Web Technology II

Web Technology refers to the various tools and techniques that are utilized in the process of communication between different types of devices over the internet. A web browser is used to access web pages. Web browsers can be defined as programs that display text, data, pictures, animation, and video on the Internet. Hyperlinked resources on the World Wide Web can be accessed using software interfaces provided by Web browsers.

Web Technology can be classified into the following sections:

i. **World Wide Web (WWW):** The World Wide Web -- also known as the web, WWW or W3 -- refers to all the public websites or pages that users can access on their local computers and other devices through the internet. These pages and documents are interconnected by means of hyperlinks that users click on for information. This information can be in different formats, including text, images, audio and video. The World Wide Web is based on several different **technologies:** Web browsers, Hypertext Markup Language (HTML) and Hypertext Transfer Protocol (HTTP).

ii. **Web Browser:** The web browser is an application software to explore www (World Wide Web). It provides an interface between the server and the client and requests to the server for web documents and services.

iii. **Web Server:** Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP).

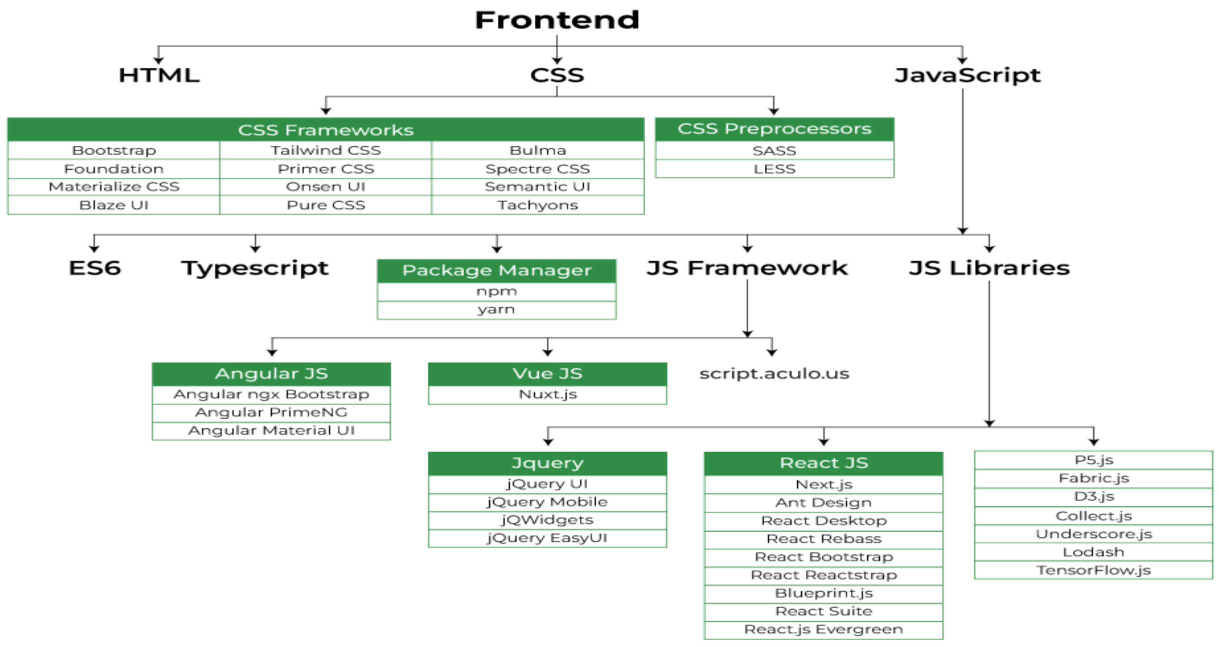
iv. **Web Pages:** A webpage is a digital document that is linked to the World Wide Web and viewable by anyone connected to the internet has a web browser.

v. **Web Site:** A website (also written as web site) is a collection of web pages and related content that is identified by a common domain name and published on at least one web server. Notable examples are ctevt.org.np, hseb.edu.np, and psc.gov.np All publicly accessible websites collectively constitute the World Wide Web. There are also private websites that can only be accessed on a private network, such as a company's internal website for its employees.

