?php
                $x = 94912.02987;
                var_dump($x);
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"floatdata.php"

Output:


localhost/FareedPhp/floatdata.ph
localhost/FareedPhp/floatdata.php
Apps
Gmail
YouTube
Google

```
float(94912.02987)
```

(3) String Datatype:

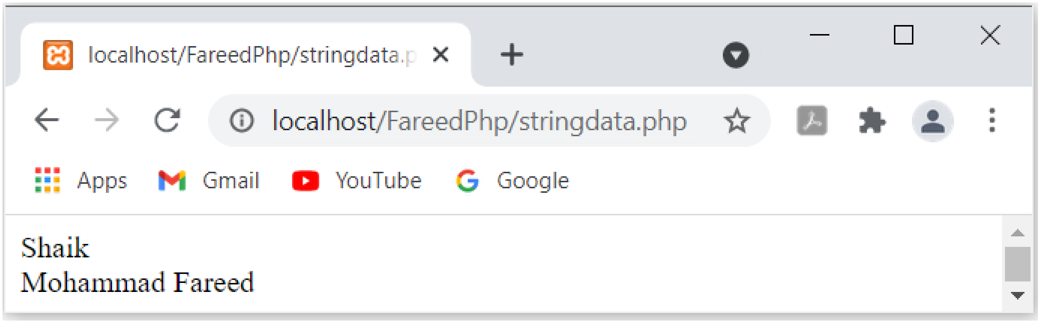
- A string is a sequence of characters.
- A sting can be any text inside quotes; you can use single or double quotes.
- An empty set of quotes is called as null string.
- If a number is enclosed in quotes it is considered a string. Ex "5" is a string whereas 5 is a number.
- Strings can contain 'Escape sequences' (a single character preceded with a backslash).
- Escape sequences icon your browser, the HTML <pre> tag can be used otherwise; the escape sequences placed within your PHP script wills not be interpreted.

Example:

```
<html>
<body>
        <?php
                $x = "Shaik";
                $y = 'Mohammad Fareed';
                echo $x;
                echo "<br>";
                echo $y;
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"stringdata.php"

Output:


localhost/FareedPhp/stringdata.p
localhost/FareedPhp/stringdata.php
Apps
Gmail
YouTube
Google

```
Shaik
Mohammad Fareed
```

Escape Sequences: Escape sequences consist of backslash followed by a single character. When enclosed in double quotes, the backslash causes the interpretation of the next character. Escape sequences ion your browser, the HTML <pre> tag can be used otherwise; the escape sequences placed within your PHP script wills not be interpreted.

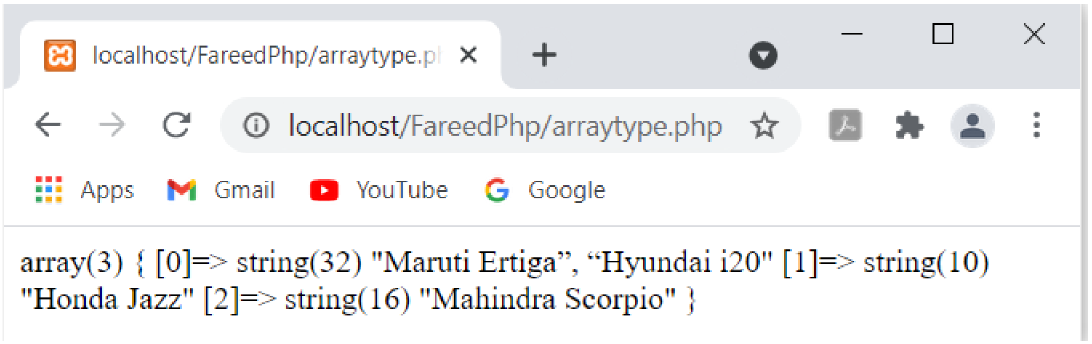
Escape Sequence	What it Represents
\ '	Single quotation marks
\ "	Double quotation marks
\ t	Tab
\ n	Newline
\ r	Return or line feed
\ $	A literal dollar sign
\ \	Backslash
\ O7	Represents the octal value
\ xO5	Represents the hexadecimal character

(4) Boolean Datatype: Boolean literals (introduced in PHP 5) are logical values that have only one of two values. Therefore 'true' or 'false'. Boolean is used in conditional testing. Both case insensitive.

Example: $answer = true / yes / 1; $answer = false / no / 0.
(5) Array Datatype: An array stores multiple data values in one single variable.

Example:
```
<html>
<body>
        <?php
                $cars = array("Maruti Ertiga", "Hyundai i20", "Honda Jazz", "Mahindra Scorpio");
                var_dump($cars);
        ?>
</body>
</html>
```
Save: C:\xampp\htdocs\FareedPhp\"arraytype.php"

Output:


localhost/FareedPhp/arraytype.p
localhost/FareedPhp/arraytype.php
Apps
Gmail
YouTube
Google

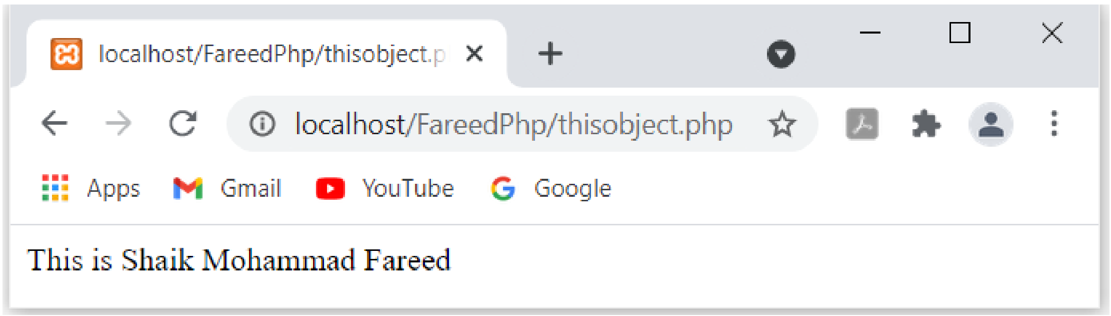
```
array(3) { [0]=> string(32) "Maruti Ertiga", "Hyundai i20" [1]=> string(10)
"Honda Jazz" [2]=> string(16) "Mahindra Scorpio" }
```

(6) Object Datatype: An object is a data type which stores data and information on how to process that data. In PHP, an object must be explicitly declared. First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example:
```
<html>
<body>
        <?php
                class myname
                {
                        private $name = "This is Shaik Mohammad Fareed";
```

```
                    public function display()
                    {
                            echo $this->name;
                    }
            }
            $obj = new myname();
            $obj->display();
    ?>
```
Save: C:\xampp\htdocs\FareedPhp\"thisobject.php"

```
</body>
</html>
```
Output:


localhost/FareedPhp/thisobject.p
localhost/FareedPhp/thisobject.php
Apps
Gmail
YouTube
Google

```
This is Shaik Mohammad Fareed
```

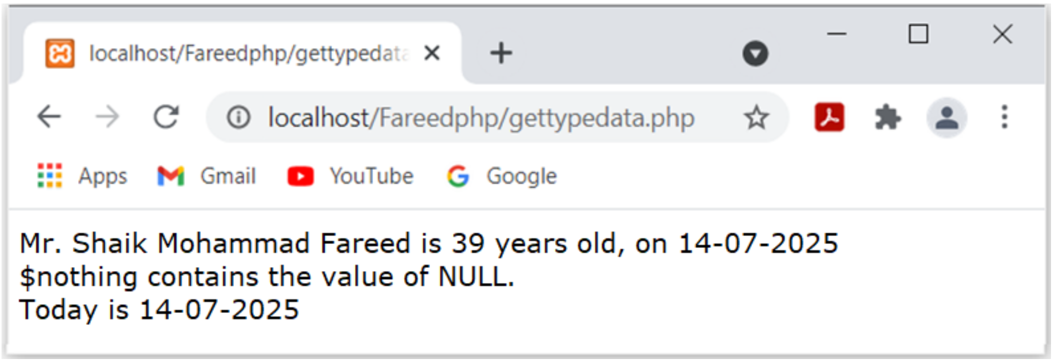
(7) Resource Datatype: A Resource is a special variable, holding a reference to an external resource such as a database objects or file handler. Resources are created and used by special functions.

The gettype() Function: The gettype() built-in function returns a string to identify the datatype of its argument. The argument might be a variable, string, and keyword and so on. You can use the gettype() function to check whether or not a variable has been defined because if there is no value associated with the variable, the gettype() function returns NULL.

Example:

```
<html>
<body>
<font face ="bookman old style" size = "+1">
        <?php
                $name = "Mr. Shaik Mohammad Fareed";
                $age = "39";
                $now = date("d/m/y");
                $nothing;
                        echo "$name is $age years old, on $now<br />";
                        echo '$nothing contains the value of '.gettype($nothing), ".<br />";
                        echo "Today is $now <br />";
        ?>
</font>
</body>
</html>
```
Save: C:\xampp\htdocs\FareedPhp\"gettypedata.php"

Output:


localhost/Fareedphp/gettypedat
localhost/Fareedphp/gettypedata.php
Apps
Gmail
YouTube
Google

```
Mr. Shaik Mohammad Fareed is 39 years old, on 14-07-2025
$nothing contains the value of NULL.
Today is 14-07-2025
```

(8) Null Datatype:

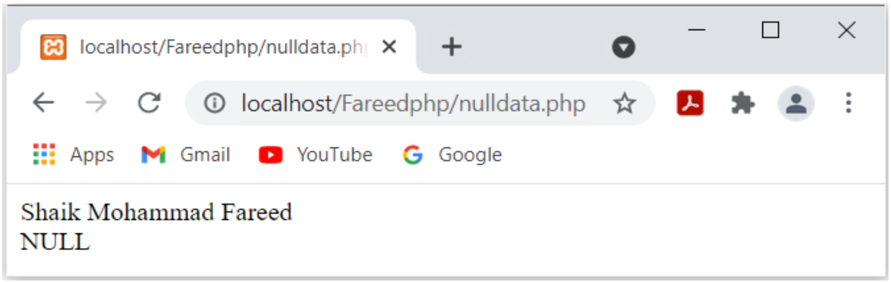
- NULL represents "No Value", meaning "Nothing".
- Null is a special datatype which can have only one value: NULL.
- A variable of datatype NULL is a variable that has no value assigned to it.
- If a variable is created without a value, it is automatically assigned a value of NULL.

Example:

```
<html>
<body>
        <?php
                $x = "Shaik Mohammad Fareed";
                echo $x ,"<br>";
                $x = Null;
                var_dump($x);
        ?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"nulldata.php"

Output:


localhost/Fareedphp/nulldata.php
localhost/Fareedphp/nulldata.php
Apps
Gmail
YouTube
Google

```
Shaik Mohammad Fareed
NULL
```

******** (Q) Define Operators and Expression in PHP?**

Operator: Operator is a symbol that can operate on operations. Operator is a sign that can place between two operands. Operators are used to perform operations on variables and values. PHP divides the operators in the following groups:

- Arithmetic operators
- Comparison operators
- Assignment operators
- Increment/Decrement operators
- Logical (or) Relational operators
- Conditional operator
- String operators

(1) Arithmetic Operators: The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y.
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y.

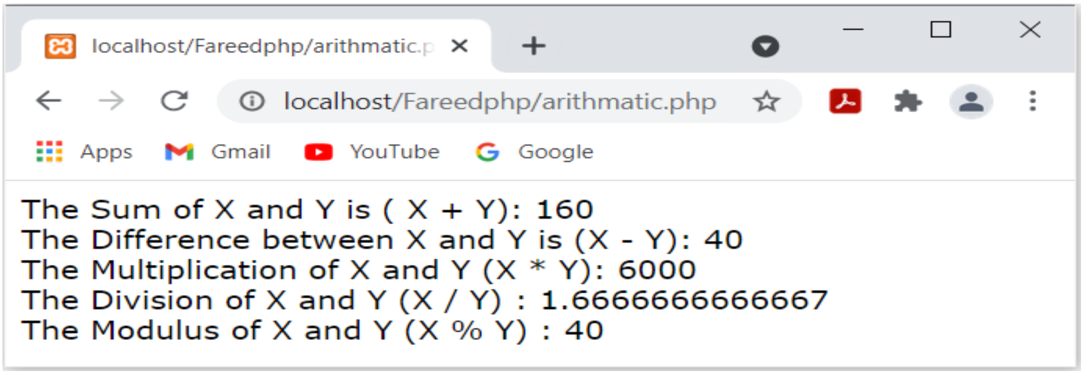
Example:

```
<html>
<body>
<font face ="verdana" font size="3">
<?php
```

```
        $x=100;
        $y=60;
                echo "The Sum of X and Y is ( X + Y): ", ($x + $y), "<br />";
                echo "The Difference between X and Y is (X - Y): ", ($x - $y), "<br />";
                echo "The Multiplication of X and Y (X * Y): ", ($x * $y), "<br />";
                echo "The Division of X and Y (X / Y): ", ($x / $y), "<br />";
                echo "The Modulus of X and Y (X % Y): ",($x % $y), "<br />";
    ?> </font>
    </body>
    </html>
```

Save: C:\xampp\htdocs\FareedPhp\"arithmatic.php"

Output:



localhost/Fareedphp/arithmatic.p
localhost/Fareedphp/arithmatic.php
Apps
Gmail
YouTube
Google

```
The Sum of X and Y is ( X + Y): 160
The Difference between X and Y is (X - Y): 40
The Multiplication of X and Y (X * Y): 6000
The Division of X and Y (X / Y) : 1.6666666666667
The Modulus of X and Y (X % Y) : 40
```

(2) Comparison Operators: In PHP, comparison operators take simple values (numbers or strings) as arguments and evaluate to either TRUE or FALSE.

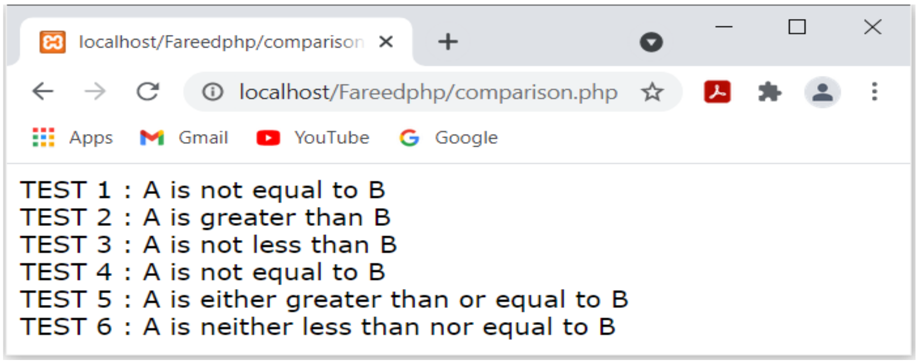
Operator	Name	Example	Result
= =	Equal	$x == $y	TRUE if $x is exactly equal to $y
!=	Not equal	$x != $y	TRUE if $x is exactly not equal to $y.
<>	Not equal	$x <> $y	TRUE if $x is exactly not equal to $y.
!==	Not identical	$x !== $y	TRUE if $x is not equal to $y, or they are not of the same type.
<	Less than	$x < $y	TRUE if $x (left-hand argument) is strictly less than $y (right-hand argument).
>	Greater than	$x > $y	TRUE if $x (left hand argument) is strictly greater than $y (right hand argument).
<=	Less than or equal to	$x <= $y	TRUE if $x (left hand argument) is less than or equal to $y (right hand argument).
>=	Greater than or equal to	$x >= $y	TRUE if $x is greater than or equal to $y.

Example:

```
    <html>
    <body>
    <font face ="verdana" font size="3">
    <?php
        $a = 42;
        $b = 20;
            if( $a == $b )
            {
                echo "TEST 1 : A is equal to B <br/>";
            }
                else
                {
                    echo "TEST 1 : A is not equal to B <br/>";
                }
```

```
                    if( $a > $b )
                    {
                            echo "TEST 2 : A is greater than B <br/>";
                    }
                            else
                            {
                                    echo "TEST 2 : A is not greater than B <br/>";
                            }
                    if( $a < $b )
                    {
                            echo "TEST 3 : A is less than B <br/>";
                    }
                            else
                            {
                                    echo "TEST 3 : A is not less than B <br/>";
                            }

                    if( $a != $b )
                    {
                            echo "TEST 4 : A is not equal to B <br/>";
                    }
                            else
                            {
                                    echo "TEST 4 : A is equal to B <br/>";
                            }
                    if( $a >= $b )
                    {
                            echo "TEST 5 : A is either greater than or equal to B <br/>";
                    }
                            else
                            {
                                    echo "TEST 5 : A is neither greater than nor equal to B <br/>";
                            }
                    if( $a <= $b )
                    {
                            echo "TEST 6 : A is either less than or equal to B <br/>";
                    }
                            else
                            {
                                    echo "TEST 6 : A is neither less than nor equal to B <br/>";
                            }
        ?>
        </font>
        </body>
        </html>
```

Output:


localhost/Fareedphp/comparison
localhost/Fareedphp/comparison.php
Apps
Gmail
YouTube
Google

```
TEST 1 : A is not equal to B
TEST 2 : A is greater than B
TEST 3 : A is not less than B
TEST 4 : A is not equal to B
TEST 5 : A is either greater than or equal to B
TEST 6 : A is neither less than nor equal to B
```

(3) Assignment Operators: The PHP assignment operators are used with numeric values to write a value to a variable. The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

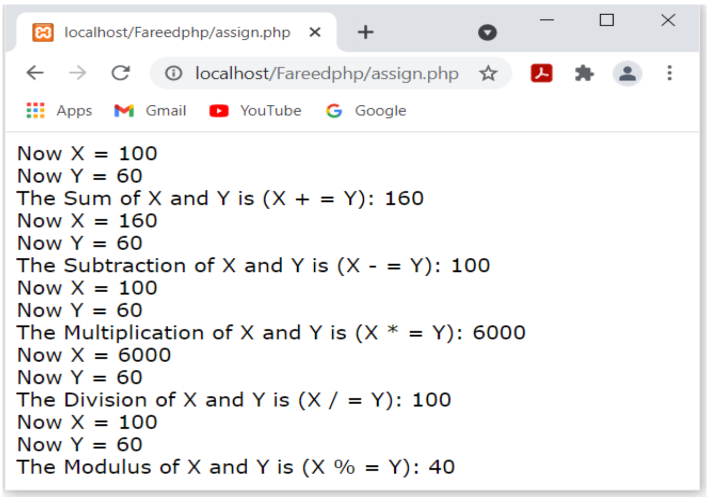
Assignment Operator	Same as Description	
x + = y	x = x + y	Addition
x - = y	x = x - y	Subtraction
x * = y	x = x * y	Multiplication
x / = y	x = x / y	Division
x % = y	x = x % y	Modulus

Example:

```
<html>
<body>
<font face ="verdana" font size="3">
<?php
        $x=100;
        $y=60;
                echo "Now X = ", $x, "<br>";
                echo "Now Y = ", $y, "<br>";
                        echo "The Sum of X and Y is (X + = Y): ", ($x = $x + $y), "<br />";
                echo "Now X = ", $x, "<br>";
                echo "Now Y = ", $y, "<br>";
                        echo "The Subtraction of X and Y is (X - = Y): ", ($x = $x - $y), "<br />";
                echo "Now X = ", $x, "<br>";
                echo "Now Y = ", $y, "<br>";
                        echo "The Multiplication of X and Y is (X * = Y): ", ($x = $x * $y), "<br />";
                echo "Now X = ", $x, "<br>";
                echo "Now Y = ", $y, "<br>";
                        echo "The Division of X and Y is (X / = Y): ", ($x = $x / $y), "<br />";
                echo "Now X = ", $x, "<br>";
                echo "Now Y = ", $y, "<br>";
                        echo "The Modulus of X and Y is (X % = Y): ", ($x = $x % $y), "<br />";
?>
</font>
<body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"assign.php"

Output:



```
Now X = 100
Now Y = 60
The Sum of X and Y is (X + = Y): 160
Now X = 160
Now Y = 60
The Subtraction of X and Y is (X - = Y): 100
Now X = 100
Now Y = 60
The Multiplication of X and Y is (X * = Y): 6000
Now X = 6000
Now Y = 60
The Division of X and Y is (X / = Y): 100
Now X = 100
Now Y = 60
The Modulus of X and Y is (X % = Y): 40
```

(4) Increment/Decrement Operators: When coding in PHP, you will often find it necessary to increment or decrement a variable that is an integer type.

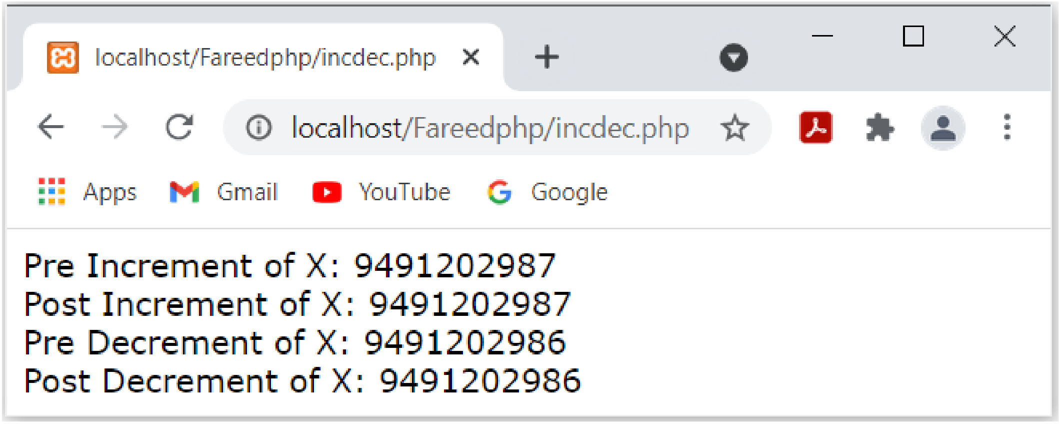
Operator	Name	Description
++$X	Pre-Increment	Increase $X by one , then return $X
$X++	Post-Increment	Return $X, then Increase $X by one
--$X	Pre-Decrement	Decrease $X by one , then return $X
$X--	Post-Decrement	Return $X, then decrease $X by one

Example:

```
<html>
<body>
<font face ="verdana" font size="3">
        <?php
                $x = 9491202986;
                echo "Pre Increment of X: ",++$x, "<br>";
                echo "Post Increment of X: ", $x++, "<br>";
        ?>
        <?php
                $x = 9491202987;
                echo "Pre Decrement of X: ", --$x, "<br>";
                echo "Post Decrement of X: ", $x-- , "<br>";
        ?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"incdec.php"

Output:


localhost/Fareedphp/incdec.php
localhost/Fareedphp/incdec.php
Apps
Gmail
YouTube
Google

```
Pre Increment of X: 9491202987
Post Increment of X: 9491202987
Pre Decrement of X: 9491202986
Post Decrement of X: 9491202986
```

(5) Logical (or) Relational Operator: Logical operators test combinations of Boolean values.