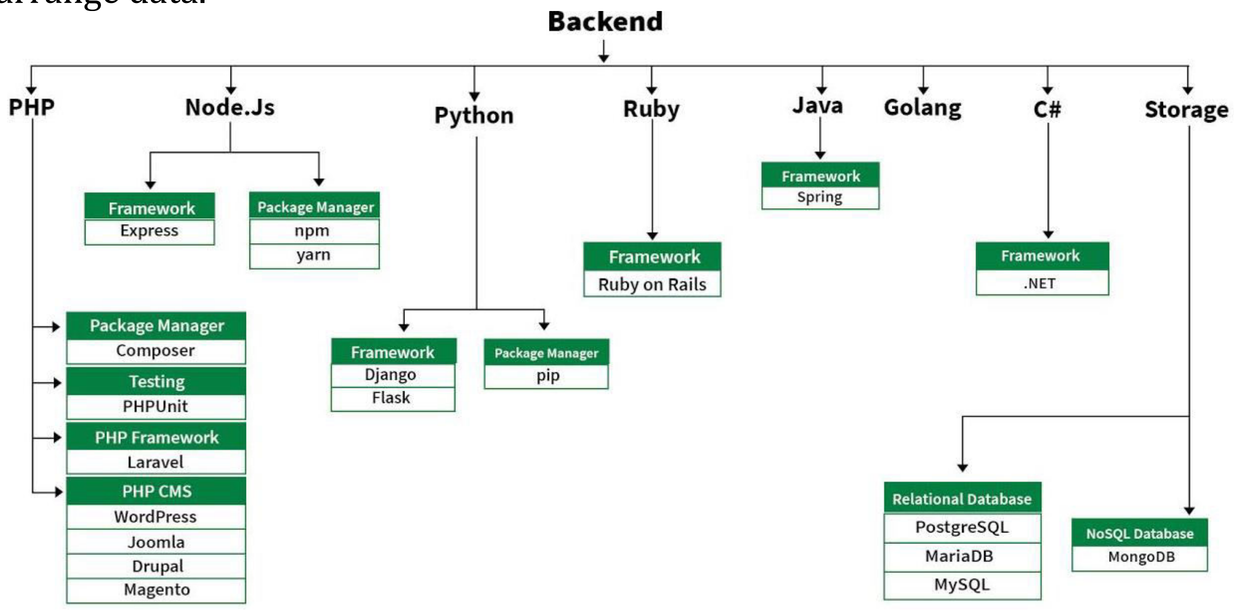
vi. **Web Development:** Web development refers to the building, creating, and maintaining of websites. It includes aspects such as web design, web publishing, web programming, and database management. It is the creation of an application that works over the internet i.e., websites.

Web Development can be classified into two ways:

a) **Frontend Development:** The part of a website that the user interacts directly is termed as front end. It is also referred to as the 'client side' of the application.



b) **Backend Development:** Backend is the server side of a website. It is the part of the website that users cannot see and interact. It is the portion of software that does not come in direct contact with the users. It is used to store and arrange data.



Frontend Languages:

The front-end portion is built by using some languages which are discussed below:

i. **HTML:** HTML stands for Hypertext Markup Language. It is used to design the front-end portion of web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text documentation within the tag which defines the structure of web pages.

ii. **CSS:** Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

iii. **JavaScript:** JavaScript is a famous scripting language used to create magic on the sites to make the site interactive for the user. It is used to enhancing the functionality of a website to running cool games and web-based software.

iv. **AJAX:** Ajax is an acronym for Asynchronous JavaScript and XML. It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.

There are many other languages through which one can do front-end development depending upon the framework for
example *Flutter* user *Dart*, *React* uses *JavaScript* and *Django* uses *Python*, and much more.

Backend Languages:

The back-end portion is built by using some languages which are discussed below:

i. **PHP:** PHP is a server-side scripting language designed specifically for web development. Since PHP code executed on the server-side, so it is called a server-side scripting language.

ii. **Node.js:** Node.js is an open-source and cross-platform runtime environment for executing JavaScript code outside a browser. You need to remember that NodeJS is not a framework, and it's not a programming language. Most people are confused and understand it's a framework or a programming language. We often use Node.js for building back-end services like APIs like Web App or Mobile App. It's used in production by large companies such as PayPal, Uber, Netflix, Wallmart, and so on.

iii. **Python:** Python is a programming language that lets you work quickly and integrate systems more efficiently.

iv. **Ruby:** Ruby is a dynamic, reflective, object-oriented, general-purpose programming language. Ruby is a pure Object-Oriented language developed by Yukihiro Matsumoto. Everything in Ruby is an object except the blocks but there are replacements too for it i.e., procs and lambda. The objective of Ruby's development was to make it act as a sensible buffer between human programmers and the underlying computing machinery.

v. **Java:** Java is one of the most popular and widely used programming languages and platforms. It is highly scalable. Java components are easily available.

vi. **JavaScript:** JavaScript can be used as both (front end and back end) programming.

vii. **Golang:** Golang is a procedural and statically typed programming language having the syntax similar to C programming language. Sometimes it is termed as Go Programming Language.

viii. **C#:** C# is a general-purpose, modern and object-oriented programming language pronounced as "C sharp".

ix. **DBMS:** The software which is used to manage database is called Database Management System (DBMS).

Server-side and Client-side scripting

When we talk about the web site then we must know that the web site is classified into **two types:**

i. **The static website:** Static webpages are relatively straightforward. They are written in HTML, JavaScript, CSS, and other languages. When a server receives a request from a static webpage, it transmits the response to the client without performing any more processing.

ii. **The dynamic website:** Dynamic websites are those that generate webpages in real-time. Web scripting code, such as PHP or ASP, is used on these pages. The Web server parses the code, and the generated HTML is transmitted to the client's browser when a client requests a dynamic page.

Server-side scripting is used at the backend, where the source code is not viewable or hidden at the client side (browser).

- It helps work with the back end.
- It doesn't depend on the client.
- It runs on the web server.
- It helps provide a response to every request that comes in from the user/client.
- This is not visible to the client side of the application.
- It requires the interaction with the server for the data to be process.
- Server-side scripting requires languages such as PHP, ASP.net, ColdFusion, Python, Ruby on Rails.
- It is considered to be a secure way of working with applications.
- It can be used to customize web pages.
- It can also be used to provide dynamic websites.

Client-side scripting is used at the front end which users can see from the browser.

- It helps work with the front end.
- It is visible to the users.
- The scripts are run on the client browser.
- It runs on the user/client's computer.
- It depends on the browser's version.
- It doesn't interact with the server to process data.
- Client-side scripting involves languages such as HTML, CSS, JavaScript.
- It helps reduce the load on the server.
- It is considered to be less secure in comparison to server-side scripting.

Difference between client-side scripting and server-side scripting :

Client-side scripting	Server-side scripting
Source code is visible to the user.	Source code is not visible to the user because its output of server-side is an HTML page.
Its main function is to provide the requested output to the end user.	Its primary function is to manipulate and provide access to the respective database as per the request.
It usually depends on the browser and its version.	In this any server-side technology can be used and it does not depend on the client.
It runs on the user's computer.	It runs on the webserver.
There are many advantages linked with this like faster. response times, a more interactive application.	The primary advantage is its ability to highly customize, response requirements, access rights based on user.
It does not provide security for data.	It provides more security for data.
It is a technique used in web development in which scripts run on the client's browser.	It is a technique that uses scripts on the webserver to produce a response that is customized for each client's request.
HTML, CSS, and JavaScript are used.	PHP, Python, Java, Ruby are used.
No need of interaction with the server.	It is all about interacting with the servers.

Introduction of Internet Technology

Internet Technologies is a technical field that covers the necessary skills to develop applications on the Internet or Internet based systems, connecting e-commerce, cloud, mobile, and Web based technologies.
The Internet (or internet) is the global system of interconnected computer networks that uses the Internet protocol suite (TCP/IP) to communicate between networks and devices. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The Internet carries a vast range of information resources and services, such as the inter-linked hypertext documents and applications of the World Wide Web (WWW), electronic mail, telephony, and file sharing.

JavaScript

JavaScript is a scripting language which is done on a client-side. The various browser supports JavaScript technology. DHTML uses the JavaScript technology for accessing, controlling, and manipulating the HTML elements. The statements in JavaScript are the commands which tell the browser for performing an action.