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

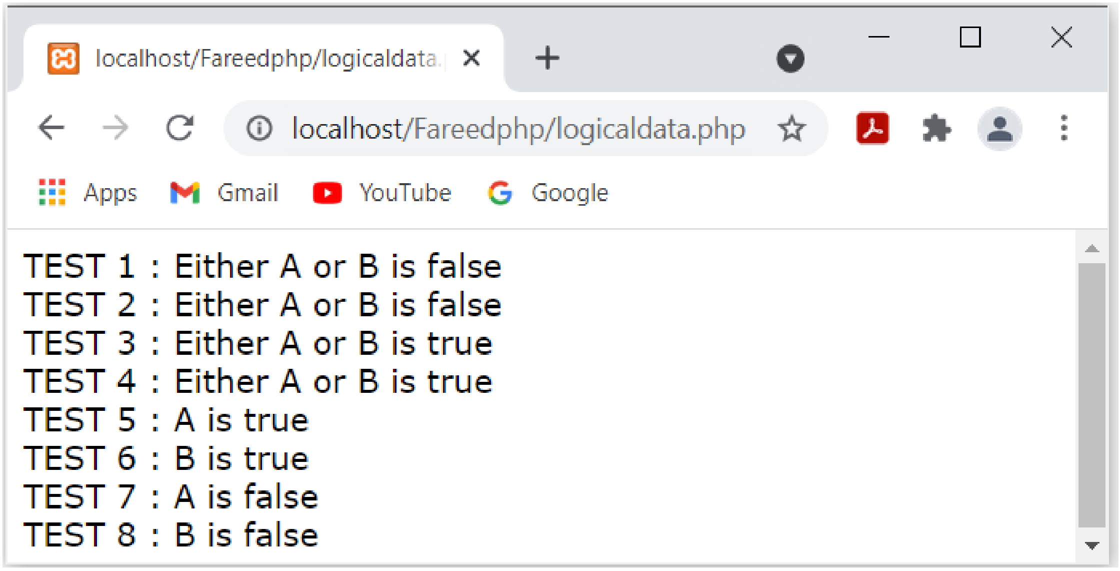
Example:

```
        <html>
        <body>
        <font face ="verdana" font size="3">
                <?php
                        $a = 42;
                        $b = 0;
                        if( $a  &&  $b )
                        {
                                echo "TEST 1 : Both A and B are true <br/>";
                        }
                                else
                                {
                                        echo "TEST 1 : Either A or B is false <br/>";
                                }
                        if( $a and $b )
                        {
                                echo "TEST 2 : Both A and B are true <br/>";
                        }
                                else
                                {
                                        echo "TEST 2 : Either A or B is false <br/>";
                                }
                        if( $a || $b )
                        {
                                echo "TEST 3 : Either A or B is true <br/>";
                        }
                                else
                                {
                                        echo "TEST 3 : Both A and B are false <br/>";
                                }
                        if( $a or $b )
                        {
                                echo "TEST 4 : Either A or B is true <br/>";
                        }
                                else
                                {
                                        echo "TEST 4 : Both A and B are false <br/>";
                                }
                $a = 10;
                $b = 20;
                        if( $a )
                        {
                                echo "TEST 5 : A is true <br/>";
                        }
                                else
                                {
                                        echo "TEST 5 : A  is false <br/>";
                                }
                        if( $b )
                        {
```

```
                        echo "TEST 6 : B is true <br/>";
                }
                        else
                        {
                                echo "TEST 6 : B  is false <br/>";
                        }
                if( !$a )
                {
                        echo "TEST 7 : A is true <br/>";
                }
                        else
                        {
                                echo "TEST 7 : A  is false <br/>";
                        }
                if( !$b )
                {
                        echo "TEST 8 : B is true <br/>";
                }
                        else
                        {
                                echo "TEST 8 : B  is false <br/>";
                        }
        ?>
    </font>
    </body>
    </html>
```

Save: C:\xampp\htdocs\FareedPhp\"logicaldata.php"

Output:


localhost/Fareedphp/logicaldata.
localhost/Fareedphp/logicaldata.php
Apps
Gmail
YouTube
Google

```
TEST 1 : Either A or B is false
TEST 2 : Either A or B is false
TEST 3 : Either A or B is true
TEST 4 : Either A or B is true
TEST 5 : A is true
TEST 6 : B is true
TEST 7 : A is false
TEST 8 : B is false
```

(6) Conditional Operator: There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax.

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

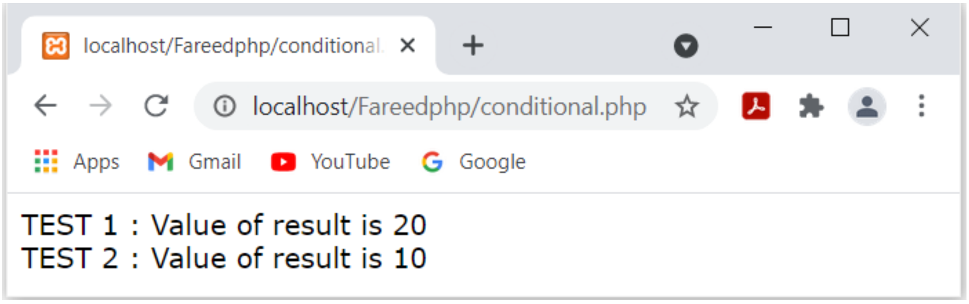
Example:

```
    <html>
    <body>
    <font face ="verdana" font size="3">
    <?php
```

```
            $a = 10;
            $b = 20;
            $result = ($a > $b ) ? $a : $b;       //If condition is true then assign A to result otherwise B
            echo "TEST 1 : Value of result is $result<br/>";
            $result = ($a < $b ) ? $a : $b;       //If condition is true then assign A to result otherwise B
            echo "TEST 2 : Value of result is $result<br/>";
      ?>
      </font>
      </body>
      </html>
```

Save: C:\xampp\htdocs\FareedPhp\"conditional.php"

Output:


localhost/Fareedphp/conditional.
localhost/Fareedphp/conditional.php
Apps
Gmail
YouTube
Google

```
TEST 1 : Value of result is 20
TEST 2 : Value of result is 10
```

(7). String Operator: PHP has two operators that are specially designed for strings.

Operator	Name	Description
.	Concatenation	Concatenation of $txt1 . $txt2
. =	Concatenation Assignment	Append of $txt1 .= $txt2

Precedence of PHP Operators: Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator.

For example x = 7 + 3 * 2; Here x is assigned 13, not 20 because operator * has higher precedence than + so it first get multiplied with 3*2 and then adds into 7.

Here operators with the highest precedence appear at the top of the table; those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associatively
Unary	! ++ --	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=	Right to left

(Q) Define Constants in PHP?

PHP Constants:

- PHP constants are name or identifier that can't be changed during the execution of the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- By default, a constant is case-sensitive.
- By convention, constant identifiers are always uppercase.
- A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.
- PHP constants follow the same PHP variable rules.

Note: Unlike variables, constants are automatically global across the entire script.

Differences between constants and variables are

- There is no need to write a dollar sign ($) before a constant, where as in variable one has to write a dollar sign.
- Constants cannot be defined by simple assignment, they may only be defined using the define() function.
- Constants may be defined and accessed anywhere without regard to variable scoping rules.
- Once the Constants have been set, may not be redefined or undefined.

Type of define constant in PHP: PHP constants can be defined by two ways:

(i). Using 'define()' function.
(ii). Using 'const' keyword.

(i). Using 'define()' function: Let's see the syntax of 'define()' function.

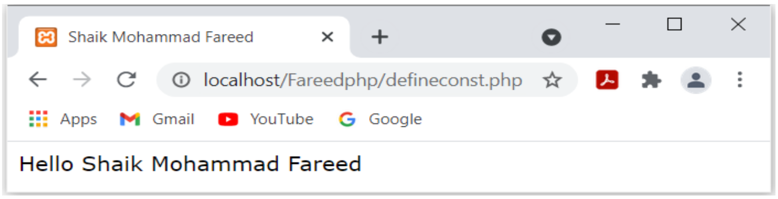
Syntax: define(name, value, case-insensitive);

- **name** : specifies the constant name.
- **value** : specified the constant value.
- **case-insensitive** : default value is false. It means it is case sensitive by default.

Example:

```
<html>
<title> Shaik Mohammad Fareed </title>
<body>
<font face ="verdana" font size="3">
        <?php
                define("MYNAME", "Hello Shaik Mohammad Fareed", false);
                echo MYNAME;
        ?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"defineconst.php"



```
Hello Shaik Mohammad Fareed
```

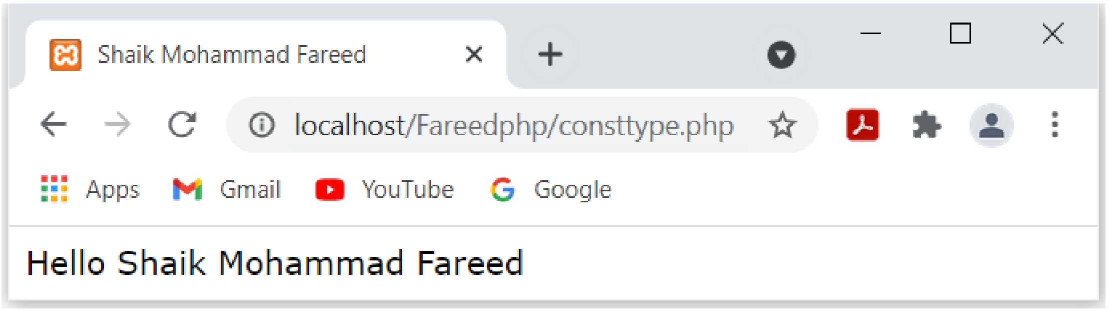
(ii). Using 'const' keyword: The const keyword defines constants at compiler time. It is a language construct not a function. It is bit faster than 'define()'. It is always case sensitive.

Example:

```
<html>
<title> Shaik Mohammad Fareed </title>
<body>
<font face ="verdana" font size="3">
        <?php
                const MYNAME = "Hello Shaik Mohammad Fareed";
                echo MYNAME;
        ?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"consttype.php"

Output:



```
Hello Shaik Mohammad Fareed
```

VVIMP*********(Q) Define the Flow Controls Statements / Functions in PHP?**

Flow Control Functions in PHP: In programming terms this is known as flow control and looping. In the simplest terms this involves some standard scripting structures provided by languages such as PHP to control the logic and overall behavior of a script. These PHP structures can be broken down into a number of categories as follows.

(I). Conditional Statements:

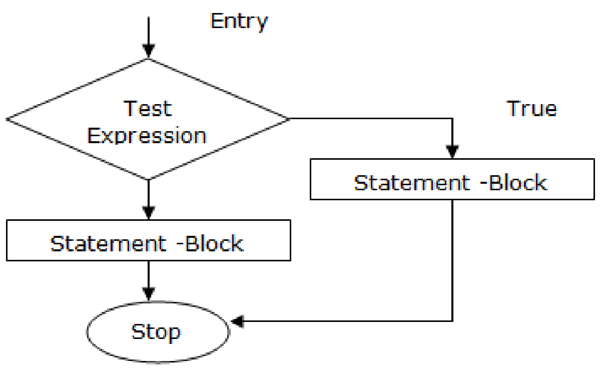
(a). if Statement
(b). if…else Statement
(c). else if…else Statement
(d). switch Statement

(II). Looping Statements:

(e). while Statement
(f). do…while Statement
(g). for Statement
(h). foreach Statement

(I). Conditional Statements:

(a). if Statement: Simple If Statement is used to execute one set of statements based on the results of a given test condition. If condition is "True" then executed only within If-Block statements only. Otherwise skipped and the execution other statements.



Syntax:

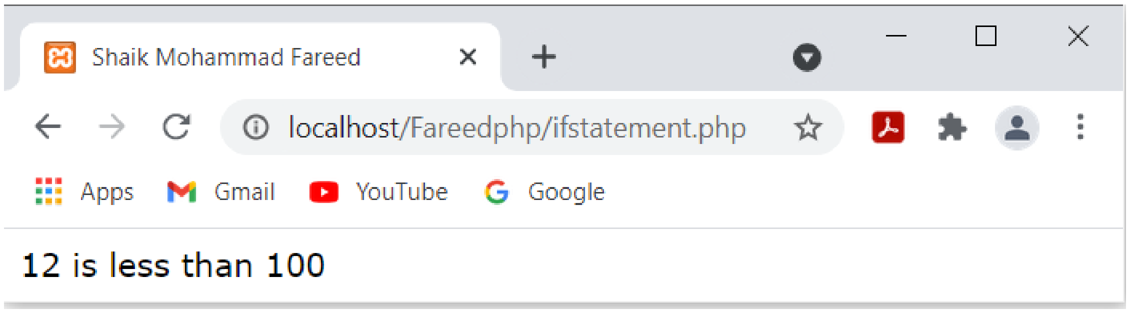
```
                if (condition)
                {
                        Code to be executed if condition is true;
                }
```

Example:

```
        <html>
        <title> Shaik Mohammad Fareed </title>
        <body>
        <font face ="verdana" font size="3">
        <?php
                $num = 12;
                if ($num < 100)
                {
                        echo "$num is less than 100";
                }
        ?>
        </font>
        </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"ifstatement.php"

Output:



```
12 is less than 100
```

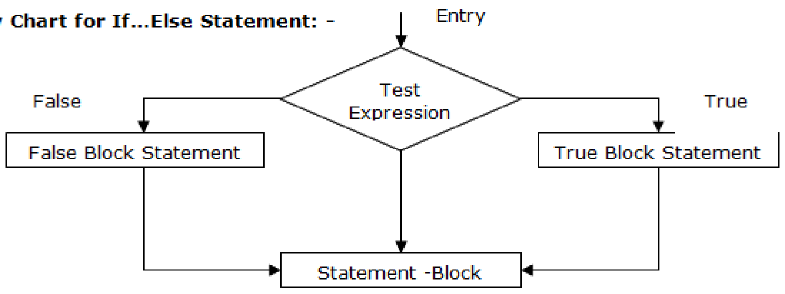
(b). if…else Statement:

- If the Test Expression / Condition are "True", then executed the True Block of Statement(s).That means 'If-Block Statement' is executed only.
- If the Test Expression / Condition are "False", then executed the False Block of Statement(s). That means 'Else-Block Statement' is executed only.
- But they have not executed both blocks.

Syntax:

```
if (condition)
   {
       code to be executed if condition is true;
   }
   else
   {
      code to be executed if condition is false;
  }
```

Flow Chart for If…Else Statement: -



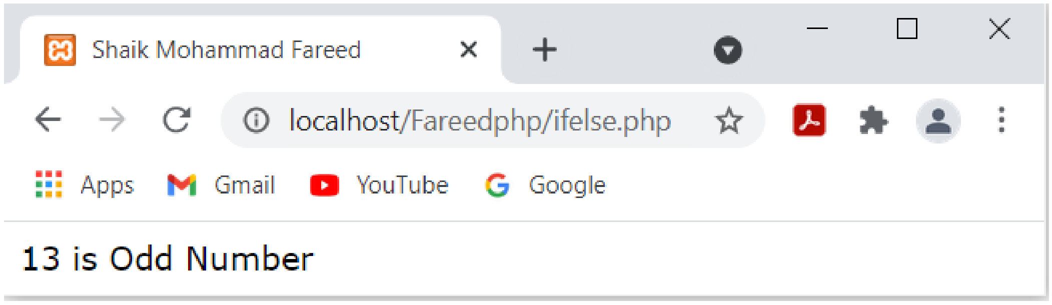
Example:

```
        <html>
        <title> Shaik Mohammad Fareed </title>
        <body>
        <font face ="verdana" font size="3">
        <?php
```

```
                $num = 13;
                if ($num%2 == 0)
                {
                        echo "$num is Even Number";
                }
                        else
                        {
                                echo "$num is Odd Number";
                        }
        ?>
        </font>
        </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"ifelse.php"

Output:

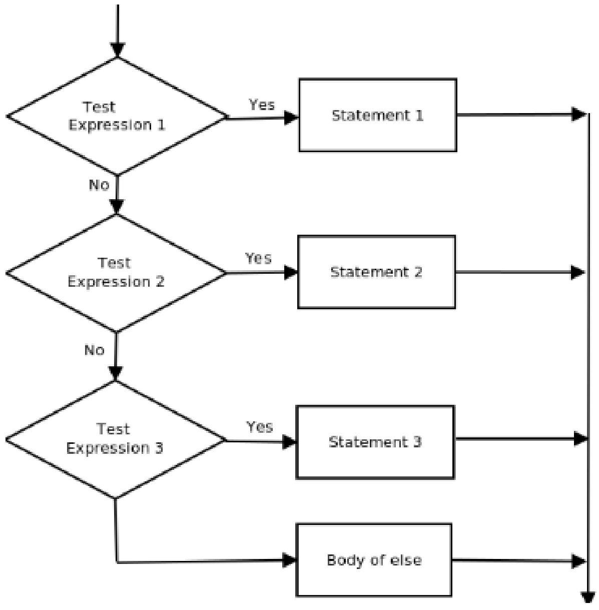


```
13 is Odd Number
```

(c). else if…else Statement: if the given condition is true execute the if-block statements, otherwise the condition is false, check another if condition in else-block. Then all condition is false, executed default else-block statement.

Syntax:

```
if (condition1)
{
    code to be executed if condition is true;
}
        else if (condition 2)
        {
                code to be executed if condition is true;
        }
                else
                {
                code to be executed all else condition is false;
                }
```

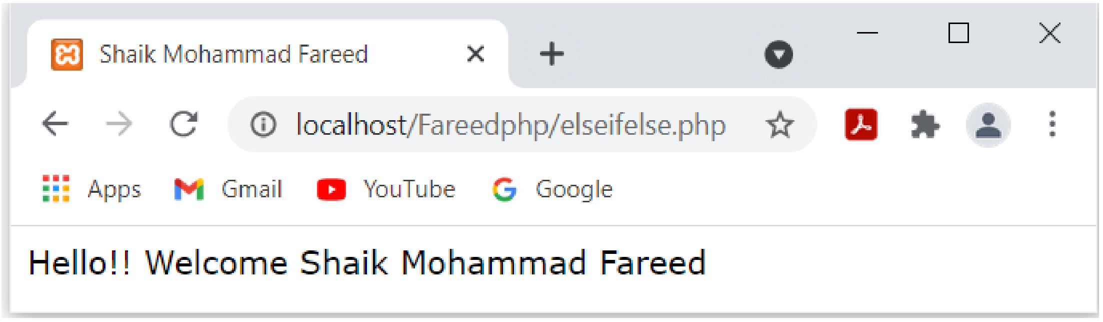


Example:

```
        <html>
        <title> Shaik Mohammad Fareed </title>
        <body>
        <font face ="verdana" font size="3">
                <?php
                        $customername ="Mohammad Fareed";
                        if($customername == "Fareed")
                        {
                                echo "Hello Fareed !";
                        }
                        else if ($customername=="Mohammad Fareed")
                        {
                                echo "Hello!! Welcome Shaik Mohammad Fareed";
                        }
                                else
                                {
                                        echo "Who are u Man ???";
                                }
        ?>
        </font>
        </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"elseifelse.php"

Output:

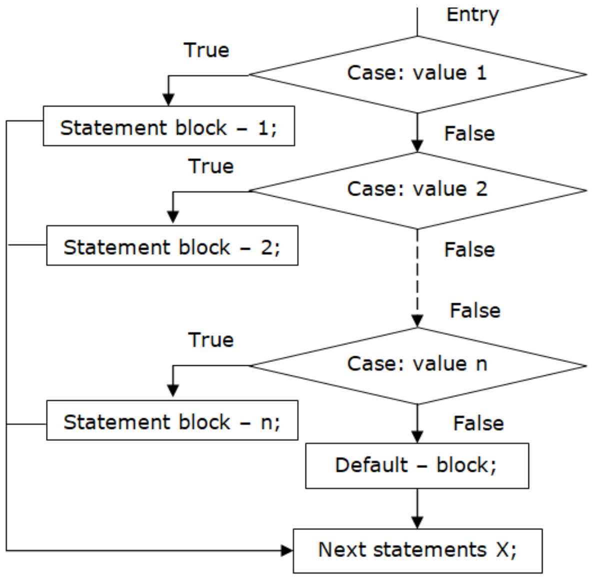


Hello!! Welcome Shaik Mohammad Fareed

(d) switch Statement: PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.

Syntax:

```
switch (expression)
{
        case value1:
                //code to be executed
        break;
        case value2:
                //code to be executed
        break;
        ......
        default:
         code to be executed if all cases are not matched;
}
```

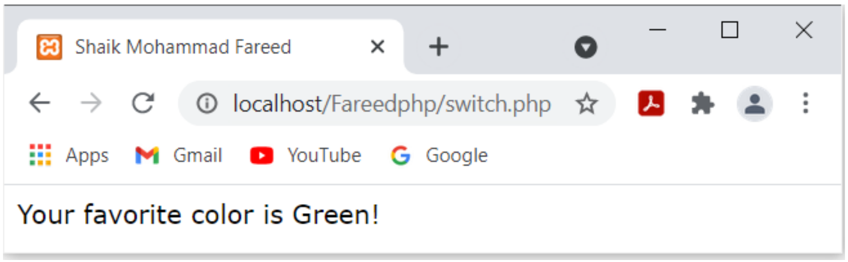


Example:

```
        <html>
        <title> Shaik Mohammad Fareed </title>
        <body>
        <font face ="verdana" font size="3">
        <?php
                $favcolor = "green";
                switch ($favcolor)
                {
                        case "red":
                                echo "Your favorite color is Red!";
                                break;
                        case "green":
                                echo "Your favorite color is Green!";
                                break;
                        case "blue":
                                echo "Your favorite color is Blue!";
                                break;
                        default:
                                echo "Your favorite color is either Red, Blue, or Green!";
                }
        ?>
        </font>
        </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"switch.php"

Output:



```
Your favorite color is Green!
```

(II). Looping Statements: Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

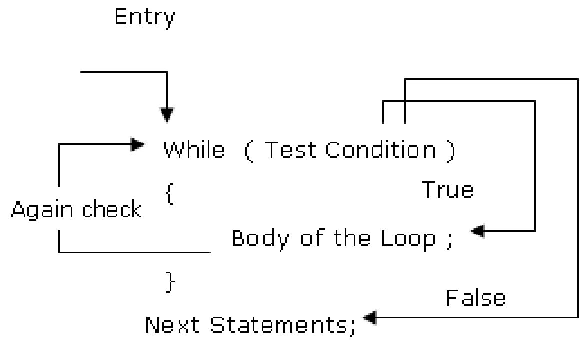
- **while:** while loops through a block of code as long as the specified condition is true
- **do...while**: do...while loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for:** for loops through a block of code a specified number of times
- **foreach:** foreach loops through a block of code for each element in an array

(e). while Statement: The while statement will execute a block of code if and as long as a test expression is true. If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

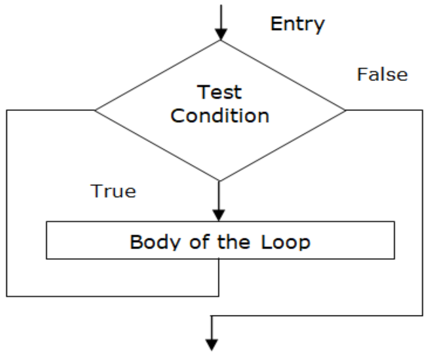
Syntax for While Statement:

```
initialization;
while (Test Condition)
  {
        Body of the Loop;
  }
```

The Control flow in the While Statement is shown below:



Flow Chart for while Statement:

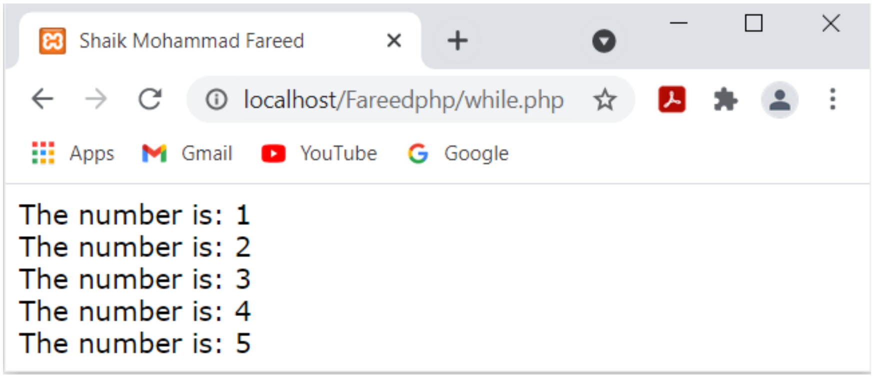


Example:

```
<html>
<title> Shaik Mohammad Fareed </title>
<body>
<font face ="verdana" font size="3">
<?php
        $x = 1;
                while ($x <= 5)
                {
                        echo "The number is: $x <br>";
                        $x++;
                }
?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"while.php"

Output:



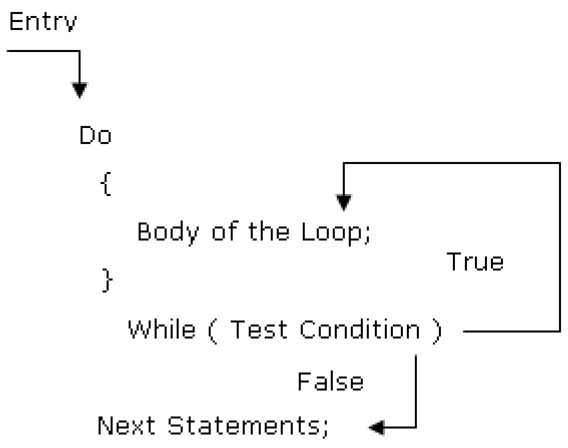
```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

(f). do…while Statement: The do…while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

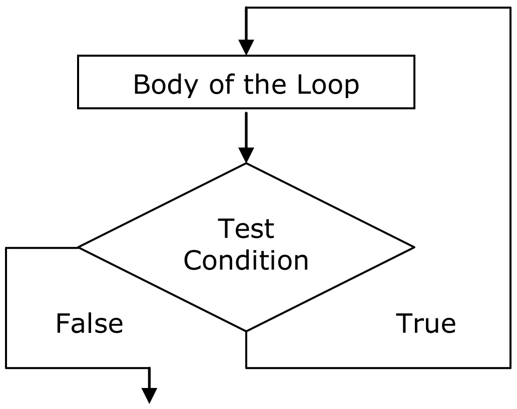
Syntax for Do…While Loop Statement:

```
do
 {
     Body of the Loop;
 }
    while ( Test Condition );
        Next Statements;
```

The Control flow in the Do…While Statement: -



Flow Chart for Do…While Statement:

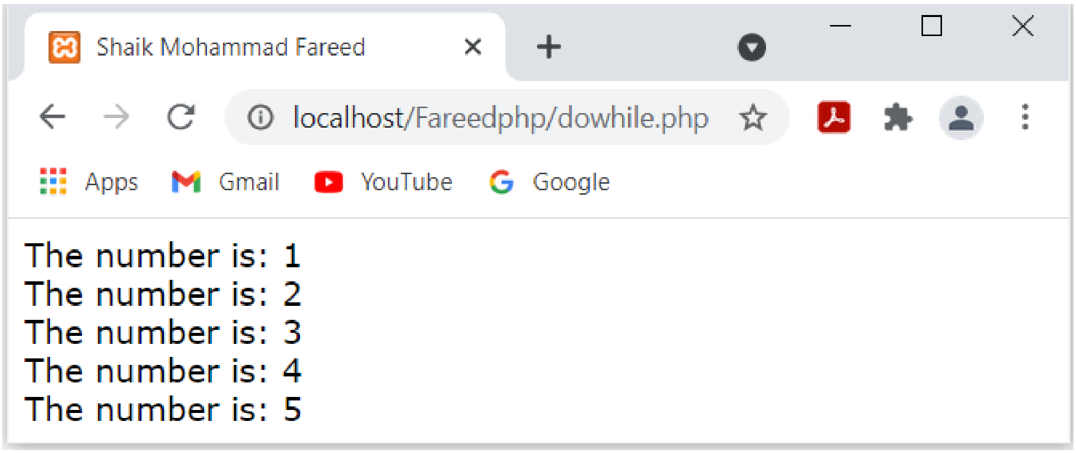


Example:

```
<html>
<title> Shaik Mohammad Fareed </title>
<body>
<font face ="verdana" font size="3">
<?php
        $x = 1;
        do
                {
                        echo "The number is: $x <br>";
                        $x++;
                }
        while ($x <= 5);
?>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\“dowhile.php”

Output:



```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

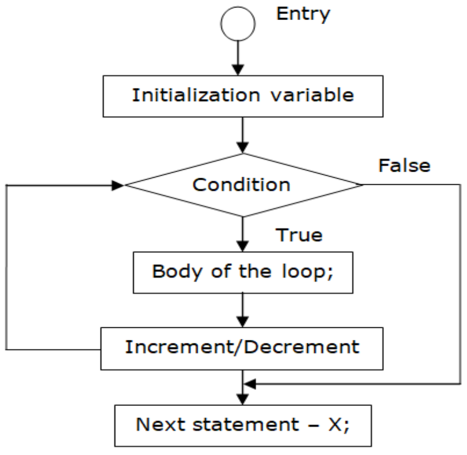
(g). for Statement: The for loop is used when you know in advance how many times the script should run.
Syntax of for loop:

```
for (init counter; test counter / condition; increment counter or decrement)
{
   Code to be executed;
}
```

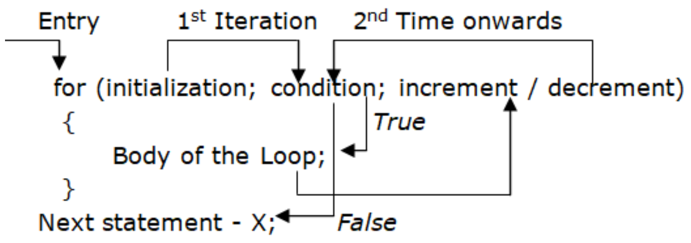
Parameters:

- **init counter:** Initialize the loop counter value
- **test counter:** Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- **increment counter:** Increases the loop counter value

Flow Chart of For Statement:



The Control Flow of For Statement:

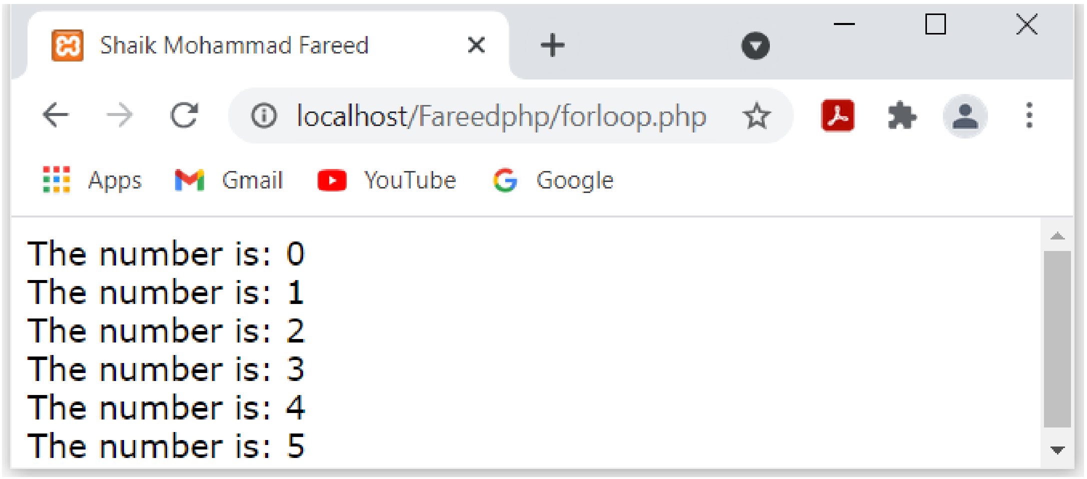


Example:

```
<html>
<title> Shaik Mohammad Fareed </title>
<body>
<font face ="verdana" font size="3">
<?php
        for ($x = 0; $x <= 5; $x++)
        {
                echo "The number is: $x <br>";
        }
?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"forloop.php"

Output:



```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

(h) foreach Statement: The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

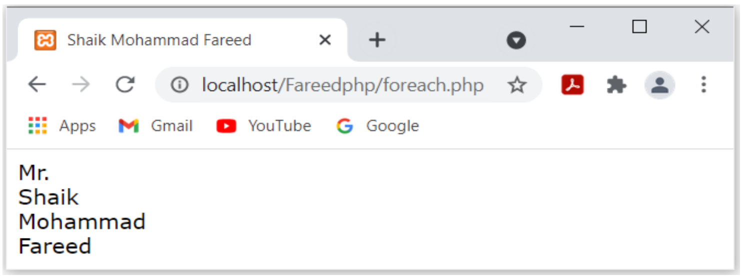
Syntax of foreach:

```
foreach (array as value)
{
        code to be executed;
}
```

Example:

```
<html>
<title> Shaik Mohammad Fareed </title>
<body>
<font face ="verdana" font size="3">
<?php
        $colors = array("Mr.", "Shaik", "Mohammad", "Fareed");
        foreach ($colors as $value)
        {
                echo "$value <br>";
        }
?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"foreach.php"



```
Mr.
Shaik
Mohammad
Fareed
```

(Q) Define Break and Continue Statements in PHP?

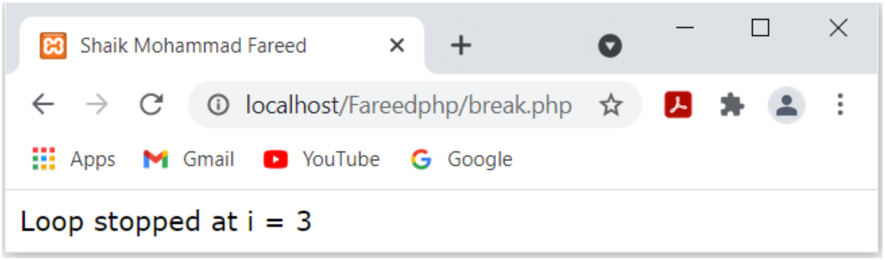
Break Statement: The PHP break keyword is used to terminate the execution of a loop prematurely. The break statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.

Syntax: break;

Example:

```
        <html>
        <title> Shaik Mohammad Fareed </title>
        <body>
<font face ="verdana" font size="3">
        <?php
                $i = 0;
                while( $i < 10)
                {
                        $i++;
                        if( $i == 3 )
                        break;
                }
                echo ("Loop stopped at i = $i");
        ?>
        </font>
        </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\“break.php”



```
Loop stopped at i = 3
```

Output:

Continue Statement: The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop.

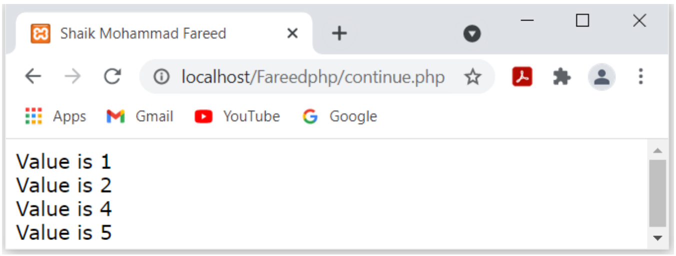
Just like the break statement the continue statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

Syntax: continue;

Example:

```
        <html>
        <title> Shaik Mohammad Fareed </title> <body>
        <font face ="verdana" font size="3">
        <?php
                $array = array( 1, 2, 3, 4, 5);
                        foreach( $array as $value )
                        {
                                if( $value == 3 )
                                continue;
                        echo "Value is $value <br />";
                        }
        ?>   </font> </body> </html>
```

Save: C:\xampp\htdocs\FareedPhp\“continue.php”



```
Value is 1
Value is 2
Value is 4
Value is 5
```

Output:

IMP (Q) What is Array? How to create an array in PHP? Explain about types of Arrays in PHP?

Array in PHP: An array is a data structure that stores one or more similar type of values in a single value.

(or)

An array is a special variable, which can hold more than one value under a single name, and you can access the values by referring to an index number.

Create an Array in PHP: Here we have used "array()" function to create array. By default array index starts from 0 (zero).

Syntax: $variable_name = array (list of parameters);

- An array can hold any number of values.
- Each value in an array is called an element.
- You access each element via its index, which is a numeric or string value. Every element in an array has its own unique index.
- An element can store any type of value, such as an integer, a string, or a Boolean.

Advantage of PHP Array:

- **Less code:** We don't need to define multiple variables.
- **Easy to traverse:** By the help of single loop, we can traverse all the elements of an array.
- **Sorting:** We can sort the elements of array.

PHP Array Types: There are 3 types of array in PHP.

1. **Numeric / Indexed Array:** Arrays with a numeric index.
2. **Associative Array:** Arrays association with named key value.
3. **Multidimensional Array:** Arrays containing one or more array and values are accessed using multiple indices.

(1). Numeric / Indexed Array: PHP indexed array is an array which is represented by an index number by default. All elements of array are represented by an index number which starts from 0.

PHP indexed array can store numbers, strings or any object. PHP indexed array is also known as Numeric Array.

There are two ways to define Indexed Array:

1st Way: $myname = array ("Shaik", "Mohammad", "Fareed");

2nd Way:
```
$myname[0] = "Shaik";
$myname[1] = "Mohammad";
$myname[2] = "Fareed";
```

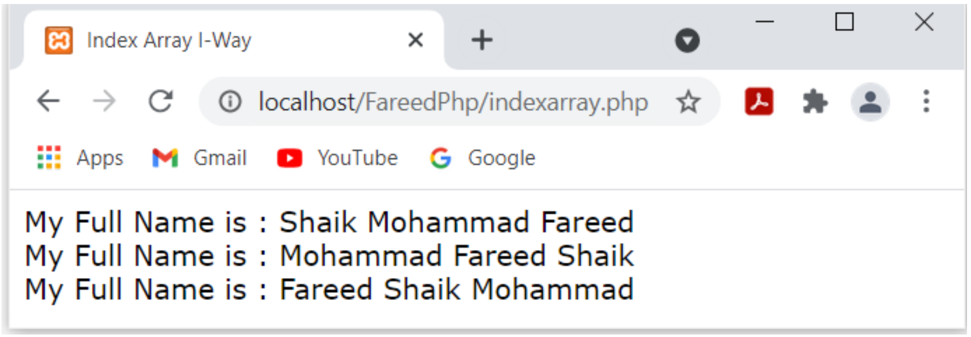
PHP Indexed Array Example

Example: 1

```
<html>
        <head><title>Index Array I-Way</title></head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                $myname = array ("Shaik", "Mohammad", "Fareed");
                echo "My Full Name is : $myname[0] $myname[1] $myname[2]" ,"<br />";
                echo "My Full Name is : $myname[1] $myname[2] $myname[0]", "<br />";
                echo "My Full Name is : $myname[2] $myname[0] $myname[1]";
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"indexarray.php"

Output:



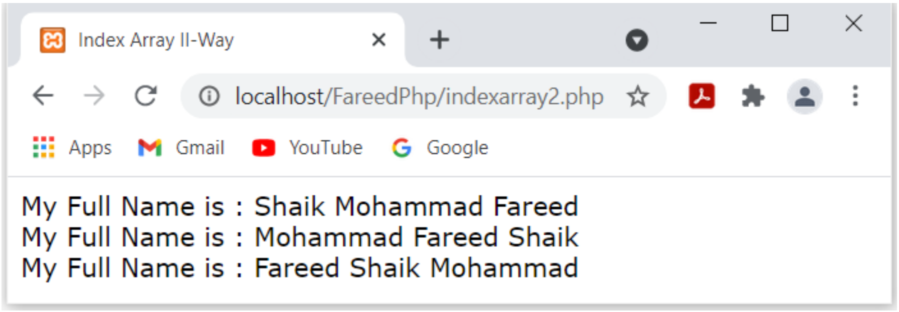
```
My Full Name is : Shaik Mohammad Fareed
My Full Name is : Mohammad Fareed Shaik
My Full Name is : Fareed Shaik Mohammad
```

Example: 2

```
<html>
        <head>
        <title>Index Array II-Way</title></head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                $myname[0] = "Shaik";
                $myname[1] = "Mohammad";
                $myname[2] = "Fareed";
                echo "My Full Name is : $myname[0] $myname[1] $myname[2]", "<br />";
                echo "My Full Name is : $myname[1] $myname[2] $myname[0]", "<br />";
                echo "My Full Name is : $myname[2] $myname[0] $myname[1]";
        ?>
        </font>
        </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"indexarray2.php"

Output:



```
My Full Name is : Shaik Mohammad Fareed
My Full Name is : Mohammad Fareed Shaik
My Full Name is : Fareed Shaik Mohammad
```

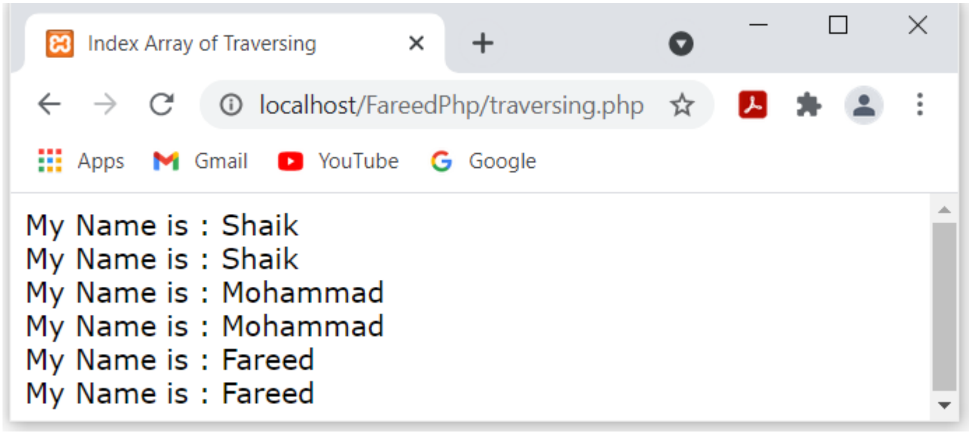
Traversing PHP Indexed Array: We can easily traverse array in PHP using foreach loop. Let's see a simple example to traverse all the elements of PHP array.

Example:

```
        <html>
                <head><title>Index Array of Traversing</title></head>
                <body>
                <font face ="verdana" font size="3">
                <?php
                        $myname = array ("Shaik", "Mohammad", "Fareed");
                        foreach( $myname as $myname )
                        {
                                echo "My Name is : $myname <br />";
                                echo "My Name is : $myname <br />";
                        }
                ?>
        </font>
        </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"traversing.php"

Output:



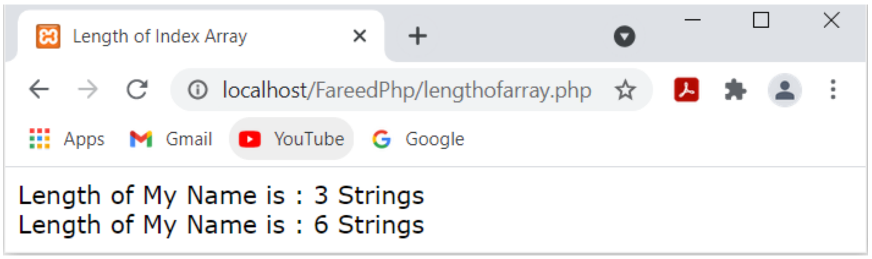
```
My Name is : Shaik
My Name is : Shaik
My Name is : Mohammad
My Name is : Mohammad
My Name is : Fareed
My Name is : Fareed
```

Count Length of PHP Indexed Array: PHP provides count() function which returns length of an array.

```
<html>
        <head><title> Length of Index Array </title></head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                $myname = array ("Shaik", "Mohammad", "Fareed");
                echo " Length of My Name is : ", count($myname), " Strings <br />";
        ?>
        <?php
                $myname = array ("Shaik", "Mohammad", "Fareed", "MCA", "M.Sc(CS)", "IRPM");
                echo "Length of My Name is : ", count($myname), " Strings";
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"lengthofarray.php"

Output:



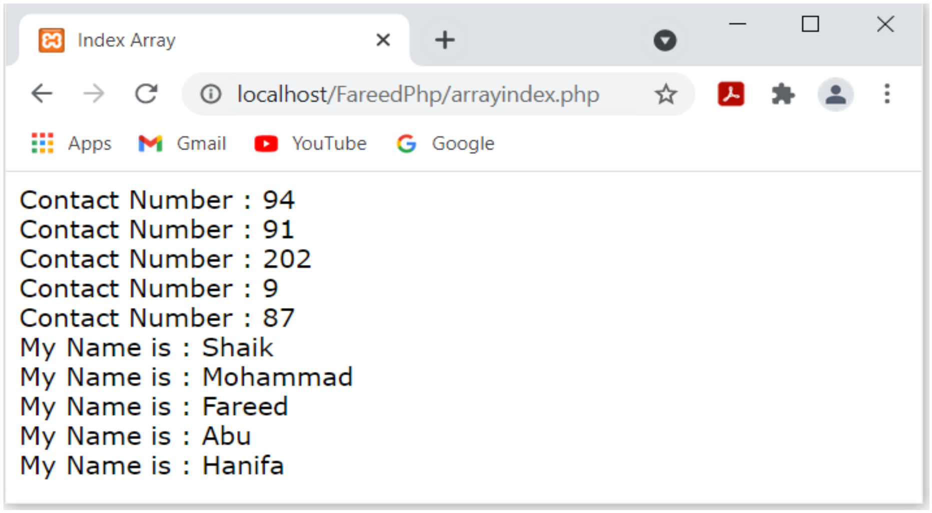
```
Length of My Name is : 3 Strings
Length of My Name is : 6 Strings
```

Example: Illustrate a PHP program using Indexed Arrays?

```
<html>
        <head><title> Index Array </title></head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                $numbers = array ( 94, 91, 202, 9, 87); /* First method to create array. */
                        foreach( $numbers as $value )
                {
                                echo "Contact Number : $value <br />";
                }
                        $numbers[0] = "Shaik";  /* Second method to create array. */
                $numbers[1] = "Mohammad";
                $numbers[2] = "Fareed";
                $numbers[3] = "Abu";
                $numbers[4] = "Hanifa";
                        foreach( $numbers as $value )
                        {
                                        echo "My Name is : $value <br />";
                        }
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"arrayindex.php"

Output:



```
Contact Number : 94
Contact Number : 91
Contact Number : 202
Contact Number : 9
Contact Number : 87
My Name is : Shaik
My Name is : Mohammad
My Name is : Fareed
My Name is : Abu
My Name is : Hanifa
```

(2). Associative array: The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

PHP allows you to associate name/label with each array elements in PHP using **=>** symbol. Such way, you can easily remember the element because each element is represented by label than an incremented number.

NOTE: Don't keep associative array inside double quote while printing otherwise it would not return any value.

Example: To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

There are two ways to define Associative Array:

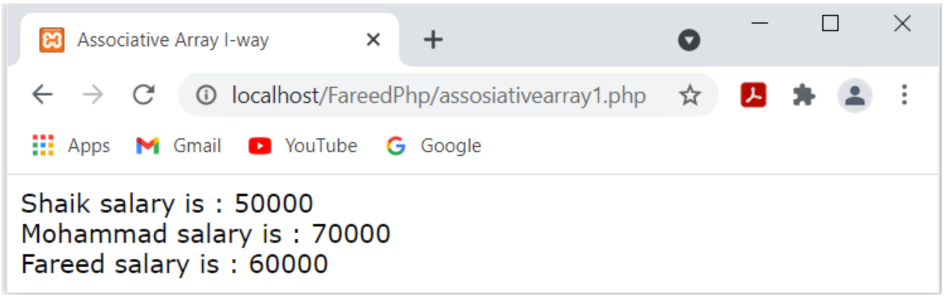
1st Way: $salary = array ("Shaik"=>"50000","Mohammad"=>"70000","Fareed"=>"60000");

2nd Way:
```
$salary[`Shaik']="50000";
$salary[`Mohammad']="70000";
$salary[`Fareed']="60000";
```

Example 1: I-Way
```
    <html>
        <head><title> Associative Array I-Way </title></head>
        <body>
        <font face ="verdana" font size="3">
        <?php
            $salary = array("Shaik" => 50000, "Mohammad" => 70000, "Fareed" => 60000);
            echo "Shaik salary is : ", $salary['Shaik'], "<br />";
            echo "Mohammad salary is : ", $salary['Mohammad'], "<br/>";
            echo "Fareed salary is : ", $salary['Fareed'];
        ?>
    </font>
    </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"assosiativearray1.php"

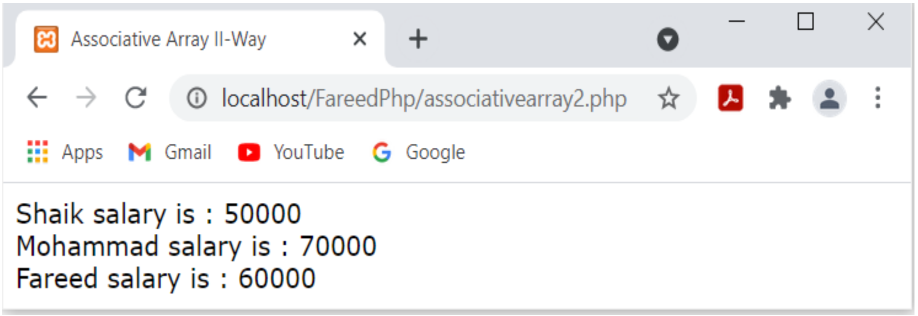
Output:


```
Shaik salary is : 50000
Mohammad salary is : 70000
Fareed salary is : 60000
```

Example 2: II-Way
```
    <html>
        <head><title> Associative Array II-Way </title></head>
        <body>
        <font face ="verdana" font size="3">
        <?php
            $salary['Shaik'] = "50000";
            $salary['Mohammad'] = "70000";
            $salary['Fareed'] = "60000";
            echo "Shaik salary is : ", $salary['Shaik'], "<br/>";
            echo "Mohammad salary is : ", $salary['Mohammad'], "<br/>";
            echo "Fareed salary is : ", $salary['Fareed'];
        ?>
    </font>
    </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"associativearray2.php"

Output:


```
Shaik salary is : 50000
Mohammad salary is : 70000
Fareed salary is : 60000
```

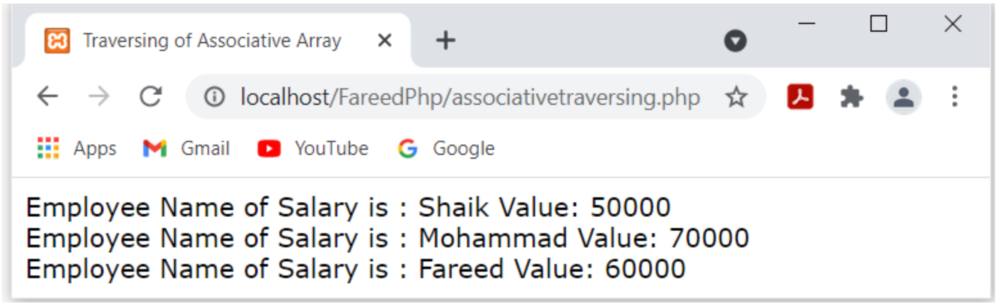
Traversing PHP Associative Array: By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.

Example:

```
<html>
        <head><title> Traversing of Associative Array </title></head>
        <body>
        <font face ="verdana" font size="3">
<?php
        $salary = array("Shaik" => "50000", "Mohammad" => "70000", "Fareed" => "60000");
        foreach ($salary as $names => $val)
        {
                echo "Employee Name of Salary is : ", $names, " Value: ", $val, "<br/>";
        }
?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"associativetraversing.php"

Output:



Traversing of Associative Array
localhost/FareedPhp/associativetraversing.php
Apps
Gmail
YouTube
Google

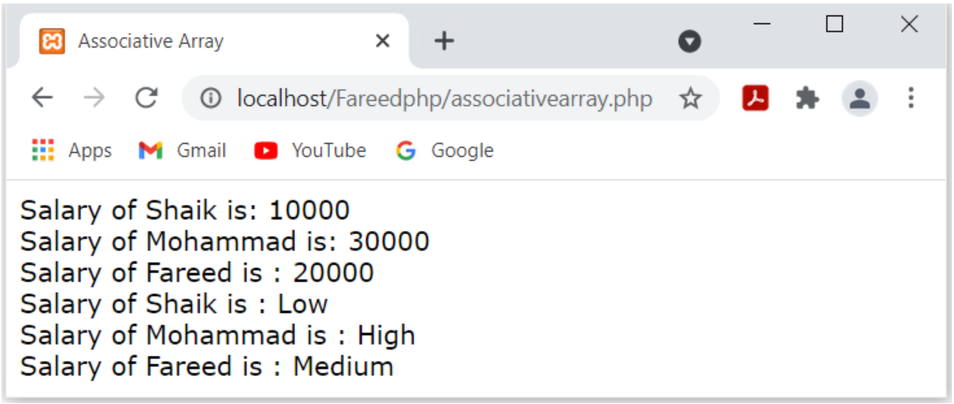
```
Employee Name of Salary is : Shaik Value: 50000
Employee Name of Salary is : Mohammad Value: 70000
Employee Name of Salary is : Fareed Value: 60000
```

Example: Illustrate the PHP program using Associate Array?

```
<html>
        <head><title> Associative Array </title></head>
        <body>
        <font face ="verdana" font size="3">
 <?php
        /* First method to create associate array. */
        $salaries = array ("Shaik" => 10000, "Mohammad" => 30000, "Fareed" => 20000);
        echo "Salary of Shaik is: ", $salaries['Shaik'], "<br />";
         echo "Salary of Mohammad is: ", $salaries['Mohammad'], "<br />";
        echo "Salary of Fareed is : ", $salaries['Fareed'], "<br />";

        /* Second method to create associate array. */
        $salaries['Shaik'] = "Low";
        $salaries['Mohammad'] = "High";
        $salaries['Fareed'] = "Medium";
        echo "Salary of Shaik is : ", $salaries['Shaik'], "<br />";
        echo "Salary of Mohammad is : ", $salaries['Mohammad'], "<br />";
        echo "Salary of Fareed is : ", $salaries['Fareed'], "<br />";
?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"associativearray.php"

Output:



```
Salary of Shaik is: 10000
Salary of Mohammad is: 30000
Salary of Fareed is : 20000
Salary of Shaik is : Low
Salary of Mohammad is : High
Salary of Fareed is : Medium
```

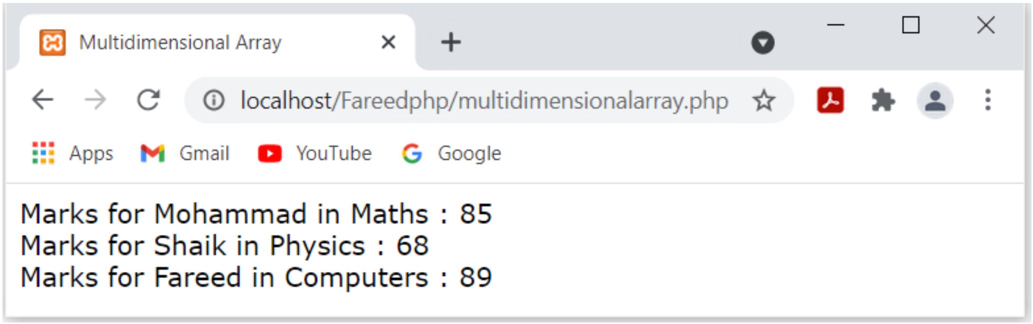
(3). Multidimensional Array: A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple indexes.

Example: In this example we create a two dimensional array to store marks of three students in three subjects. This example is an associative array, you can create numeric array in the same fashion.

```
<html>
        <head><title> Multidimensional Array </title></head>
        <body>
        <font face ="verdana" font size="3">
    <?php
        $marks = array ("Shaik" => array ("Maths" => 78, "Physics" => 68, "Computers" => 69),
                         "Mohammad" => array ("Maths" => 85, "Physics" => 80, "Computers" => 90),
                         "Fareed" => array ("Maths" => 80, "Physics" => 75, "Computers" => 89));

       /* Accessing multi-dimensional array values */

       echo "Marks for Mohammad in Maths : ";
       echo $marks['Mohammad']['Maths'] , "<br />";

       echo "Marks for Shaik in Physics : " ;
       echo $marks['Shaik']['Physics'], "<br />";

       echo "Marks for Fareed in Computers : " ;
       echo $marks['Fareed']['Computers'], "<br />";
    ?>
        </font>
       </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"multidimenionalarray.php"

Output:



```
Marks for Mohammad in Maths : 85
Marks for Shaik in Physics : 68
Marks for Fareed in Computers : 89
```

******* (Q) Explain various Array Functions in PHP?**

PHP Array Functions: PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

(1) PHP array() Function: PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

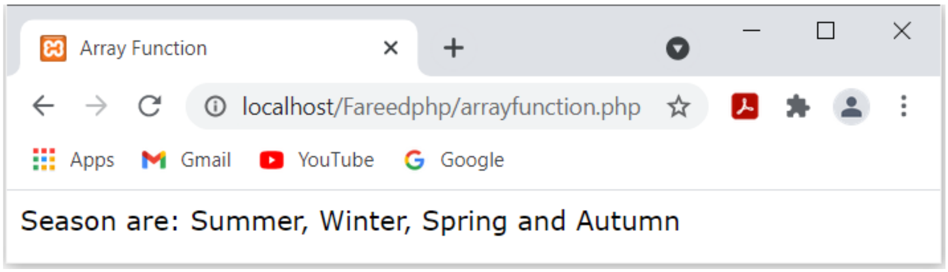
Syntax: array array ([mixed $...])

Example:

```
        <html>
                <head><title> Array Function </title></head>
                <body>
                <font face ="verdana" font size="3">
         <?php
                $season = array ("Summer", "Winter", "Spring", "Autumn");
                echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
        ?>
                </font>
                </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"arrayfunction.php"

Output:



```
Season are: Summer, Winter, Spring and Autumn
```

(2) PHP array_change_key_case() Function: PHP array_change_key_case() function changes the case of all key of an array.

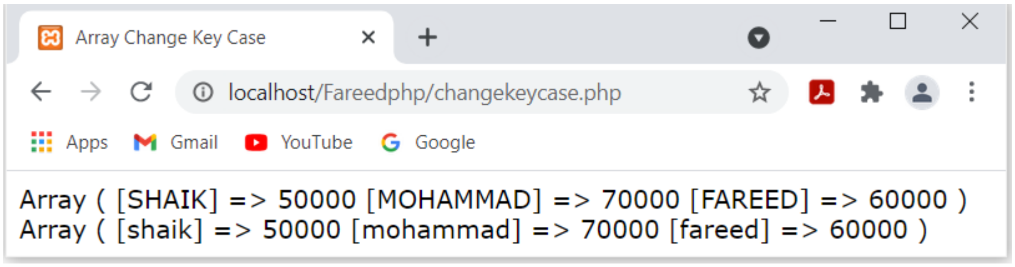
Syntax: array array_change_key_case (array $array [, int $case = CASE_LOWER])

Example:

```
<html>
        <head><title> Array Change Key Case </title></head>
        <body>
        <font face ="verdana" font size="3">
    <?php
        $salary = array("Shaik" => "50000", "Mohammad" => "70000", "Fareed" => "60000");
        print_r(array_change_key_case ($salary, CASE_UPPER));
    ?>
        <?php
                $salary = array("Shaik" => "50000", "Mohammad" => "70000", "Fareed" => "60000");
                print_r(array_change_key_case ($salary, CASE_LOWER));
        ?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"changekeycase.php"

Output:



```
Array ( [SHAIK] => 50000 [MOHAMMAD] => 70000 [FAREED] => 60000 )
Array ( [shaik] => 50000 [mohammad] => 70000 [fareed] => 60000 )
```

(3) PHP count() function: PHP count() function counts all elements in an array.

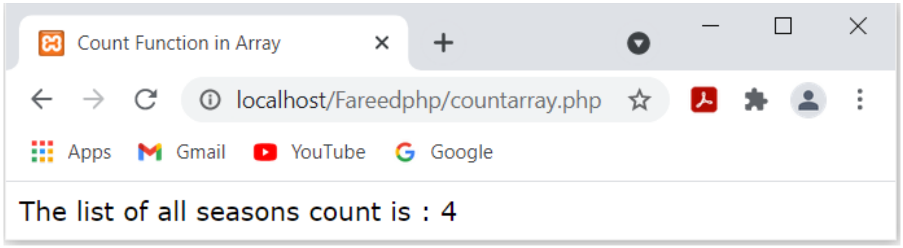
Syntax: int count (mixed $array_or_countable [, int $mode = COUNT_NORMAL])

Example:

```
        <html>
                <head><title> Count Function in Array </title></head>
                <body>
                <font face ="verdana" font size="3">
         <?php
                $season = array ("Summer", "Winter", "Spring", "Autumn");
                echo "The list of all seasons count is : ", count($season);
        ?>
                </font>
                </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"countarray.php"

Output:



```
The list of all seasons count is : 4
```

(4) PHP sort() function: PHP sort() function sorts all the elements in an array.

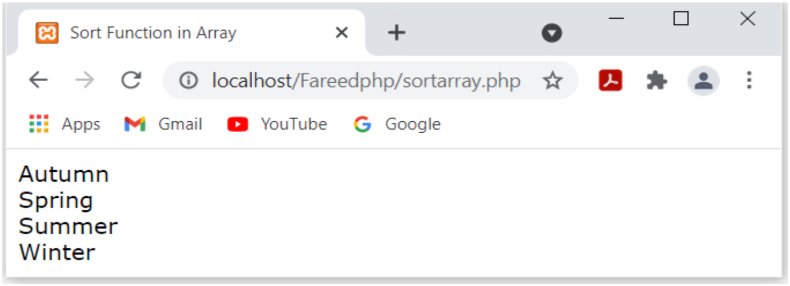
Syntax: bool sort (array &$array [, int $sort_flags = SORT_REGULAR])

Example:

```
        <html>
                <head><title> Sort Function in Array </title></head>
                <body>
                <font face ="verdana" font size="3">
         <?php
                $season = array ("Summer", "Winter", "Spring", "Autumn");
                sort($season);
                foreach( $season as $s )
                {
                        echo "$s<br />";
                }
        ?>
        </font>
        </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"sortarray.php"

Output:



```
Autumn
Spring
Summer
Winter
```

(5) PHP array_reverse() function: PHP array_reverse() function returns an array containing elements in reversed order.

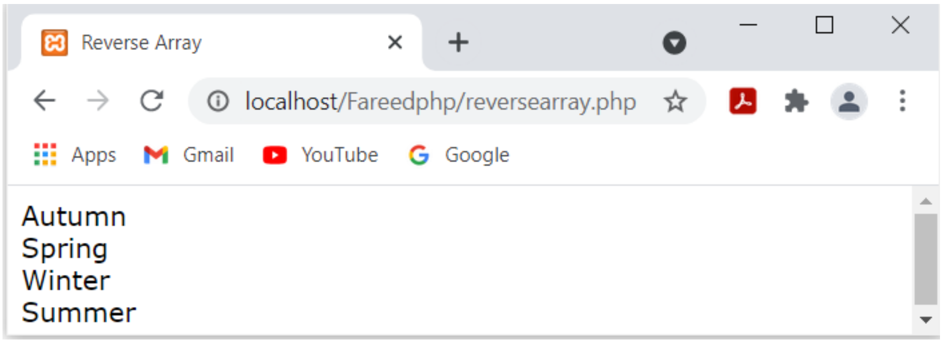
Syntax: array array_reverse (array $array [, bool $preserve_keys = false])

Example:

```
	<html>
		<head><title> Reverse Array </title></head>
		<body>
		<font face ="verdana" font size="3">
	<?php
		$season = array("Summer", "Winter", "Spring", "Autumn");
		$reverseseason = array_reverse ($season);
		foreach ($reverseseason as $s )
		{
			echo "$s<br />";
		}
	?>
		</font>
		</body>
	</html>
```

Save: C:\xampp\htdocs\FareedPhp\"reversearray.php"

Output:


Reverse Array
localhost/Fareedphp/reversearray.php

```
Autumn
Spring
Winter
Summer
```

(6) PHP array_search() function: PHP array_search() function searches the specified value in an array. It returns key if search is successful.

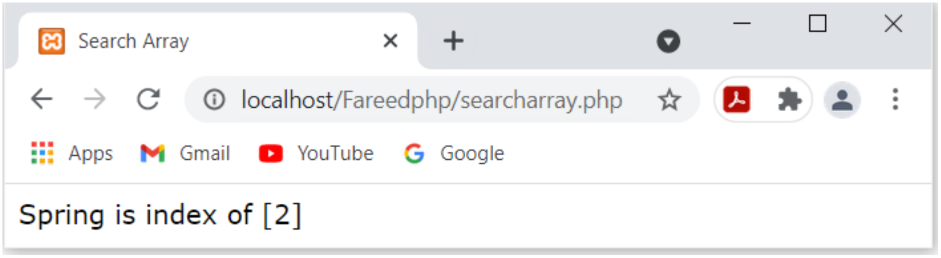
Syntax: mixed array_search (mixed $needle , array $haystack [, bool $strict = false])

Example:

```
	<html>
		<head><title> Search Array </title></head>
		<body>
		<font face ="verdana" font size="3">
	<?php
		$season = array("Summer", "Winter", "Spring", "Autumn");
		$key = array_search ("Spring", $season);
		echo "Spring is index of [", $key, "]";
	?>
		</font>
		</body>
	</html>
```

Save: C:\xampp\htdocs\FareedPhp\"searcharray.php"

Output:


Search Array
localhost/Fareedphp/searcharray.php

```
Spring is index of [2]
```

(7) PHP array_intersect() function: PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two arrays.

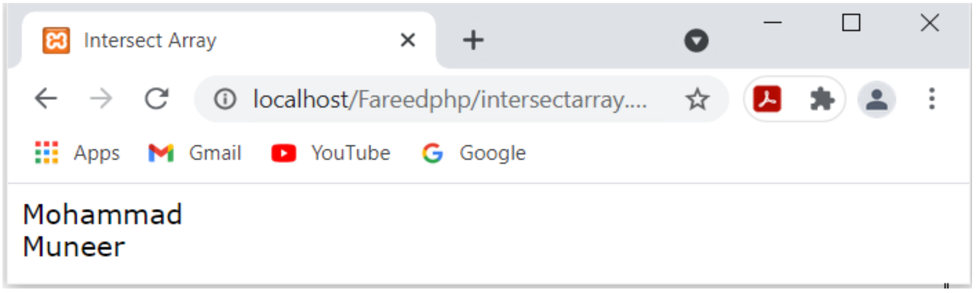
Syntax: array array_intersect (array $array1 , array $array2 [, array $...])

Example:

```
        <html>
                <head><title> Intersect Array </title></head>
                <body>
                <font face ="verdana" font size="3">
        <?php
                $name1 = array ("Shaik", "Mohammad", "Muneer", "Mujeeb");
                $name2 = array ("Fareed", "Masthan", "Muneer", "Mohammad");
                $name3 = array_intersect ($name1, $name2);
                foreach ( $name3 as $n )
                {
                        echo "$n<br />";
                }
        ?>
                </font>
                </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"intersectarray.php"

Output:



```
Mohammad
Muneer
```

(Q) What is Function? Explain detailed array related functions in PHP with an example?

Function: PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value. They are built-in functions but PHP gives you option to create your own functions as well.

There are two parts of function in PHP:

- Creating a PHP Function
- Calling a PHP Function

The real power of PHP comes from its functions; it has more than 1000 built-in functions. PHP User Defined Functions. Besides the built-in PHP functions, we can create our own functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.
- In PHP, we can define Conditional Function, Function within Function and Recursive function also.

Advantage of PHP Functions:

- **Code Reusability**: PHP functions are defined only once and can be invoked many times, like in other programming languages.

- **Less Code**: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

- **Easy to understand**: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-Defined Functions: We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

Syntax for User-Define Function:

```
function function_Name()
{
        Code to be executed
}
```

Note: Function name must be start with letter and underscore only like other labels in PHP. It can't be start with numbers or special symbols.

(I). Creating & Calling PHP Function:

It's very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

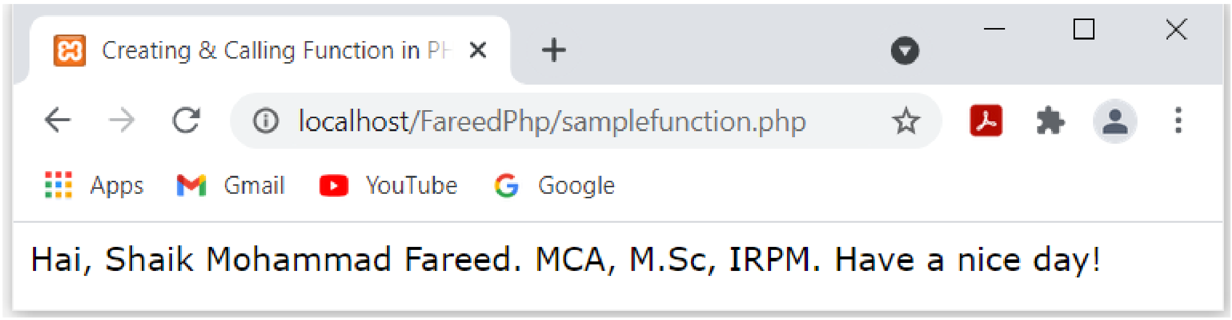
Note that while creating a function its name should start with keyword function and all the PHP code should be put inside { and } braces as shown in the following example below.

Example:

```
<html>
        <head>
                <title>Creating & Calling Function in PHP</title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                function writeMessage()          /* Creating or Declaration a PHP Function */
                {
                        echo "Hai, Shaik Mohammad Fareed. MCA, M.Sc, IRPM. Have a nice day!";
                }
                writeMessage();              /* Definition or Calling a PHP Function */
        ?>
</font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"samplefunction.php"

Output:



Creating & Calling Function in PH
localhost/FareedPhp/samplefunction.php
Apps
Gmail
YouTube
Google

```
Hai, Shaik Mohammad Fareed. MCA, M.Sc, IRPM. Have a nice day!
```

(II). PHP Functions passing with Parameters or Arguments (or) Call by Values:

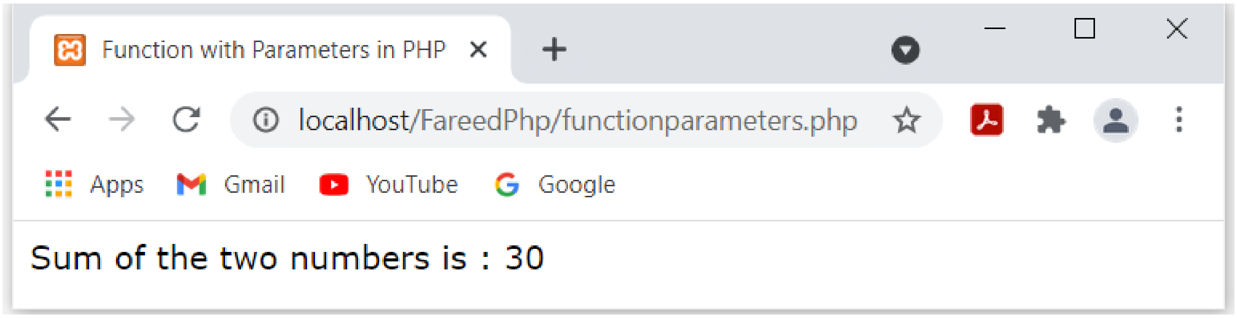
Information can be passed to functions through arguments. PHP gives option to pass your parameters inside a function. You can add or pass as many arguments as you want, just separate them with a comma. These parameters works like variables inside your function.