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

History of JavaScript

In 1993, **Mosaic**, the first popular web browser, came into existence. In the **year 1994**, **Netscape** was founded by **Marc Andreessen**. He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part-time programmers. Consequently, in 1995, the company recruited **Brendan Eich** intending to implement and embed Scheme programming language to the browser. But, before Brendan could start, the company merged with **Sun Microsystems** for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platforms. Now, two languages were there: Java and the scripting language. Further, Netscape decided to give a similar name to the scripting language as Java's. It led to 'JavaScript'. Finally, in May 1995, Marc Andreessen coined the first code of JavaScript named '**Mocha**'. Later, the marketing team replaced the name with '**LiveScript**'. But, due to trademark reasons and certain other reasons, in December 1995, the language was finally renamed to 'JavaScript'. From then, JavaScript came into existence.

Application of JavaScript

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

Advantages of JavaScript

The merits of using JavaScript are –

- **Less server interaction:** You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity:** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces:** You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features –

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multi-threading or multiprocessor capabilities.

JavaScript - Syntax

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** or **<script type="text/javascript">... </script>** HTML tags in a web page.

We can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.

The **<script>** tag alerts the browser program to start interpreting all the text between these tags as a script.

A simple syntax of your JavaScript will appear as follows.

```
<script ...>
        JavaScript code
</script>
```

The script tag takes two important attributes:

- **Language**: This attribute specifies what scripting language you are using. Typically, its value will be JavaScript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type**: This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

  So our JavaScript segment will look like:

```
<script language = "javascript" type = "text/javascript">
        JavaScript code
</script>
```

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
        document.write("Hello World!");
    </script>
  </body>
</html>
```

Semicolons are Optional

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

```
<script language = "javascript" type = "text/javascript">
    var1 = 10
    var2 = 20
  document.write(var1)
</script>
```

Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

NOTE – Care should be taken while writing variable and function names in JavaScript.

Comments in JavaScript

JavaScript supports both C-style and C++-style comments, thus:

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

Example

```
<script language = "javascript" type = "text/javascript">
  <!--
    // This is a comment. It is similar to comments in C++

    /*
    * This is a multi-line comment in JavaScript
    * It is very similar to comments in C Programming
    */
  -->
</script>
```

JavaScript - Placement in HTML File

There is a flexibility given to include JavaScript code anywhere in an HTML document. However, the most preferred ways to include JavaScript in an HTML file are as follows:

- An Internal script can be added anywhere in the same HTML file.

```
<html>
  <head>
    <script type = "text/javascript">
        function sayHello() {
          alert("Hello World")
        }

    </script>
  </head>
  <body>
    <input type = "button" onclick = "sayHello()" value = "Say Hello" />
  </body>
</html>
```

- An External script file must be saved by .js extension and then include in <head>...</head> section.

  message.js

  ```
  function msg(){
          alert("Hello students");
  }
  ```

  Include the JavaScript file into html page. It use the JavaScript function on button click.

  index.html

  ```
  <html>
          head>
                  <script type="text/javascript" src="message.js"></script>
          </head>
          <body>
                  <p>Welcome to JavaScript</p>
                  <form>
                          <input type="button" value="click" onclick="msg()"/>
                  </form>
          </body>
  </html>
  ```

JavaScript Variable

There are some rules while declaring a JavaScript variable (also known as identifiers).

- Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
- After first letter we can use digits (0 to 9), for example value1.
- JavaScript variables are case sensitive, for example x and X are different variables.

  Correct JavaScript variables