Example: Write a PHP Program to calculate addition of any two values using functions?

```
<html>
        <head>
                <title>Function with Parameters in PHP</title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                function addFunction($num1, $num2)  /* Create or Declaration of Function Called */
                {
                        $sum = $num1 + $num2;
                        echo "Sum of the two numbers is : $sum";
                }
                addFunction(10, 20);                /* Calling Function or Definition of Function. */
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"functionparameters.php"

Output:


Function with Parameters in PHP
localhost/FareedPhp/functionparameters.php
Apps
Gmail
YouTube
Google

```
Sum of the two numbers is : 30
```

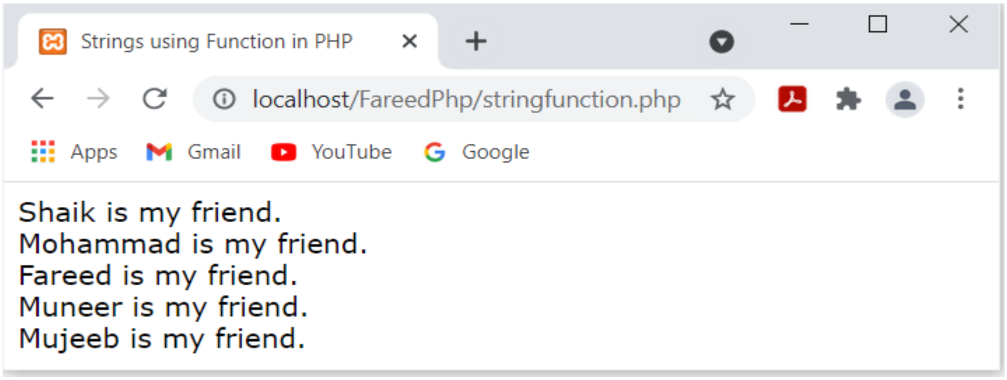
Example: Write a PHP program to display the list of strings using functions?

```
<html>
        <head>
                <title>Strings using Function in PHP</title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                function friendNames ($fname)          /* Create or Declaration of Function Called */
                {
                        echo "$fname is my friend. <br>";
                }
                friendNames("Shaik");
                friendNames("Mohammad");
                friendNames("Fareed");
                friendNames("Muneer");
```

```
            friendNames("Mujeeb");
        ?>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\“stringfunction.php”

Output:


Strings using Function in PHP
localhost/FareedPhp/stringfunction.php
Apps
Gmail
YouTube
Google

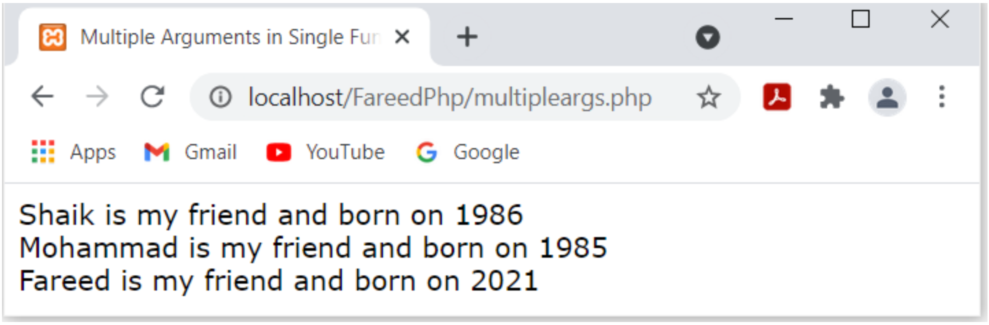
```
Shaik is my friend.
Mohammad is my friend.
Fareed is my friend.
Muneer is my friend.
Mujeeb is my friend.
```

Example: Write a PHP program to passing multiple arguments in function?

```
<html>
        <head>
            <title>Multiple Arguments in Single Function</title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
            function friendNames($fname, $year)    /* Passing multiple arguments in single function */
            {
                echo "$fname is my friend and born on $year <br>";
            }
            friendNames ("Shaik","1986");
            friendNames ("Mohammad","1985");
            friendNames ("Fareed","2021");
        ?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\“multipleargs.php”

Output:


Multiple Arguments in Single Fur
localhost/FareedPhp/multipleargs.php
Apps
Gmail
YouTube
Google

```
Shaik is my friend and born on 1986
Mohammad is my friend and born on 1985
Fareed is my friend and born on 2021
```

(III). Passing Arguments by Reference (or) Call by Reference in function:

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

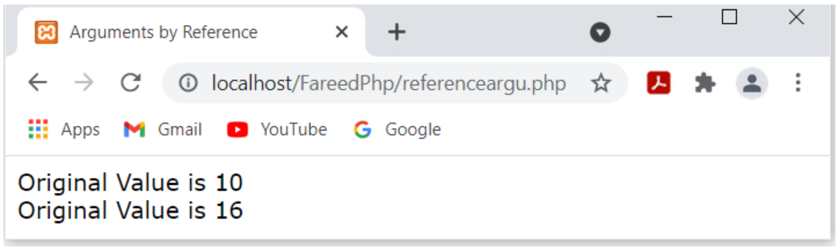
Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand (&) to the variable name in either the function call or the function definition.

Example: Write a PHP program illustrate the Passing Arguments by References in functions?

```
<html>
        <head>
                <title>Arguments by Reference</title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                function addFive($num)      /* without reference value */
                {
                                $num += 5;
                }
                function addSix(&$num)          /* Reference function using & sign */
                {
                                $num += 6;
                }
        $original_num = 10;
        addFive($original_num );
                echo "Original Value is $original_num <br />";
        addSix( $original_num );
                echo "Original Value is $original_num <br />";
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"referenceargu.php"

Output:



```
Original Value is 10
Original Value is 16
```

(IV). PHP Function: Default Argument Value:

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.

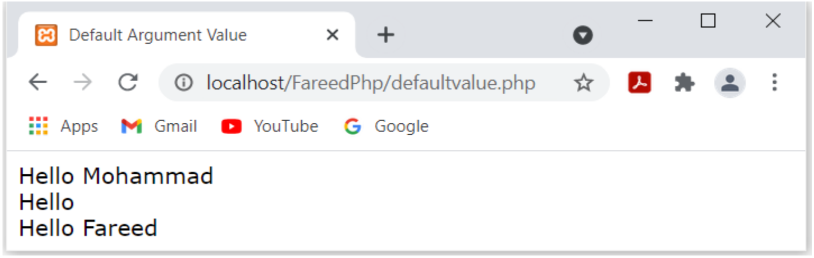
Example: Write a PHP program to illustrate default arguments in function?

```
        <html>
                <head>
                        <title>Default Argument Value</title>
                </head>
                <body>
                <font face ="verdana" font size="3">
                <?php
                        function sayHello($name="Mohammad")
                        {
```

```
                echo "Hello $name<br/>";
            }
            sayHello();             /* passing no value  or default value in function */
            sayHello(null);         /* passing null or empty value */
            sayHello("Fareed");     /* passing specific value */
        ?>
        </font>
        </body>
    </html>
```

Save: C:\xampp\htdocs\FareedPhp\"defaultvalue.php"

Output:



```
Hello Mohammad
Hello
Hello Fareed
```

(V) PHP Function: Returning Values:

A function can return a value using the return statement in conjunction with a value or object. Return stops the execution of the function and sends the value back to the calling code.

You can return more than one value from a function using **return array (1, 2, 3, 4)**.

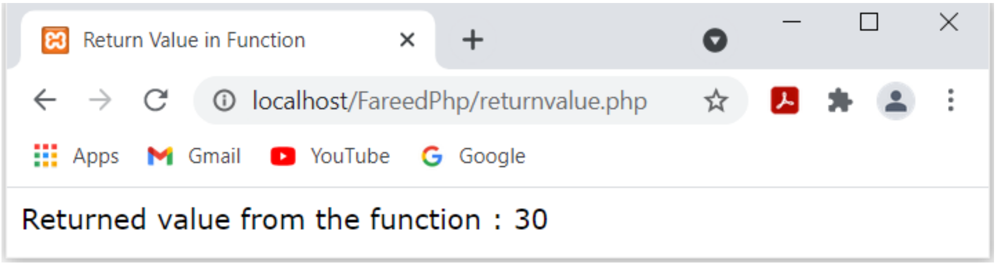
Following example takes two integer parameters and adds them together and then returns their sum to the calling program. Note that return keyword is used to return a value from a function.

Example: Write a PHP program to illustrate the Return Values in Function?

```
    <html>
        <head>
            <title>Return Value in Function</title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
            function addFunction($num1, $num2)
            {
                        $sum = $num1 + $num2;
                        return $sum;
            }
            $return_value = addFunction(10, 20);
            echo "Returned value from the function : $return_value";
        ?>
        </font>
        </body>
    </html>
```

Save: C:\xampp\htdocs\FareedPhp\"returnvalue.php"

Output:



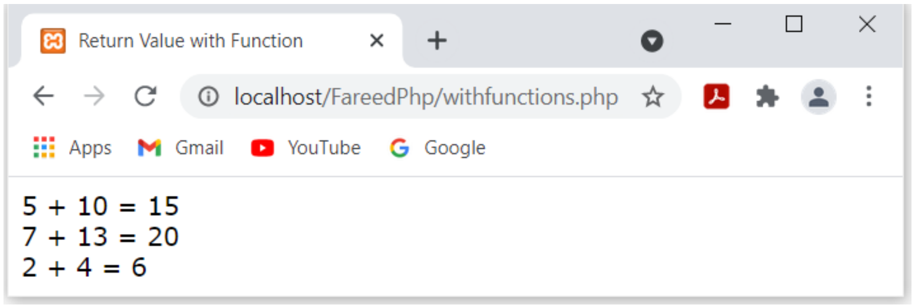
```
Returned value from the function : 30
```

Example: Write a PHP program to illustrate the Return Values in Function?

```
        <html>
                <head>
                        <title>Return Value with Function</title>
                </head>
                <body>
                <font face ="verdana" font size="3">
                <?php
                        function sum($x, $y)
                        {
                                $z = $x + $y;
                                return $z;
                        }
                        echo "5 + 10 = " .sum(5,10), "<br>";
                        echo "7 + 13 = " .sum(7,13),  "<br>";
                        echo "2 + 4 = " .sum(2,4);
                ?>
                </font>
                </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"withfunctions.php"

Output:


Return Value with Function
localhost/FareedPhp/withfunctions.php
Apps
Gmail
YouTube
Google

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```

(VI). PHP Function: Parameterized Functions:

PHP Parameterized functions are the functions with parameters. You can pass any number of parameters inside a function. These passed parameters act as variables inside your function.

They are specified inside the parentheses, after the function name. The output depends upon the dynamic values passed as the parameters into the function.

Example: Write a PHP program to illustrate Addition and Subtraction in functions?

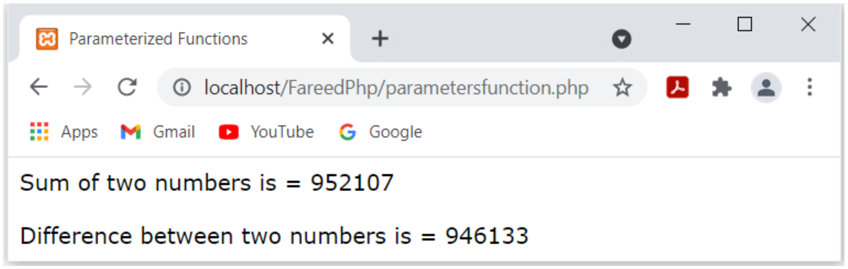
In this example, we have passed two parameters **$x** and **$y** inside two functions **add()** and **sub()**.

```
<html>
        <head>
                <title> Parameterized Functions </title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
```

```
                function add($x, $y)      /* Adding two numbers  */
                {
                        $sum = $x + $y;
                        echo "Sum of two numbers is = $sum <br><br>";
                }
                add(949120, 2987);
                        function sub($x, $y)          /* Subtracting two numbers  */
                        {
                                $diff = $x - $y;
                                echo "Difference between two numbers is = $diff";
                        }
                        sub(949120, 2987);
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"parametersfunction.php"

Output:



```
Sum of two numbers is = 952107

Difference between two numbers is = 946133
```

(VII). PHP Function: Dynamic Function Call:

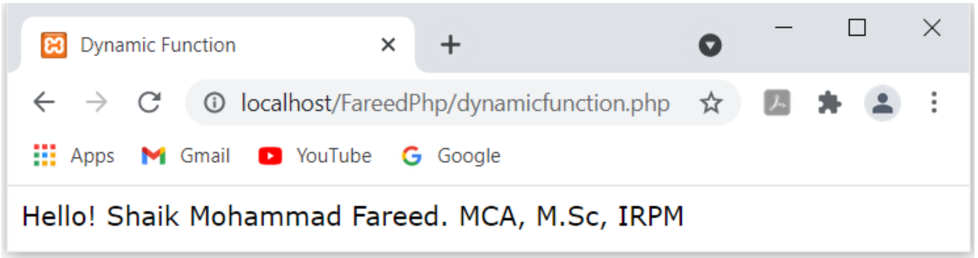
It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself. Following example depicts this behavior.

Example: Write a PHP program to illustrate the dynamic function call values?

```
        <html>
                <head>
                        <title>Dynamic Function</title>
                </head>
                <body>
                <font face ="verdana" font size="3">
                <?php
                        function sayHello()
                        {
                                echo "Hello! Shaik Mohammad Fareed<br />";
                        }
                        $function_holder = "sayHello";
                        $function_holder();
                ?>
                </font>
                </body>
        </html>
```

Save: C:\xampp\htdocs\FareedPhp\"dynamicfunction.php"

Output:



```
Hello! Shaik Mohammad Fareed. MCA, M.Sc, IRPM
```

(VIII). PHP Function: Variable Length Argument Function:

PHP supports variable length argument function. It means you can pass 0, 1 or n number of arguments in function. To do so, you need to use 3 ellipses (dots) before the argument name.

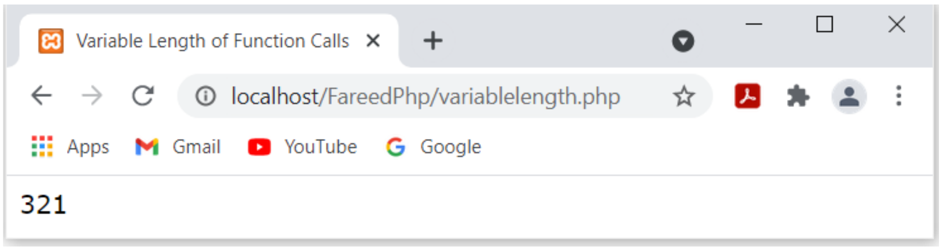
The 3 dot (...) concept is implemented for variable length argument since PHP 5.6. Let's see a simple example of PHP variable length argument function.

Example: Write a PHP program to illustrate the Variable Length of Function Calls?

```
<html>
        <head>
                <title>Variable Length of Function Calls</title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                function add(...$numbers)
                {
                        $sum = 0;

                        foreach ($numbers as $n)
                        {
                                $sum += $n;
                        }
                return $sum;
                }
                echo add(94, 91, 20, 29, 87);
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"variablelength.php"

Output:


Variable Length of Function Calls
localhost/FareedPhp/variablelength.php
Apps
Gmail
YouTube
Google

```
321
```

(IX). PHP Function: Recursive Function:

PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.

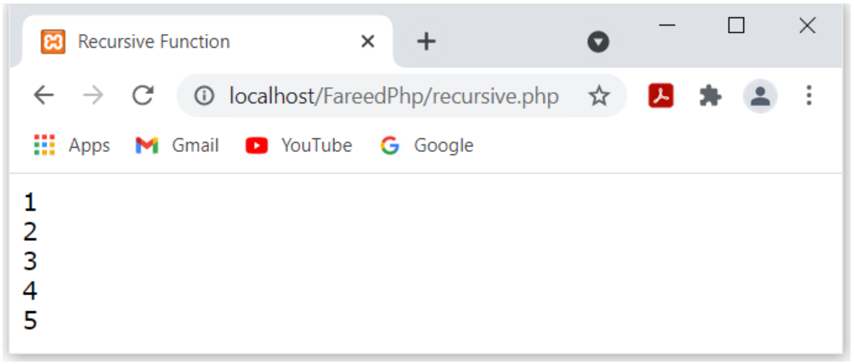
Example : Write a PHP program to printing numbers using recursive function?

```
<html>
        <head>
                <title>Recursive Function</title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
```

```
                function display($number)
                {
                        if($number<=5)
                        {
                                echo "$number <br/>";
                                display($number+1);
                        }
                }
                display(1);
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\“recursive.php”

Output:



Recursive Function

localhost/FareedPhp/recursive.php

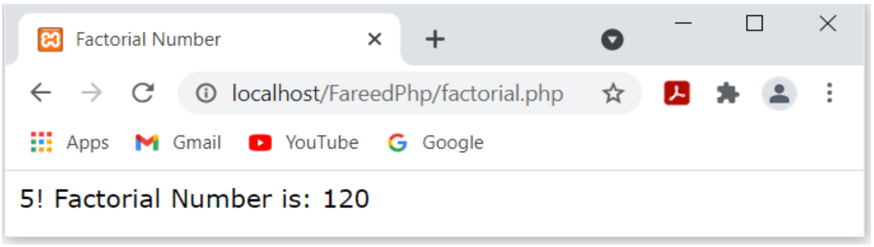
```
1
2
3
4
5
```

Example: Write a PHP program to calculate factorial number using recursive function?

```
<html>
        <head>
                <title> Factorial Number </title>
        </head>
        <body>
        <font face ="verdana" font size="3">
        <?php
                function factorial($n)
                {
                        if ($n < 0)
                        return -1;                      /*Wrong value*/
                                if ($n == 0)
                                return 1;       /*Terminating condition*/
                                return ($n * factorial ($n -1));
                }
                echo "5! Factorial Number is: ” , factorial(5);
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\“factorial.php”

Output:



Factorial Number

localhost/FareedPhp/factorial.php

```
5! Factorial Number is: 120
```

Chapter 2: Working with Objects

Introduction:

In PHP Classes and Objects are very flexible in accessing member and member functions. Specially there ate list of concepts in PHP program.

- **Class:** This is a programmer-defined data type, which included local functions as well as local data. A class as a template for making many instances (objects) of the same kind of class.

- **Object:** An individual instance of the data structure defined by a class. If define a class once and then make many objects that belong to it. Objects are also known as instance.

- **Member Variables:** These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attributes of the object once an object is created.

- **Member Function:** These are the function defined inside a class and are used to access object data in outside of a class.

- **Inheritance:** When a class is define by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.

- **Parent Class:** A class that is generated / inherited another class. This is also called as base class or super class.

- **Child Class:** A class that access / inherits from another class. This is also called as subclass or derived class.

- **Polymorphism:** This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it makes take different number of arguments and can do different task.

- **Overloading:** A type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.

- **Data Abstraction:** Any representation of data in which the implementation details are hidden.

- **Encapsulation:** Refers to a concept where we encapsulated all the data and members functions together to form an object.

- **Constructor:** Refers to a special type of function which will be called automatically wherever there is an object formation from a class.

- **Destructor:** Refers to a special type of function which will be called automatically whenever an object is deleting or goes out of scope.

VVIMP********* (Q) Explain how to create a Classes and Objects in PHP?**

(I). Creating a Classes in PHP: The general form for defining a new classed in PHP is as follows:

```
<?php
        class phpClass_name        // Creating a Class.
        {
                var $var1;
                var $var1;
                 - - - - - - - - -
                - - - - - - - - -
                function function_name (&arg1, &arg2, ….)          // Declare a Function.
                {
                        - - - - - - - - -
                        - - - - - - - - -
                        - - - - - - - - -
                }
        }
?>
```

- Here the special form class followed by the name of the class that you want to define.
- A set of braces enclosing any number of variables declarations and function declarations.
- Variables declarations start with the special for "var", which is followed by a conventional $ variable name; they may also have an initial assignment to a constant value.
- Function declaration look much like standalone PHP functions but are local to the class and will be used to set and access object data in outside of a class.

Example: Illustrate a class and its members in PHP?

```
<?php
        class Spectrums
        {
                        var $pages;          // Members of Variables in a class Spectrums.
                        var $titles;
                        function setTitle($fareed)      // Member Function in class Spectrums.
                        {
                                $this -> titles = $fareed;
                        }
                        function getTitle()      // Member Function in class Spectrums.
                        {
                                echo $this -> titles, "<br/>";
                        }
                        function setPages($fareed)      // Member Function in class Spectrums.
                        {
                                $this -> pages = $fareed;
                        }
                        function getPages()      // Member Function in class Spectrums.
                        {
                                echo $this -> pages, "<br/>";
                        }
        }
?>
```

NOTE: The variable "$this" is a special variable and it refers to the same object therefore itself.

(II). Creating an Objects in PHP: Once you defined your class, then you can create as many objects as you like of that class type. Object is an instance of class. If you create objects in PHP using “new” keyword.

Syntax: object_name = new class_name();

Example:

```
$paper7 = new Spectrums;
$cluster1 = new Spectrums;
$cluster2 = new Spectrums;
```

Here we have created three objects and these objects are independent of each other and they will have their existence separately.

(III). Calling Member Functions: After creating your objects, you will be able to call remember functions related to that object.

Example: Following example shows how to set price and title for the three books by calling member functions.

```
$paper7 -> setTitle ("Paper - VII: Web-Technology");
$cluster1 -> setTitle ("Cluster1 : PHP-MySQL & Wordpress");
$cluster2 -> setTitle ("Cluster2 : Advanced JavaScript");

$paper7 -> setPages 168);
$cluster1 -> setPages (120);
$cluster2 -> setPages (130);
```

Now you call another member functions to get the values set by in above:

```
$paper7 -> getTitle ();
$cluster1 -> getTitle ();
$cluster2 -> getTitle ();

echo "Total Number of Pages in Web-Technology : ", $paper7 -> getPages();
echo "Total Number of Pages in PHP-MYSQL & Wordpress : ", $cluster1 -> getPages();
echo "Total Number of Pages in Advanced JavaScript : ", $cluster2 -> getPages();
```

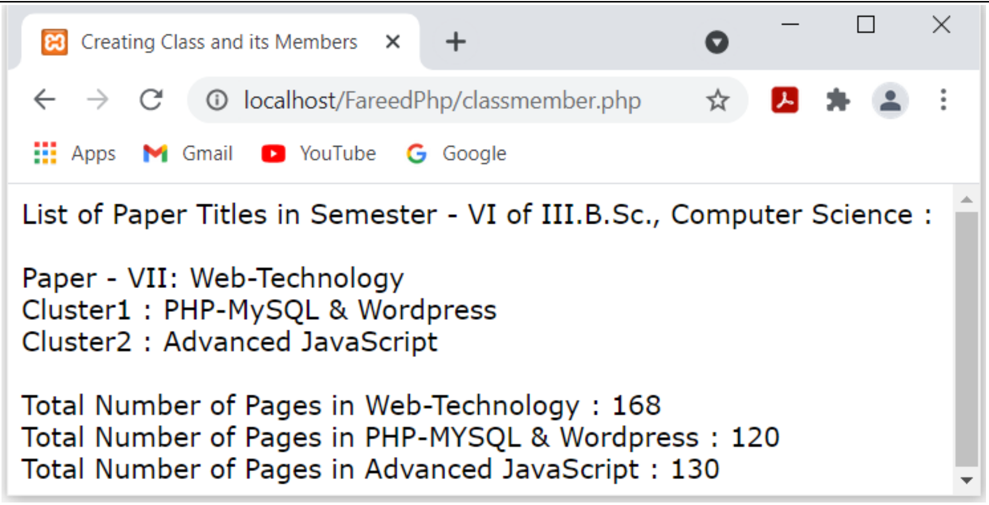
Example: Write a PHP program to display semester-vi spectrums information using classes and objects?

```
<html>
        <head><title> Creating Class and its Members </title></head>
        <body>
        <font face ="verdana" font size="3">
     <?php
        class Spectrums
        {
                        var $pages;                     // Members of Variables in a class Spectrums.
                        var $titles;
                        function setTitle ($fareed)             // Member Function in class Spectrums.
                        {
                                $this -> titles = $fareed;
                        }
```

```
                function getTitle()                    // Member Function in class Spectrums.
                {
                        echo $this -> titles, "<br/>";
                }
                function setPages ($fareed)            // Member Function in class Spectrums.
                {
                        $this -> pages = $fareed;
                }
                function getPages()                    // Member Function in class Spectrums.
                {
                        echo $this -> pages, "<br/>";
                }
        }
                /* Creating an Objects on Class Spectrums*/

                $paper7 = new Spectrums;
                $cluster1 = new Spectrums;
                $cluster2 = new Spectrums;

            /* Calling Member Functions from Class Spectrums*/

            $paper7 -> setTitle ("Paper - VII: Web-Technology");
            $cluster1 -> setTitle ("Cluster1 : PHP-MySQL & Wordpress");
            $cluster2 -> setTitle ("Cluster2 : Advanced JavaScript");

            $paper7 -> setPages (168);
            $cluster1 -> setPages (120);
            $cluster2 -> setPages (130);

        /* Now you call another member functions to get the values set by in above: */

        echo "List of Paper Titles in Semester - VI of III.B.Sc., Computer Science :  <br />";
        echo "<br />";
        $paper7 -> getTitle();
        $cluster1 -> getTitle();
        $cluster2 -> getTitle();

        echo "<br />";
        echo "Total Number of Pages in Web-Technology : ", $paper7 -> getPages();
        echo "Total Number of Pages in PHP-MYSQL & Wordpress : ", $cluster1 -> getPages();
        echo "Total Number of Pages in Advanced JavaScript : ", $cluster2 -> getPages();
  ?>
</font>
</body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"classmember.php"

Output:

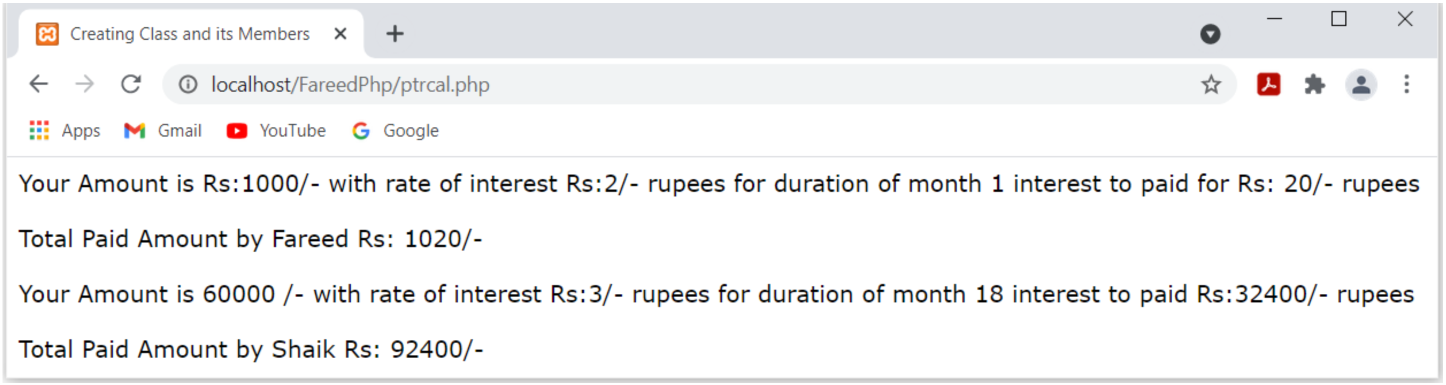

Creating Class and its Members
localhost/FareedPhp/classmember.php
Apps
Gmail
YouTube
Google

```
List of Paper Titles in Semester - VI of III.B.Sc., Computer Science :

Paper - VII: Web-Technology
Cluster1 : PHP-MySQL & Wordpress
Cluster2 : Advanced JavaScript

Total Number of Pages in Web-Technology : 168
Total Number of Pages in PHP-MYSQL & Wordpress : 120
Total Number of Pages in Advanced JavaScript : 130
```

Example: Write a PHP program how to calculate simple interest for given principle amount using class and objects?

```
<html>
        <head><title> Creating Class and its Members </title></head>
        <body>
        <font face ="verdana" font size="3">
     <?php
        class interest
        {
                public $rate;          // Members of Variables in a class interest.
                public $time;
                public $amount;
                public $fareedtotal;
                public $shaiktotal;
                        public function interestCompute()
                        {
                                return ($this -> amount * $this -> time * $this -> rate ) /100;
                        }
        }
        $fareed = new interest();
        $fareed -> amount = 1000;
        $fareed -> time = 1;
        $fareed -> rate = 2;
        $total_amount1 = $fareed -> interestCompute();

        $shaik = new interest();
        $shaik -> amount = 60000;
        $shaik -> time = 18;
        $shaik -> rate = 3;
        $total_amount2 = $shaik -> interestCompute();

echo "Your Amount is Rs:", $fareed -> amount, "/- with rate of interest Rs:", $fareed -> rate, "/- rupees for duration of month ", $fareed -> time, " interest to paid for Rs: ", $total_amount1,"/- rupees <br />";
echo "<br />";
        $fareedtotal = $fareed -> amount + $total_amount1;
        echo " Total Paid Amount by Fareed Rs: ", $fareedtotal, "/- <br />";
echo "<br />";

echo "Your Amount is ", $shaik -> amount, " /- with rate of interest Rs:", $shaik -> rate, "/- rupees for duration of month ", $shaik -> time, " interest to paid Rs:", $total_amount2,"/- rupees <br />";

echo "<br />";

        $shaiktotal = $shaik -> amount + $total_amount2;
        echo " Total Paid Amount by Shaik Rs: ", $shaiktotal, "/- <br/>";

        ?>
</font>
<body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\"ptrcal.php"
Output:



Creating Class and its Members
localhost/FareedPhp/ptrcal.php
Apps
Gmail
YouTube
Google

```
Your Amount is Rs:1000/- with rate of interest Rs:2/- rupees for duration of month 1 interest to paid for Rs: 20/- rupees

Total Paid Amount by Fareed Rs: 1020/-

Your Amount is 60000 /- with rate of interest Rs:3/- rupees for duration of month 18 interest to paid Rs:32400/- rupees

Total Paid Amount by Shaik Rs: 92400/-
```

*********** (Q) Define a Constructor Functions in PHP?**

(I). Constructor Function:

- Constructor functions are special type of functions which are called automatically whenever an object is created.
- If you create a "__construct()" function, PHP will automatically call this function when you create an object from a class.
- PHP provides a special function called "**_construct()**" to define a constructor.
- Notice that the construct function starts with two underscores (__)!
- You can pass as many as arguments you like into the constructor function.

Syntax for Constructor:

```
function __construct(args1, args2, . . . )
{
        // initialize the object and its properties by assigning values
}
```

Note: The constructor is defined in the public section of the Class. Even the values to properties of the class are set by Constructors.

Example: The following example will create one constructor for Spectrums class and it will initialize pages and titles for the spectrums at the time of object creation.

```
                function__construct ($fareed1, $fareed2)
                 {
                        $this →titles = $fareed1;
                        $this →pages = $fareed2;
                 }
```

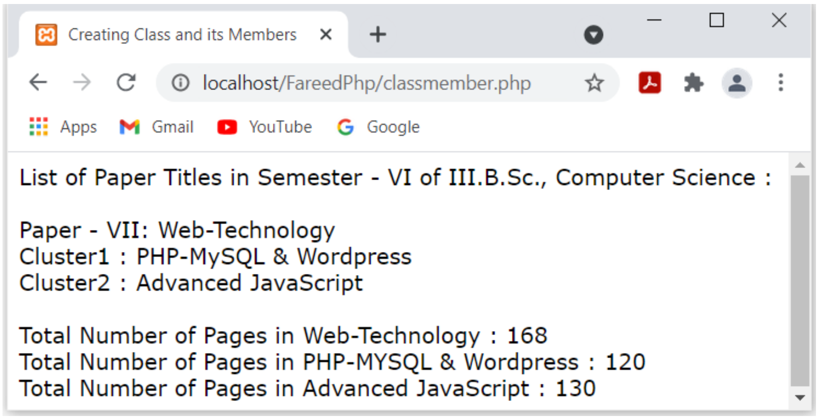
Now we don't need to call set function separately to set titles and pages. We can initialize these two members' variables at the time of object creation only.

```
        $paper7 = new Spectrums ("Paper - VII: Web-Technology", 168);
        $cluster1 = new Spectrums ("Cluster1: PHP-MySQL & Wordpress", 120);
        $cluster2 = new Spectrums ("Cluster2: Advanced JavaScript", 130);
```

Now Get those Set value:

```
        $paper7 -> getTitle();
        $cluster1 -> getTitle();
        $cluster2 -> getTitle();

                $paper7 -> getPages ();
                $cluster1 -> getPages ();
                $cluster2 -> getPages ();
```

Output:


Creating Class and its Members
localhost/FareedPhp/classmember.php
Apps
Gmail
YouTube
Google

```
List of Paper Titles in Semester - VI of III.B.Sc., Computer Science :

Paper - VII: Web-Technology
Cluster1 : PHP-MySQL & Wordpress
Cluster2 : Advanced JavaScript

Total Number of Pages in Web-Technology : 168
Total Number of Pages in PHP-MYSQL & Wordpress : 120
Total Number of Pages in Advanced JavaScript : 130
```

Types of Constructor:

1. **Default Constructor:** It has no parameters, but the values to the default constructor can be passed dynamically.

2. **Parameterized Constructor:** It takes the parameters, and also you can pass different values to the data members.

3. **Copy Constructor:** It accepts the address of the other objects as a parameter.

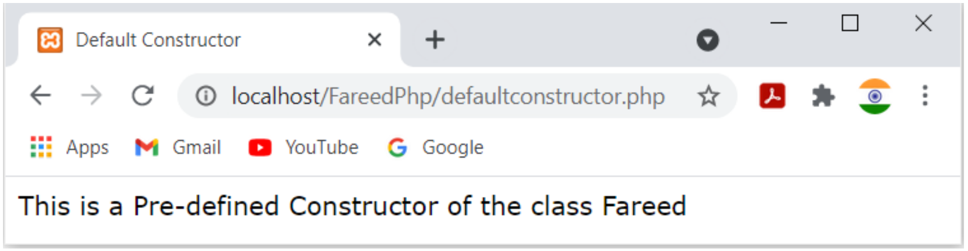
1. Default Constructor: It has no parameters, but the values to the default constructor can be passed dynamically.

Example:

```
                <html>
                        <head><title> Default Constructor </title></head>
                        <body>
                        <font face ="verdana" font size="3">
                        <?PHP
                                class Fareed
                                {
                                        function Fareed()
                                        {
                                                echo "This is User-defined Constructor of the class Fareed";
                                         }

                                         function __construct()         // Default Contractor
                                        {
                                                echo "This is a Pre-defined Constructor of the class Fareed";
                                        }
                                }
                                $obj= new Fareed();
                        ?>
                        </font>
                        </body>
                </html>
```

Save: C:\xampp\htdocs\FareedPhp\“defaultconstructor.php”

Output:



```
This is a Pre-defined Constructor of the class Fareed
```

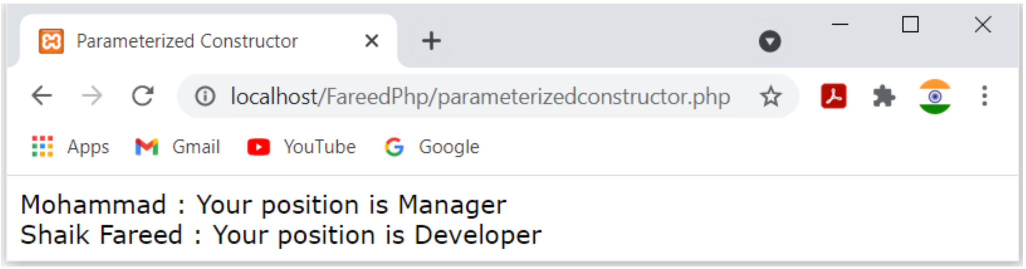
2. Parameterized Constructor: The constructor of the class accepts arguments or parameters. The '->' operator is used to set value for the variables. In the constructor method, you can assign values to the variables during object creation.

Example:

```
<html>
        <head><title> Parameterized Constructor </title></head>
        <body>
        <font face ="verdana" font size="3">
    <?PHP
        class Employee
        {
                public $name;
                public $position;
                function __construct($name, $position)
                {
                        $this -> name = $name;          // This is initializing the class properties
                        $this -> profile = $position;
                }
                function show_details()
                {
                        echo $this -> name." : ";
                        echo "Your position is ", $this->profile, "<br />";
                }
        }
                $employee2 = new Employee("Mohammad", "Manager");
                $employee2 -> show_details();

                $employee_obj = new Employee("Shaik Fareed", "Developer");
                $employee_obj -> show_details();
        ?>
        </font>
        </body>
</html>
```

Save: C:\xampp\htdocs\FareedPhp\“parameterizedconstructor.php”

Output:


Parameterized Constructor
localhost/FareedPhp/parameterizedconstructor.php
Apps
Gmail
YouTube
Google

```
Mohammad : Your position is Manager
Shaik Fareed : Your position is Developer
```

(II). Calling Parent Constructors: Instead of writing an entirely new constructor for the subclass, let's write it by calling the parent's constructor explicitly and then doing whatever is necessary in addition for instantiation of the subclass.

Example:

```
class Name
{
   var $_fname;
   var $_lname;
    function Name($first_name, $last_name)
        {
            $this →_fname = $first_name;
            $this →_lname = $last_name;
        }
   function toString()
         {
            return($this →_fname. "," .$this →_lname);
         }
}
Class MyName extends Name
{
   var $_mname;
    function MyName($first_name, $middle_name, $last_name)
      {
            Name::Name($first_name, $last_name);              // Calling Parent Constructor.
            $this →_mname = $middle_name;
       }
    function toString()
     {
          return(Name::toString(). " ".$this →_mname);
     }
}
```

Here:

- In this example, we have a parent class (Name); witch has a two arguments constructor, and a subclass (MyName), which has a three-argument constructor.
- The constructor of MyName functions by calling its parent constructor explicitly using the :: syntax (passing two of its arguments along) and then setting as additional field.
- Similarly MyName defines its non constructor toString() function in terms of the parent function that it overrides.

(III).Destructor: Like a constructor function you can define a destructor function using "**_destruct()**". You can release all the resources with-in a destructor.

(III).Destructor: Like a constructor function you can define a destructor function using "**_destruct()**". You can release all the resources with-in a destructor.

(Q) How to define Inheritance in PHP?

Inheritance: In PHP class definitions can optionally inherit from a parent class definition by using the extends clauses.

Syntax:

```
class Child_Name extends Parent_Class
 {
        - - - - - - - - - - - -
        - - - - - - - - - - - -   // Body Definition;
        - - - - - - - - - - - -
 }
```

The effect of inheritance is that child class (or subclass or derived class) has the following characteristics:

- Automatically have all the member variables declarations of the parent class.
- Automatically have all the same member functions as the parent, which (by default) will work the same way as those functions do in the parent.

Example: The following example inherits Books class and adds more functionality based on the requirements.

```
class Novel extends Books
{
    var $publisher;
     function setPublisher($par)
        {
          $this → publisher = $par;
         }
function getPublisher()
   {
      echo  $this → publisher. "<br>";
   }
}
```

(Q) How to define Function Overriding in PHP?

Function Override: Function definitions in child classes override definition with the same name in parent classes. In a child class, we can modify the definition of a function inherited from parent classes.

Example: Function Override outside of the class:

```
<?php
class Name
{
  function nameShaik()        //Function Name
    {
       echo "My Name is Mohammad";
    }
}
```

```
class MyName extends Name
{
   function nameShaik()        // Function Override. Same function name with different classes.
   {
     echo " My Name is Fareed";
   }
}
        $f1 = new Name;
        $f2 = new MyName;
                echo($f1 → Name());
                echo($f2 → MyName());
?>
```

Output:
```
My Name is Mohammad
My Name is Fareed
```

Example: Function Override inside of the class:

```
Class Addition
{
   function compute($first, $second)
     {
        return $first + $second;
     }
   function compute($first, $second, $third)      // Function Override inside of class.
     {
         return $first+$second+$third;
     }
}
```

VVIMP***(Q) Explain various types of Investigation/Manipulation Strings Functions in PHP?**

(or)

Explain the various types of Built-in String Functions in PHP?

String: A String is a sequence of letters, numbers, special characters and arithmetic values or combination of all. How Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain escape/character sequences.

Example: Clarify the differences between single and double quote stings:
```
<?php
        $variable = "name";
         $literally = 'My $variable will not print!\\n';
                print($literally);
                print "<br />";
                        $literally = "My $variable will print!\\n";
                        print($literally);
?>
```

Output:
```
My $variable will not print!\n
My name will print!\n
```

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

Investigating String Functions / Manipulating String Functions PHP Strings: in PHP provides many built-in functions for manipulating strings like calculate the length of a string, find substrings or characters, replacing part of a string with different characters, take a string apart, and many others.

If fact, strings and arrays are not as different as you might imagine. You can think of a string as an array of characters. So you can access individual characters of a string as if they were elements of an array.

```
$name = "ShaikMohammadFareed";
echo $name[0];   → display 'S'.
echo $name[5];   → display 'M'.
echo $name[13];  → display 'F'.
```

It is important to remember, therefore that when we talk about the positions or index of a characters within a string. Characters, like array elements, are indexed from zero (0).

(a). Calculate the Length of a String: The **"strlen()"** function is used to calculated the number of characters inside a string. It also includes the blank spaces inside the string.

```
<html>
<body>
<?php
echo strlen("Hello Shaik Mohammad Fareed!!!");
?>
</body>
</html>
```

Output: 23

(b). Count The Number of Words in a String: The PHP **"str_word_count()"** function counts the number of words in a string:

```
<html>
<body>
<?php
echo str_word_count("Hello This is Shaik Mohammad Fareed");
?>
 </body>
</html>
```

Output: 6

(C). Replacing Test within String: the **"str_replace()"** function to perform replaces all occurrences of the search text within target string.

```
<html>
<body>
<?php
        $my_name = 'Shaik Muneer Pasha';
        echo str_replace("Muneer", "Mujeeb", $my_name);
?>
</body>  </html>
```

Output: Shaik Mujeeb Pasha.

(d). Reserving / Mirror String: The **"strrev()"** function is used to perform reverses a strings.

```
<html>
<body>
<?php
        echo strrev("Shaik Mohammad Fareed!");
?>
</body>
</html>
```

Output: !deeraF dammahoM kiahS

(e). Search For a Specific Text Within a String: The PHP **"strpos()"** function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

```
<html>
<body>
<?php
        echo strpos("Shaik Mohammad Fareed", "Fareed");
?>
</body>
</html>
```

Output: 14

NOTE: The first character position in a string is 0 (not 1).

(f). String Concatenation Operator: To concatenate two string variables together, use the dot (**.**) operator.

```
<html>
<body>
<?php
        $string1="Shaik Mohammad Fareed";
        $string2="9491202987";
        echo $string1 . " " . $string2;
?>
</body>
```

```
</html>
```

Output: Shaik Mohammad Fareed 9491202987

(g). String Lowercase Function: The **"strtolower()"** function returns string in lowercase letter.

```
<html>
<body>
<?php
        $str="This is Shaik Mohammad Fareed";
        $str=strtolower($str);
                echo $str;  ?>
</body>
</html>
```

Output: this is shaik mohammad fareed

(h). String Uppercase Function: The **"strtoupper()"** function returns string in uppercase letter.

```
<html>
<body>
<?php
        $str="This is Shaik Mohammad Fareed";
        $str=strtoupper($str);
        echo $str;
?>
</body>
</html>
```

Output: THIS IS SHAIK MOHAMMAD FAREED

(i). String First Character into Uppercase Function: The **"ucfirst()"** function returns string converting first character into uppercase. It doesn't change the case of other characters.

```
<html>
<body>
<?php
        $str="shaik mohammad fareed";
        $str=ucfirst($str);
        echo $str;
?>
</body>
</html>
```

Output: Shaik mohammad fareed

(j). String First Character into Lowercase Function: The **"lcfirst()"** function returns string converting first character into lowercase. It doesn't change the case of other characters.

```
<html>
<body>
<?php
        $str="SHAIK MOHAMMAD FAREED";
        $str=lcfirst($str);
        echo $str;
?>
</body>
</html>
```

Output: shaik MOHAMMAD FAREED

(k). String First Character of a Word into Uppercase Function: The **"ucwords()"** function returns string converting first character of each word into uppercase.

```
<html>
<body>
<?php
        $str="shaik mohammad fareed";
        $str=ucwords($str);
        echo $str;
?>
</body>
</html>
```

Output: Shaik Mohammad Fareed

(Q) Explain the Formatting Strings with PHP?

The following String Formats in PHP program

- print() → Output a String.
- sprintf() → Return a formatted string.
- vprintf() → Output a formatted string.
- sscanf() → Parse Input form a string according to a format.
- fscanf() → Parse Input form a file according to a format.
- flush() → Flush system output buffer.

VIMP***** (Q) Explain the Date and Time Functions in PHP?**

The PHP date() function convert a timestamp to a more readable data and time. The computer stores dates and times in a format called UNIX Timestamp, which measures time as a number of seconds since the beginning of the UNIX (January 1, 1970 00:00:00 GMT Greenwich Mean Time). The syntax of the PHP data() function can be use.

Syntax: date(format, timestamp);

- The "format" parameter in the date() function is required which specifies the format of returned date and time.
- The "timestamp" is an optional parameter. If not included then current date and time will be used.

(I). Formatting the Date in PHP:

Get a Simple Date: The required *format* parameter of the date() function specifies how to format the date (or time). Here are some characters that are commonly used for dates:

- d → Represents the day of the month (01 to 31).
- D → Represents the day of the week in text as an abbreviation (Mon to Sun).
- m → Represents a month (01 to 12).
- M → Represents a month in text as an abbreviation (Jan to Dec)
- y → Represents a year in two digits (08 or 17)
- Y - Represents a year (in four digits).
- l (lowercase 'L') - Represents the day of the week.

Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

Example:

```
<html>
<body>
<?php
        echo "Today is " . date("Y/m/d") . "<br>";
        echo "Today is " . date("Y.m.d") . "<br>";
        echo "Today is " . date("Y-m-d") . "<br>";
        echo "Today is " . date("l");
?>
</body>
</html>
```

Output:

```
Today is 2025/07/14
Today is 2025.07.14
Today is 2025-07-14
Today is Monday
```

(II). Formatting the Time in PHP: The **"mktime()"** function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix (January 1 1970 00:00:00 GMT) and the time specified.

Get a Simple Time: Here are some characters that are commonly used for times:

- h → 12-hour format of an hour with leading zeros (01 to 12).
- H → 24-hour format of an hour with leading zeros (00 – 23)
- i → Minutes with leading zeros (00 to 59).
- s → Seconds with leading zeros (00 to 59).
- a → Lowercase Ante Meridiem and Post Meridiem (am or pm).
- A → Uppercase Ante Meridiem and Post Meridiem (AM or PM).

Example:

```
<html>
<body>
< ?php
        echo "The Time is " . date("h:i:s:a");
?>
```

```
        </body>
        </html>
```

Output: The Time is 03:20:45 am

Create a Date From a String in PHP: The PHP **"strtotime()"** function is used to convert a human readable string to a Unix time.

```
        <html>
        <body>
        <?php
                $d=strtotime("10:30 pm April 08 2025");
                echo "Resulted date is " . date("Y-m-d h:i:s:a", $d);
        ?>
        </body>
        </html>
```

Output:

```
Resulted date is 2025-04-08 10:30:00 pm
```

Example: Display the Tomorrow and next Saturday and after 3 month of day?

```
        <html>
        <body>
        <?php
                $d=strtotime("Tomorrow");
                echo date("Y-m-d h:i:sa", $d) . "<br>";
                $d=strtotime("Next Saturday");
                echo date("Y-m-d h:i:sa", $d) . "<br>";
                $d=strtotime("+3 Months");
                echo date("Y-m-d h:i:sa", $d) . "<br>";
        ?>
        </body>
        </html>
```

Output:

```
2025-07-15 12:00:00 am      → Tomorrow Date (Tuesday) but Today is Monday 2025-07-14
2025-07-19 12:00:00 am      → Next Saturday (Saturday)
2025-10-14 12:00:00 pm      → After 3 Month of day (Tuesday)
```

Example: Display Next Six Saturdays in the month?

```
        <html>
        <body>
        <?php
                $startdate=strtotime("Saturday");
                $enddate=strtotime("+6 weeks", $startdate);
                        while ($startdate < $enddate) {
                                 echo date("M d", $startdate) . "<br>";
                                $startdate = strtotime("+1 week", $startdate);
                        }
        ?>
        </body>
        </html>
```

Output:

```
July 19      July 26      August 02      August 09      August 16      August 23.
```

UNIT-IV
Chapter 1: Working with Forms

(Q) What is Form? Explain Form objects with a one example?

Forms are a mechanism that allow you to type information into fields on a web browser screen, and then submit data to a web server. This type of pages is called ad "interactive web pages".

Forms have many functions they can be used for gathering information, submitting information in online and providing values in different controls. A form will be created by using <form> tag; this tag allows different controls such as 'textbox, checkbox, list box, submit button, password, etc'.

The Form <form>tag has mainly three attributes:

- Name
- Action
- Method

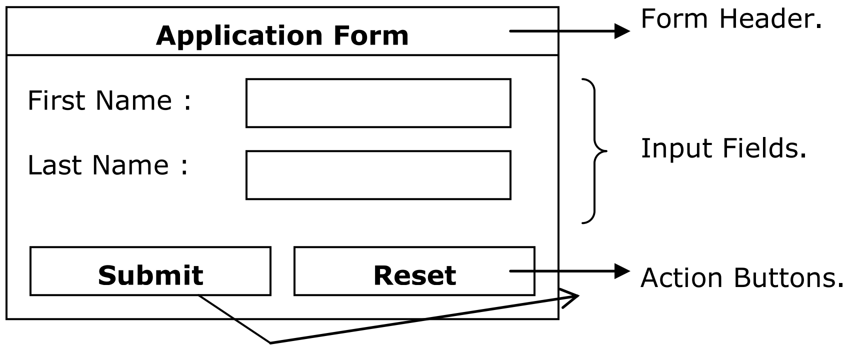
1. **Name Attribute:** Specify the name of the form.
2. **Action Attribute:** Program location of CGI (Common Gateway Interface).
3. **Method Attribute:** To perform the action means submitting the input values from a form to the web browser. The method accepts two values that are "get" & "post".

Syntax for Form Attributes:

<form name = "Form Name" action = "URL (uniform Resource Locator) / Address" method= "get / post">

Example: <form name = "Student Details" action = "http:\\Fareed\Student data.html" method ="get">

Creating Form: Forms can be classified into 3 sections namely "Form Header", "Input Fields" & "Action Buttons".



Form Inputs / Input Elements on Form: Input elements or form elements that can accept the input values from the user. The following are some input elements that can be added to a form.

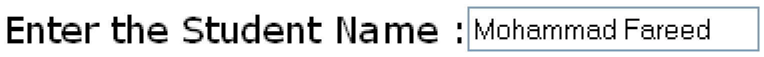
- Text Box.
- Text Area.
- List Box / Select / Pull Down Option.
- Check Box.
- Radio Buttons.
- Command Buttons.
- Password Textbox.
- Submit & Reset Button / Action Buttons. etc.

1. Text Box: A Text Box is a rectangular shaped field in which a user can enter the text. A Text Box is created by using <input> tag in HTML.

Attributes of Text Box: There are following attributes can support the Text Box.

- **Type:** This attribute specifies the type of the form elements will be added to the form like: textbox, checkbox, submit button etc.
- **Name:** It specifies the name of the text box in the form.
- **Size:** It specifies the textbox size.
- **Align:** It specifies the alignments of the text in the text box.
- **Value:** It is specifies the default values.
- **Maxlength:**It specifies the maximum number of characters can allowed in the text box.

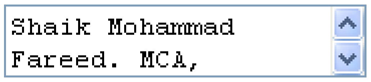
Example: Enter the Student Name :<input type = "text" name = "Students Details" size = "30" align = "center" Maxlength = "25" value = "Mohammad Fareed">.


Enter the Student Name : Mohammad Fareed

2. Text Area: Which specifies an input box that can contain multiple line of text. The Text Area is created by using <textarea> tag in HTML.

Attributes of Text Area: There are following attributes can support the Text Area.

- **Rows:** It specifies number of rows can be defined.
- **Cols:** It specifies number of columns can be defined.


Shaik Mohammad
Fareed. MCA,

Example: In order to define a text area which can accept five rows of 50 characters. The code can be

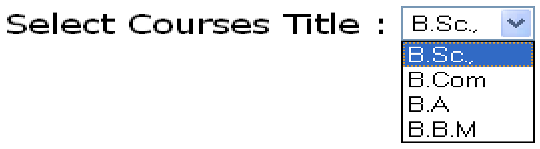
<textarea rows = "5" cols = "50">Shaik Mohammad Fareed. MCA, IRPM., Department of Computer Science & Applications </textarea>

3. List Box / Selection: A List Box displays multiple values, for to create a list box <select> tag using. To specify each list items in a list box <option> tag are used. When ever created the list of items the pull down button will appear. The list box appears as a list with vertical scrollbars, which the user can scroll to display the entire list of items.

Attributes of List Box: There are following attributes can support the Text Area.

- **Name:** It specifies the name of the Selection List.
- **Size:** It specifies the number data items can be inserted.

Example:
```
<select>
        <option> B.Sc., </option>
        <option> B.Com </option>
        <option> B.A </option>
        <option> B.B.M </option>
    </select>
```

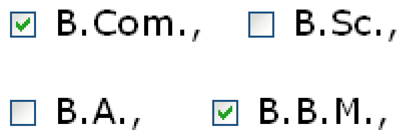

Select Courses Title : B.Sc.,
B.Sc.,
B.Com
B.A
B.B.M

4. Check Box: The Check box can allow only two values. Therefore checked and unchecked. Here user can select more one data items by using checkbox. For create a checkbox <input> tag are used.

Attributes of Check box: There are following attributes can support the check box.

- **Type:** Here the type must be checkbox, indicates that checkbox is the interface elements.
- **Name:** Name of the Checkbox by which it will be identified.
- **Value:** The value is reading will be assigned to the checkbox when it is unchecked state.

Example: <input type = "checkbox"> B.Com., &nbsp; <input type = "checkbox"> B.Sc., &nbsp; <br> <input type = "checkbox"> B.A., &nbsp; &nbsp; <input type = "checkbox"> B.B.M., &nbsp;

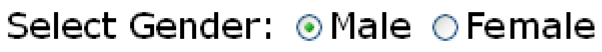


5. Radio Button: The Radio buttons can allow only two values. Therefore selected and deselected. Here user can select only one data items at a time by using selected. For create a radio buttons <input> tag are used.

Attributes of Radio Button: There are following attributes can support the Radio button.

- **Type:** Radio the type value is radio indicates that radio is the interface elements.
- **Name:** Name of the Radio group by which it will be identified.
- **Value:** The value is reading will be assigned to the select when it is deselecting state.

Example: Select Gender: <Input Type="Radio" Name="Gender">Male &nbsp;
<Input Type="Radio" Name="Gender">Female.

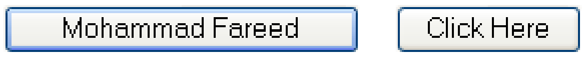


6. Command Button: The Command Button is created by using <input> tag. To create a button specifies the button as to the type attribute in <input> tag.

Attributes of Command Button: There are following attributes can support the Command button.

- **Type:** Command the type value is ONCLICK indicates button that is the interface elements.
- **Value:** To display the text on the command button.

Example: <input type ="button" value = "Mohammad Fareed"> &nbsp;
<input type ="button" value = "Click Here">

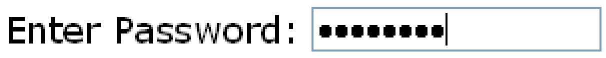


7. Password Textbox: Passwords is same as a input textbox where the character are displayed in a from of bullet or asterisk symbols. To create password field, specified the password as to the type attribute in <input> tag.

Attributes of Password Textbox: There are following attributes can support the Password.

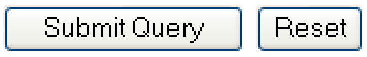
- **Type:** Password the type value.
- **Name:** Name of the Password field.
- **Maxlength:** Size of the Password.

Example: Enter Password: <Input Type="Password" Name="TPassword" maxlength ="8">



8. Submit & Reset Buttons / Action Button: The Submit & Reset buttons are action buttons, the Submit Button send input data value which is enter in the HTML Form. The Reset button clears the all fields in the form. To create a Submit Button used in <input> tag and type is "submit". For create a Reset Button used in <input> tag and type is "reset".

Example: <input type = "submit"> <input type = "reset">



IMP***** (Q) Explain how to Create a Form in PHP?**

Form: A Document that containing block of fields, that the user can fill the data or user can select the data. Casually the data will store in the database.

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST. The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form: Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

PHP Post Form: Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Example: Illustrate the post() method in form.

```
<html>
<body>
<form action="welcome.php" method="post">
 Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

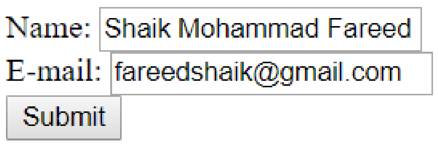
Save: welcome.php

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

Output:



Name: Shaik Mohammad Fareed
E-mail: fareedshaik@gmail.com

Welcome Shaik Mohammad Fareed
Your email address is: fareedshaik@gmail.com

Example: Illustrate the get() method in form.

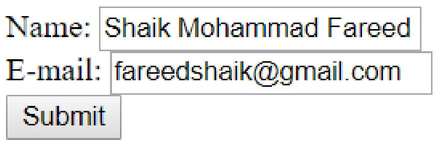
The same result could also be achieved using the HTTP GET method:

```
<html>
<body>
<form action="welcome_get.php" method="get">
        Name: <input type="text" name="name"><br>
        E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

Save: welcome_get.php

```
<html>
<body>
        Welcome <?php echo $_GET["name"]; ?><br>
        Your email address is: <?php echo $_GET["email"]; ?>
</body>
</html>
```

Output:



Name: Shaik Mohammad Fareed
E-mail: fareedshaik@gmail.com

```
Welcome Shaik Mohammad Fareed
Your email address is: fareedshaik@gmail.com
```

When to use POST? - Information sent from a form with the POST method is **Invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send. Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server. However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

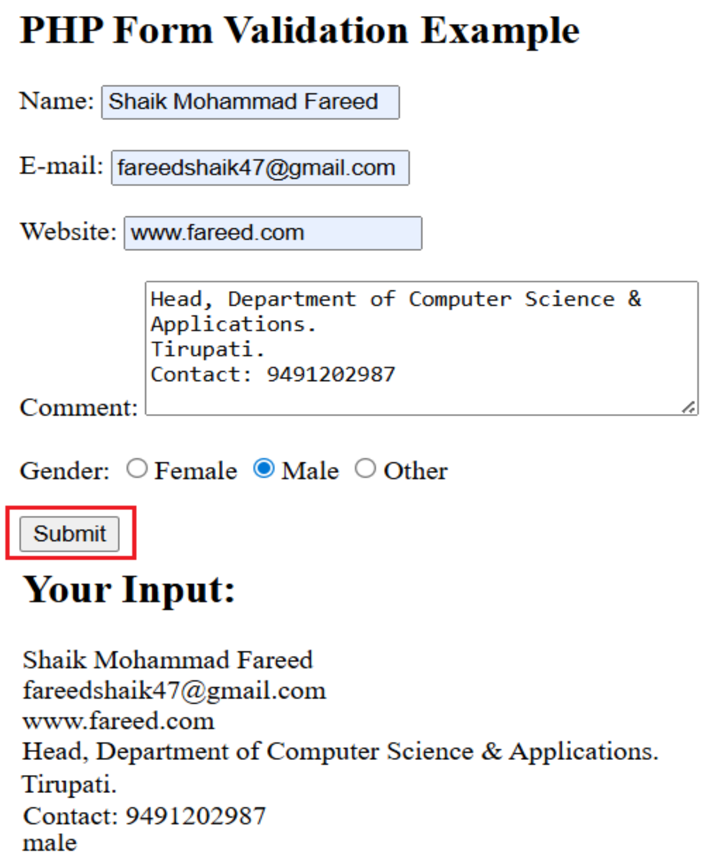
When to use GET? - Information sent from a form with the GET method is **Visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

Example: Illustrate the Form Validation using Get & Post Methods in PHP?

```
<html>
<head>
</head>
<body>

<?php
    $name = $email = $gender = $comment = $website = ""; // define variables and set to empty values
       if ($_SERVER["REQUEST_METHOD"] == "POST")
```

```
        {
                $name = test_input($_POST["name"]);
                $email = test_input($_POST["email"]);
                $website = test_input($_POST["website"]);
                $comment = test_input($_POST["comment"]);
                $gender = test_input($_POST["gender"]);
        }

function test_input($data)
        {
                $data = trim($data);
                $data = stripslashes($data);
                $data = htmlspecialchars($data);
                return $data;
        }
?>
<h2>PHP Form Validation Example</h2>
        <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
                Name: <input type="text" name="name">  <br><br>
                E-mail: <input type="text" name="email">  <br><br>
                Website: <input type="text" name="website">  <br><br>
                Comment: <textarea name="comment" rows="5" cols="40"></textarea> <br><br>
                Gender:  <input type="radio" name="gender" value="female">Female
                         <input type="radio" name="gender" value="male">Male
                         <input type="radio" name="gender" value="other">Other  <br><br>
                         <input type="submit" name="submit" value="Submit">
        </form>
<?php
        echo "<h2>Your Input:</h2>";
        echo $name;
        echo "<br>";
        echo $email;
        echo "<br>";
        echo $website;
        echo "<br>";
        echo $comment;
        echo "<br>";
        echo $gender;
?>
</body>
</html>
```

Output:



PHP Form Validation Example

Name: Shaik Mohammad Fareed

E-mail: fareedshaik47@gmail.com

Website: www.fareed.com

Comment:
```
Head, Department of Computer Science &
Applications.
Tirupati.
Contact: 9491202987
```

Gender: ( ) Female [✓ Male] ( ) Other

Submit

Your Input:

Shaik Mohammad Fareed
fareedshaik47@gmail.com
www.fareed.com
Head, Department of Computer Science & Applications.
Tirupati.
Contact: 9491202987
male

(Q) Explain how to access form Input with GET and POST methods Form Arrays in PHP?

Access from Input with GET() & POST(): Both POST and GET methods are creates an arrays. The arrays have 'key' and 'value'. Where 'key' is the name of the form control and the 'value' are the input data from the user.

Syntax: array (Key => Value, Key => Value, Key => Value,…);

Both POST and GET are treated as $_POST and $_GET. These are super global, which means that they are always accessible, regardless of scope, and you can access then from any functions, class or file without having to do anything special operations.

$_POST is an array of variables passed to the current script via the HTTP POST method.
$_GET is any array of variables passed to the current script via the URL parameter.

The other way to use a PHP array is to think of the subscript as a key that can be used to retrieve data, similar to Perl's hash. In this case, the "subscript" will be a string. As with integer subscripts, there are two ways of assigning values to an array. The code fragments below are equivalent.

```
$myname = array('FirstName' => 'Shaik', 'MiddleName' => 'Mohammad', 'LastName’ => 'Fareed');
```

Arrays created in this way are manipulated using strings for subscripts.

```
$myname[‘FirstName’] = ‘Shaik’;
$myname[‘MiddleName’] = ‘Mohammad’;
$myname[‘LastName’] = ‘Fareed’;
```

The statement **echo $myname['LastName'];** will print "**Fareed**".

The *foreach* construct can be used to extract both subscripts and values.

```
foreach($myname as $smf =>mca) {
 echo ‘My Name is : ’ $smf, $mca, ‘<br/>’; }
```

Output:
```
My Name is : FirstName Shaik
My Name is : MiddleName Mohammad
My Name is : LastName Fareed
```

(Q) Define how to combining HTML and PHP code on a single page?

In some circumstances, you may want to include form-parsing code on the same page as a hard-coded HTML form. The PHP is designed to interact with HTML and PHP scripts can be included in an HTML page without a problem. Anything in a PHP script that is not contained within <?php ?> tags is ignored by the PHP compiler and passed directly to the web browser

Example:<h3>Hai This is Shaik Mohammad Fareed. <? Php echo "Today is:”. date("Y/m/d") . "<br>";> </h3>

Output:
```
Hai This is Shaik Mohammad Fareed.   Today is : 2025/07/14.
```

(Q) Explain about Handling Files in PHP?

In PHP, File handling is the process of interacting with files on the server, such as reading files, writing to a file, creating new files, or deleting existing ones. File handling is essential for applications that require the storage and retrieval of data, such as logging systems, user-generated content, or file uploads.

Types of File Operations in PHP: Several types of file operations can be performed in PHP:

- **Reading Files:** PHP allows you to read data from files either entirely or line by line.
- **Writing to Files**: You can write data to a file, either overwriting existing content or appending to the end.
- **File Metadata**: PHP allows you to gather information about files, such as their size, type, and last modified time.
- **File Uploading**: PHP can handle file uploads via forms, enabling users to submit files to the server.

Common File Handling Functions in PHP:

- **fopen():** Opens a file
- **fclose():** Closes a file
- **fread():** Reads data from a file
- **fwrite():** Writes data to a file
- **file_exists():** Checks if a file exists
- **unlink():** Deletes a file

(1). Opening and Closing Files: Before you can read or write to a file, you need to open it using the fopen() function, which returns a file pointer resource. Once you're done working with the file, you should close it using fclose() to free up resources.

Example:
```
<?php
        $file = fopen("FareedNotes.txt", "r");       // Open the file in read mode
        if ($file)
          {
                echo "File opened successfully!";
                fclose($file);                        // Close the file
           }
            else
                {
                        echo "Failed to open the file.";
                }
?>
```

File Modes in PHP: Files can be opened in any of the following modes:

- **"w":** Opens a file for writing only. If the file does not exist, then a new file is created, and if the file already exists, then the file will be truncated (the contents of the file are erased).
- **"r":** File is open for reading only.
- "a": File is open for writing only. The file pointer points to the end of the file. Existing data in the file is preserved.
- **"w+":** Opens file for reading and writing both. If the file does not exist, then a new file is created, and if the file already exists, then the contents of the file are erased.

- **"r+":** File is open for reading and writing both.
- **"a+":** File is open for write/read. The file pointer points to the end of the file. Existing data in the file is preserved. If the file is not there, then a new file is created.
- **"x":** New file is created for write only.

(2). Reading from Files: There are two ways to read the contents of a file in PHP. These are:

2.1. Reading the Entire File: You can read the entire content of a file using the fread() function or the file_get_contents() function.

Example:

```
<?php
        $file = fopen("FareedNotes.txt", "r");
        $content = fread($file, filesize("FareedNotes.txt"));
        echo $content;
        fclose($file);
?>
```

2.2. Reading a File Line by Line: You can use the fgets() function to read a file line by line.

Example:

```
<?php
        $file = fopen("FareedNotes.txt", "r");
        if ($file)
           {
                while (($line = fgets($file)) !== false)
                {
                        echo $line . "<br>";
                }
                fclose($file);
          }
?>
```

(3). Writing to Files: You can write to files using the fwrite() function. It writes data to an open file in the specified mode.

Example:

```
<?php
        $file = fopen("FareedNotes.txt", 'w');        // Open the file in write mode
        if ($file)
          {
                $text = "Hello, This is Shaik Mohammad Fareed. MCA, M.Sc, IRPM.\n";
                fwrite($file, $text);
                fclose($file);
          }
?>
```

(4). Deleting Files: Use the unlink() function to delete the file in PHP.

Example:
```
        <?php
                if (file_exists("FareedNotes.txt"))
                  {
                        unlink("FareedNotes.txt");
                        echo "File deleted successfully!";
                  }
                else
                  {
                        echo "File does not exist.";
                 }
        ?>
```

(Q) Explain how to Upload a File in PHP?

PHP File Upload: With PHP, it is easy to upload files to the server. However, with ease comes danger, so always be careful when allowing file uploads!

Create The HTML Form: To create an HTML form that allow users to choose the image file they want to upload:

```
<html>
<body>
        <form action="upload.php" method="post" enctype="multipart/form-data">
                Select image to upload:
                <input type="file" name="fileToUpload" id="fileToUpload">
                <input type="submit" value="Upload Image" name="submit">
        </form>
</body>
</html>
```

Some rules to follow for the HTML form above:

- Make sure that the form uses method="post"
- The form also needs the following attribute:enctype="multipart/form-data". It specifies which content-type to use when submitting the form
- Without the requirements above, the file upload will not work.

NOTE: Other things to notice:

- The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control.
- The form above sends data to a file called "upload.php", which we will create next.

Create The Upload File PHP Script: The "upload.php" file contains the code for uploading a file:

```
<?php
        $target_dir = "uploads/";
        $target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
        $uploadOk = 1;
        $imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
        // Check if image file is a actual image or fake image
                if(isset($_POST["submit"]))
                 {
                        $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
                        if($check !== false)
                         {
                                echo "File is an image - " . $check["mime"] . ".";
                                $uploadOk = 1;
                         }
                         else
                                {
                                        echo "File is not an image.";
                                        $uploadOk = 0;
                                }
                }
?>
```

Explanation:

- $target_dir = "uploads/" - specifies the directory where the file is going to be placed
- $target_file specifies the path of the file to be uploaded
- $uploadOk=1 is not used yet (will be used later)
- $imageFileType holds the file extension of the file (in lower case)
- Next, check if the image file is an actual image or a fake image

Note: You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

Check if File Already Exists: First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:

```
        if (file_exists($target_file))          // Check if file already exists
        {
                echo "Sorry, file already exists.";
                $uploadOk = 0;
        }
```

Limit File Size: The file input field in our HTML form above is named "fileToUpload". Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:

```
        if ($_FILES["fileToUpload"]["size"] > 500000)      // Check file size
          {
                echo "Sorry, your file is too large.";
                $uploadOk = 0;
        }
```

Limit File Type: The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:

```
// Allow certain file formats
        if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
                && $imageFileType != "gif" )
         {
                echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
                $uploadOk = 0;
        }
```

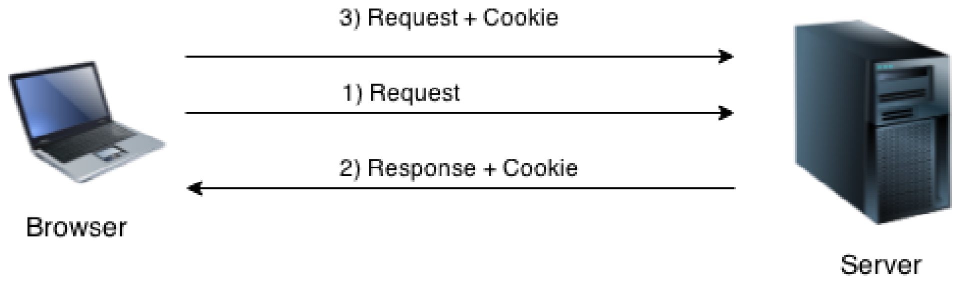
Complete Upload File PHP Script: The complete "upload.php" file now looks like this:

```
<?php
        $target_dir = "uploads/";
        $target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
        $uploadOk = 1;
        $imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

        if(isset($_POST["submit"]))             // Check if image file is a actual image or fake image
          {
                $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
                if($check !== false)
                  {
                        echo "File is an image - " . $check["mime"] . ".";
                        $uploadOk = 1;
                  }
                else
                    {
                        echo "File is not an image.";
                        $uploadOk = 0;
                    }
           }

        if (file_exists($target_file))                          // Check if file already exists
           {
                echo "Sorry, file already exists.";
                $uploadOk = 0;
           }

        if ($_FILES["fileToUpload"]["size"] > 500000)          // Check file size
        {
                echo "Sorry, your file is too large.";
                $uploadOk = 0;
        }
                // Allow certain file formats
        if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
                        && $imageFileType != "gif" )
        {
                echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
                $uploadOk = 0;
        }
        if ($uploadOk == 0)                     // Check if $uploadOk is set to 0 by an error
        {
                echo "Sorry, your file was not uploaded.";
        }
        else                                    // if everything is ok, try to upload file
        {
                if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file))
                {
                        echo "The file ". htmlspecialchars( basename( $_FILES["fileToUpload"]["name"])).
                          "has been uploaded.";
                }
                else
                  {
                        echo "Sorry, there was an error uploading your file.";
                  }
        }
?>
```

Chapter 2: Session Function Overview

(Q) What is Cookies? Explain how to working a cookies using PHP?

Cookies: PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user. Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



- A Cookie contains information that is stored by a web server on a web site visitor's computer.
- Cookies are often used to store information about which pages a user has viewed on a web site, how many times a users has visited a site, and what information a user has entered during past visits.
- A Single Cookie can be store a maximum of 20 names / values and its allows maximum of 4096 characters.
- PHP transparently supports HTTP cookies.
- The Cookies are classified into two types:

    1. Temporary Cookies.
    2. Persistent Cookies.

1. Temporary Cookies: The Temporary Cookies stores information in the main memory of the work station and its life time will be until the browser session is terminated. That means once the browser window is closed automatically the cookies information will be erased.

2. Persistent Cookies: Where as Persistent Cookies store information in text files on the user workstation. These cookies have an expiration data, that information are deleted by the system after a specified time intervals. To create persistent cookies the syntax follows same as temporary cookies, but in addition to that you must specify expiry date.

Note: PHP Cookie must be used before <html> tag.

(I). The Anatomy of a Cookie: Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser). A PHP script that sets a cookie might send headers that look something like this.

- HTTP/1.1 200 OK
- Date: Wed, 14 March 2018 23:39:50 GMT
- Server: Apache/1.3.9 (UNIX) PHP/4.0b3
- Set-Cookie: name=fareed; expires=Sunday, 18-Mar-2018 23:30:18 GMT;
- path=/; domain=E:\Fareed PHPprog\cookies1.html
- Connection: close
- Content-Type: text/html

As you can see, the Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded.

A PHP script will then have access to the cookie in the environmental variables $_COOKIE or $HTTP_COOKIE_VARS[] which holds all cookie names and values. Above cookie can be accessed using $HTTP_COOKIE_VARS["name"].

(II). Setting Cookies with PHP: PHP provided **setcookie()** function to set a cookie. This function requires upto six arguments and should be called before <html> tag. For each cookie this function has to be called separately.

Syntax: setcookie(name, value, expire, path, domain, security);

- **Name**: This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.

- **Value**: This sets the value of the named variable and is the content that you actually want to store.

- **Expiry**: This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.

- **Path**: This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.

- **Domain**: This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.

- **Security**: This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

PHP setcookie() Function: PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

- setcookie("CookieName", "CookieValue");/* defining name and value only*/
- setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 hour(1*60*60 second or 3600 seconds)

Example: illustrate the setcookie() function?

```
<?php
        setcookie("UserName", "Fareed");
?>
<html>
<body>
        <?php
                if(!isset($_COOKIE["user"])) {
                echo "Sorry, cookie is not found!";
                } else {
                echo "<br/>Cookie Value: " . $_COOKIE["user"]; }
        ?>
</body>   </html>
```

Output: Sorry, cookie is not found!

NOTE: Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Output: Cookie Value: Fareed.

VIMP**************(Q) Define how to working with Session in WebPages?**

Sessions: A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user's computer.

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, passwords, favorite color, etc). By default, session variables last until the user closes the browser.

NOTE: Session variables hold information about one single user, and are available to all pages in one application.

How Do PHP Sessions Work?

- **Session Start:** When a user accesses a PHP page, the session gets started with the session_start() function. This function initiates the session and makes the session data available through the $_SESSION superglobal array.

- **Session Variables**: Data that needs to be carried across different pages is stored in the $_SESSION array. For example, a user's name or login status can be stored in this array.

- **Session ID:** PHP assigns a unique session ID to every user. This session ID is stored in a cookie in the user's browser by default. The session ID is used to retrieve the user-specific data on each page load.

- **Session Data Storage:** The session data is stored on the server, not the client side. By default, PHP stores session data in a temporary file on the server. The location of this storage is determined by the session.save_path directive in the php.ini file.

- **Session Termination**: Sessions can be terminated by calling session_destroy(), which deletes the session data. Alternatively, a session can be closed using session_write_close() to save the session data and free up server resources.

(I). Start a PHP Session: A session is started with the session_start() function. Session variables are set with the PHP global variable: $_SESSION.

Example: Now, let's create a new page called "fareed_session1.php". In this page, we start a new PHP session and set some session variables:

```
<?php
        session_start();  // Start the session
?>
<html>
<body>
<?php
        $_SESSION["myname"] = "Shaik Mohammad Fareed";            // Set session variables
        $_SESSION["myplace"] = "Tirupati";
        echo "Session variables are set.";
 ?>
</body>
</html>
```

Save: fareed_session1.php

Output: Session variables are set.

NOTE: The session_start() function must be the very first thing in your document. Before any HTML tags.

(II). Get PHP Session Variable Values: Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("fareed_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

NOTE: Also notice that all session variable values are stored in the global $_SESSION variable:

```
<?php
        session_start();
?>
```

```
<html>
<body>
<?php
        // Echo session variables that were set on previous page
                echo "My Name is: " . $_SESSION["myname"] . ".<br>";
                echo "My Native Place is: " . $_SESSION["myplace"] . ".";
?>
</body>
</html>
```

Output:
```
My Name is: Shaik Mohammad Fareed
My Native Place is: Tirupati
```

(III). Modify a PHP Session Variable: To change a session variable, just overwrite it:

Example:

```
<?php
        session_start();
?>
<html>
<body>
<?php
        // to change a session variable, just overwrite it
                $_SESSION["myname"] = "Shaik Abu Hanifa";
                print_r($_SESSION);
?>
</body>
</html>
```

Output: Array ([myname] => Shaik Abu Hanifa [myplace] => Tirupati)

(IV). Destroy a PHP Session: To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example:

```
<?php
        session_start();
?>
<html>
<body>
<?php
        session_unset();       // remove all session variables
        session_destroy();     // destroy the session
        echo "All session variables are now removed, and the session is destroyed."
?>
</body>
</html>
```

Output: All session variables are now removed, and the session is destroyed.

UNIT – V: Interacting with MySQL using PHP

Introduction of MySQL:

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is a fast, easy to use relational database. It is currently the most popular open-source database. It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications. MySQL is used for many small and big businesses. It is developed, marketed and supported by MySQL AB, a Swedish company. It is written in C and C++.

What is MySQL:

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

MySQL Database: MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons

What is a Database: A database is a separate application that stores a collection of data. Each database has one or more distinct APIs (Applications Programming Interfaces) for creating, accessing, managing, searching and replicating the data it holds. Nowadays, we use Relational Database Management Systems (RDBMS) to store and manage huge volume of data.

A Relational Database Management System (RDBMS): The RDBMS is a software that

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology: Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.

- **Database**: A database is a collection of tables, with related data.
- **Table**: A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column**: One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row**: A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.

- **Redundancy**: Storing data twice, redundantly to make the system faster.
- **Primary Key**: A primary key is unique. A key value cannot occur twice in one table. With a key, you can only find one row.
- **Foreign Key**: A foreign key is the linking pin between two tables.
- **Compound Key**: A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index**: An index in a database resembles an index at the back of a book.
- **Referential Integrity**: Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Features:

- **Relational Database Management System (RDBMS):** MySQL is a relational database management system.

- **Easy to use:** MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.

- **It is secure:** MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.

- **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.

- **Free to download:** MySQL is free to use and you can download it from MySQL official website.

- **It is scalable:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.

- **Compatible on many operating systems:** MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

- **Allows roll-back:** MySQL allows transactions to be rolled back, commit and crash recovery.

- **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.

- **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.

- **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

Disadvantages / Drawback of MySQL: Following are the few disadvantages of MySQL:

- MySQL version less than 5.0 doesn't support ROLE, COMMIT and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

(Q) Define various Data type in MySQL?

MySQL Data Types: A Data Type specifies a particular type of data, like integer, floating points, Boolean etc. It also identifies the possible values for that type, the operations that can be performed on that type and the way the values of that type are stored. MySQL supports a lot number of SQL standard data types in various categories. It uses many different data types broken into mainly three categories: numeric, date and time, and string types.

(I). Numeric Data Type:

Data Type Syntax	**Description**
INT	A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
TINYINT	A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
SMALLINT	A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
MEDIUMINT	A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
BIGINT	A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
FLOAT(m,d)	A floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a float.
DOUBLE(m,d)	A double precision floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a double. Real is a synonym for double.
DECIMAL(m,d)	An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (m) and the number of decimals (d) is required. Numeric is a synonym for decimal.

(II). Date and Time Data Type:

Data Type Syntax	Maximum Size	Explanation
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'yyyy-mm-dd'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'yyyy-mm-dd hh:mm:ss'.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'.
YEAR[(2|4)]	Year value as 2 digits or 4 digits.	Default is 4 digits.

(III). String Data Types:

Data Type Syntax	Maximum Size	Explanation
CHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
VARCHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string.
TINYTEXT(size)	Maximum size of 255 characters.	Where size is the number of characters to store.
TEXT(size)	Maximum size of 65,535 characters.	Where size is the number of characters to store.
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters.	Where size is the number of characters to store.
LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters.	Where size is the number of characters to store.
BINARY(size)	Maximum size of 255 characters.	Where size is the number of binary characters to store. Fixed-length strings. Space padded on right to equal size characters. (introduced in MySQL 4.1.2)
VARBINARY(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string. (introduced in MySQL 4.1.2)

(IV).Large Object Data Types (LOB) Data Types:

Data Type Syntax	Maximum Size
TINYBLOB	Maximum size of 255 bytes.
BLOB(size)	Maximum size of 65,535 bytes.
MEDIUMBLOB	Maximum size of 16,777,215 bytes.
LONGTEXT	Maximum size of 4gb or 4,294,967,295 characters.

(Q) Define difference between MySQL & MySQLi (Improved)?

There are too many differences between these PHP database extensions. These differences are based on some factors like performance, library functions, features, benefits, and others.

MySQL	MySQLi(Improved)
MySQL extension added in PHP version 2.0. and deprecated as of PHP 5.5.0.	MySQLi extension added in PHP 5.5 and will work on MySQL 4.1.3 or above.
Does not support prepared statements.	MySQLi supports prepared statements.
MySQL provides the procedural interface.	MySQLi provides both procedural and object-oriented interface.
MySQL extension does not support stored procedure.	MySQLi supports store procedure.
MySQL extension lags in security and other special features, comparatively.	MySQLi extension is with enhanced security and improved debugging.
Transactions are handled by SQL queries only.	MySQLi supports transactions through API.
Extension directory: ext/mysql.	Extension directory: ext/mysqli.

(Q) Explain various steps to Connect to MySQL and Interfacing with Database through PHP?
(or)
How to Connect to MySQL with PHP and Work with MySQL Data?

MySQL an open source Relational Database Management Systems (RDBMS) based on Structured Query Language (SQL).

You through establishing a MySQL connection using PHP on a web page, connecting to a MySQL table and retrieving the results and displaying them back on the web page.

Steps for Interfacing:

(I). MySQL_Connect.
(II). MySQL_Query.
(III). MySQL_Fetch_Array.
(IV). MySQL_Close.

(I). Connection to a MySQL Database: You need connection to MySQL Server address (if the database is on the same server as the web server it will most likely be localhost or 127.0.01 (IP Address)), username, password and database name. Create a file and save with 'filename.php' and open the PHP code with tags before the HTML, you can put regular HTML tag after it.

```
<?php
  $db = mysql_connect('localhost', 'username', 'password', 'database_name')
      or die('Error connecting to MySQL server');
?>
<html>
  <head>  </head>
  <body   </body>
</html>
```

NOTE: The variable '$db' is created and assigned as the connection string, it will be used in next step. If there is a failure then an error message will be displayed on the page. If it is successful you will see PHP connect to MySQL message.

(II). Open a Connection to MySQL: Before we can access data in the MySQL database, we need to be able to connect to the server:

```
<?php
        $servername = “localhost”;
        $username = “username”;
        $password = “password”;
        $conn = new mysql($servername, $username, $password);      // Create Connection
        if($conn ->connect_error)                                     // Check Connection
          {
             die(“Connection Failed :”.$conn->connect_error);
          }
        echo “Connected Successfully”;
?>
```

(III). Close the Connection: The connection will be closed automatically when the script ends. To close the connection before, use the following

```
        $conn -> close();   (or)    mysql_close($conn);
        $conn = null;
```

(Q) Discuss the Create Database in MySQL?

The following code will be created database in MySQL?

```
<html>
  <head>
    <title>Creating MySQL Database</title>
  </head>
  <body>
    <?php
      $dbhost = 'localhost:3036';
      $dbuser = 'mohammadfareed';
      $dbpass = 'fareed';
      $conn = mysql_connect($dbhost, $dbuser, $dbpass);
      if(! $conn )
      {
        die('Could not connect: ' . mysql_error());
      }
      echo 'Connected successfully<br />';

      $sql = 'CREATE DATABASE COLLEGE';
      $retval = mysql_query( $sql, $conn );
      if(! $retval ) {
        die('Could not create database: ' . mysql_error());
      }
      echo "Database COLLEGE created successfully\n";
      mysql_close($conn);
    ?>
  </body>
</html>
```

(Q) Explain various commands in MySQL?

(I). Create Table Command: To begin with, the table creation command requires the following details.

- Name of the table
- Name of the fields
- Definitions for each field

Syntax: **mysql>** CREATE TABLE table_name (column_name column_type);

Now, we will create the following table in the COLLEGE database.

```
mysql> create table employee( emp_id INT NOT NULL AUTO_INCREMENT, emp_name VARCHAR(40) NOT
          NULL, address VARCHAR(100) NOT NULL, DOB DATE, PRIMARY KEY (emp_id));
```

- Field Attribute **NOT NULL** is being used because we do not want this field to be NULL. So, if a user will try to create a record with a NULL value, then MySQL will raise an error.

- Field Attribute **AUTO_INCREMENT** tells MySQL to go ahead and add the next available number to the id field.

- Keyword **PRIMARY KEY** is used to define a column as a primary key. You can use multiple columns separated by a comma to define a primary key.

(II). Insert Command: To insert data into a MySQL table, you would need to use the SQL **INSERT INTO** command, insert data into MySQL table 'Employee'.

Syntax: Here is a generic SQL syntax of INSERT INTO command to insert data into the MySQL table.

```
mysql> INSERT INTO table_name(field1, field2,...field n ) VALUES ( value1, value2,...value N );
```

To insert string data types, it is required to keep all the values into double or single quotes.

```
mysql> use COLLEGE;
         Database changed

mysql> INSERT INTO employee(emp_id, emp_name, address, DOB) VALUES ( 101, "Shaik Mohammad
          Fareed", "Tirupati", '26-March-2025');

Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO employee(emp_id, emp_name, address, DOB) VALUES ( 102, "Shaik Muneer
          Pasha", "Chittoor", '14-April-2008');

Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO employee(emp_id, emp_name, address, DOB) VALUES ( 103, "Shaik Mujeeb
          Pasha", "Hyderabad", '18-August-2016');

Query OK, 1 row affected (0.01 sec)
```

(III). Select Command: The SQL **SELECT** command is used to fetch data from the MySQL database. You can use this command at mysql> prompt as well as in any script like PHP.

Syntax: Here is generic SQL syntax of SELECT command to fetch data from the MySQL table.

mysql> select column name1, column name2, . . . column name n from table name;

NOTE: You can use one or more tables separated by comma to include various conditions using a WHERE clause, but the WHERE clause is an optional part of the SELECT command.

mysql> SELECT column name1, column name2, ... column name n from table name1, table name 2, …. [where clause] [OFFSET M] [LIMIT N];

- You can fetch one or more fields in a single SELECT command.
- You can specify star (*) in place of fields. In this case, SELECT will return all the fields.
- You can specify any condition using the WHERE clause.
- You can specify an offset using **OFFSET** from where SELECT will start returning records. By default, the offset starts at zero.
- You can limit the number of returns using the **LIMIT** attribute.

```
mysql> use COLLEGE;
          Database changed

mysql> SELECT * from employee;
+------------+--------------------------+---------------+-------------------+
|  emp_id  |        emp_name         |   Address  |     DOB             |
+----------- +--------------------------+---------------+-------------------+
|      101    | Shaik Mohammad Fareed | Tirupati       | 26-03-2025          |
|      102    | Shaik Muneer Pasha      | Chittoor       | 14-04-2008          |
|      103    | Shaik Mujeeb Pasha      | Hyderabad     | 18-08-2016          |
+-----------+--------------------------+----------  --+---------------------+
3 rows in set (0.01 sec)
```

(IV). Update Commands: There may be a requirement where the existing data in a MySQL table needs to be modified. You can do so by using the SQL **UPDATE** command. This will modify any field value of any MySQL table.

Syntax: UPDATE table_name SET field1 = new-value1, field2 = new-value2 [WHERE Clause];

- You can update one or more field altogether.
- You can specify any condition using the WHERE clause.
- You can update the values in a single table at a time.
- The WHERE clause is very useful when you want to update the selected rows in a table.

```
mysql> use TUTORIALS;
Database changed

mysql> UPDATE employee set address = 'Nellore' where  emp_id = 102;

Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * from employee;
+------------+--------------------------+----------------+-------------------+
|   emp_id  |         emp_name         |     Address   |      DOB            |
+------------+--------------------------+----------------+-------------------+
|      101   | Shaik Mohammad Fareed   | Tirupati        | 26-03-2025       |
|      102   | Shaik Muneer Pasha       | Nellore        | 14-04-2008        |
|      103   | Shaik Mujeeb Pasha        | Hyderabad     | 18-08-2016       |
+------------+--------------------------+---------- --+---------------------+
3 rows in set (0.01 sec)
```

(V) Replace Command: The MySQL REPLCAE statement is a MySQL extension to the standard SQL. The MySQL REPLCAE Statement works as follows:

- If the new row already does not exist, the MySQL REPLACE statement inserts a new row.
- If the new row already exists, the REPLACE statement deletes the old row first and then inserts a new row. In some cases, the REPLACE statement updates the existing row only.

Syntax: Replace Into table_name(column_name1, column_name2,….)values(value1, value2,….value n);

```
mysql> replace into employee(contact_number) values (123456), (987654), (010101);
```

```
mysql> SELECT * from employee;
+------------+--------------------------+--------------------+-------------------+
|   emp_id  |         emp_name        | Contact_Number |      DOB          |
+------------+--------------------------+--------------------+-------------------+
|      101   | Shaik Mohammad Fareed | 123456             | 26-03-2025       |
|      102   | Shaik Muneer Pasha       | 987654             | 14-04-2008       |
|      103   | Shaik Mujeeb Pasha       | 010101             | 18-08-2016       |
+------------+--------------------------+--------------------+-------------------+
3 rows in set (0.01 sec)
```

(VI). Delete Command: If you want to delete a record from any MySQL table, then you can use the SQL command DELETE FROM. You can use this command at the mysql> prompt as well as in any script like PHP.

Syntax :- DELETE FROM table_name [WHERE Clause];

- If the WHERE clause is not specified, then all the records will be deleted from the given MySQL table.
- You can specify any condition using the WHERE clause.
- You can delete records in a single table at a time.

The WHERE clause is very useful when you want to delete selected rows in a table.

```
mysql> use TUTORIALS;
Database changed
```

```
mysql> DELETE FROM employee WHERE emp_id=1;
Query OK, 1 row affected (0.23 sec)
```

(VII).Where Command: We have seen the SQL SELECT command to fetch data from a MySQL table. We can use a conditional clause called as the WHERE, to filter out the results. Using this WHERE clause, we can specify a selection criteria to select the required records from a table.

Syntax: select field1, field2,…..field n from table1 where condition;

```
var mysql = require('mysql');
var con = mysql.createConnection ({
  host: "localhost",
  user: "mohammadfareed",
  password: "fareed",
  database: "COLLEGE"
});
con.connect(function(err)
{
  if (err) throw err;
        con.query("SELECT * FROM employee WHERE emp_id = '101' ", function (err, result)
   {
    if (err) throw err;
    console.log(result);
  });
});
```

Output:

```
+------------+---------------------------+---------------------+--------------------+
|   emp_id  |         emp_name        | Contact_Number |      DOB               |
+------------+---------------------------+---------------------+--------------------+
|      101   | Shaik Mohammad Fareed |  123456                26-03-2025          |
+------------+---------------------------+---------------------+--------------------+
```

- You can use one or more tables separated by a comma to include various conditions using WHERE clause, but the WHERE clause is an optional part of the select command.
- You can specify any condition using the WHERE clause.
- You can specify more than one condition using the AND or the OR operation.
- A WHERE clause can be used along with DELETE or UPDATE SQL command also to specify a condition.

****

Composed By:
Shaik Mohammad Fareed. MCA, M.Sc CS, IRPM.
Head, Department of Computer Science & Applications
Mail ID: fareedshaik47@gmail.com
Web Portal: www.fareed.com
Contact Number: 9491202987