  ```
  var x = 10;
  var _value="sonoo";
  ```

  Incorrect JavaScript variables

  ```
  var  123=30;
  var *aa=320;
  ```

  Example

  ```
  <script>
          var x = 10;
          var y = 20;
          var z=x+y;
          document.write(z);
  </script>
  ```

JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- **Global Variables:** A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

JavaScript Datatypes

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types:

- **Numbers**, e.g., 123, 120.50 etc.
- **Strings** of text e.g. "This text string" etc.
- **Boolean** e.g., true or false.

JavaScript Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	Else	instanceof	switch	boolean
enum	Int	synchronized	break	export
interface	This	byte	extends	long
throw	Case	false	native	Throws
catch	Final	new	transient	char
finally	Null	true	class	float
package	Try	const	for	Private
continue	Function	protected	var	debugger
goto	Public	void	default	if
return	Volatile	delete	implements	short
while	Do	import	static	With
double	In	super	typeof	

Operator

Operator is a symbol that operates on given data and the variable, that takes action on operands.

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.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Arithmetic operators

Arithmetic operators perform arithmetic operation on numbers (literals or variables).

Operator	Description
+	Addition
-	Subtraction
*	Multiplication

**	Exponentiation
	The exponentiation operator (**) raises the first operand to the power of the second operand.
	let z = x ** 2;
	x ** y produces the same result as Math.pow(x,y)
	let z = Math.pow(x,2);

/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

Assignment operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Ternary operator

The ternary(conditional) operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Operator	Description
?: (Conditional)	If Condition is true? Then value X: Otherwise, value Y e.g., (a>b)? True result: False result;

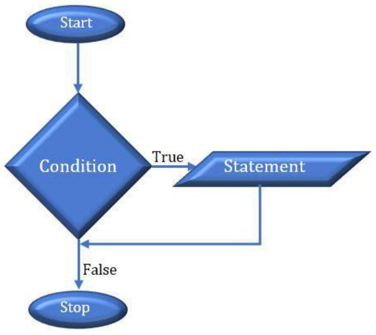
Control Statements

JavaScript supports control statements which are used to perform different actions based on different conditions.

Branching

- **if statement**

  The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.



Syntax

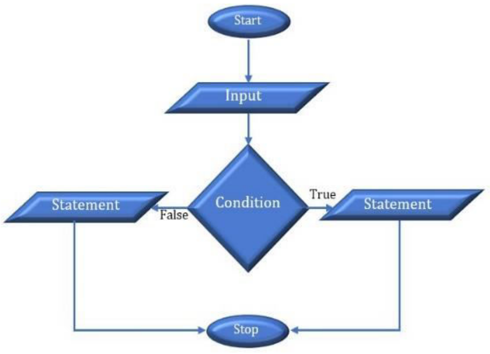
```
if (expression) {
  Statement(s) to be executed if expression is true
}
```

Example

```
<script type = "text/javascript">
        var age = 20;
        if( age > 18 ) {
          document.write("<b>Qualifies for driving</b>");
        }
 </script>
```

- **if...else statement**

  The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.



Syntax

```
if (expression) {
  Statement(s) to be executed if expression
is true
} else {
  Statement(s) to be executed if expression is false
}
```

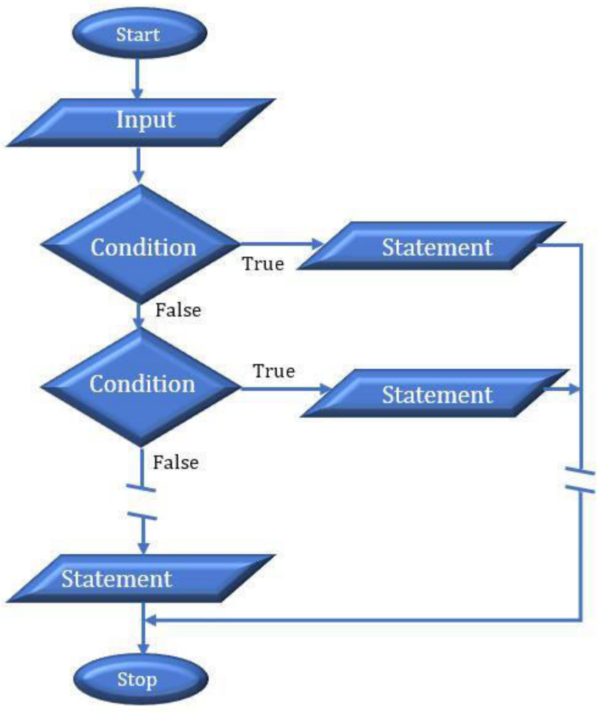
Example

```
<script type = "text/javascript">
        var age = 20;
        if( age > 18 ) {
          document.write("<b>Qualifies for driving</b>");
        }else{
          document.write("<b>Not qualifies for driving</b>");
        }
```

```
</script>
```

- **if...else if... statement**

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.



Syntax

```
if (expression 1) {
  Statement(s) to be executed if expression 1 is true
} else if (expression 2) {
  Statement(s) to be executed if expression 2 is true
} else if (expression 3) {
  Statement(s) to be executed if expression 3 is true
} else {
  Statement(s) to be executed if no expression is true
}
```

Example

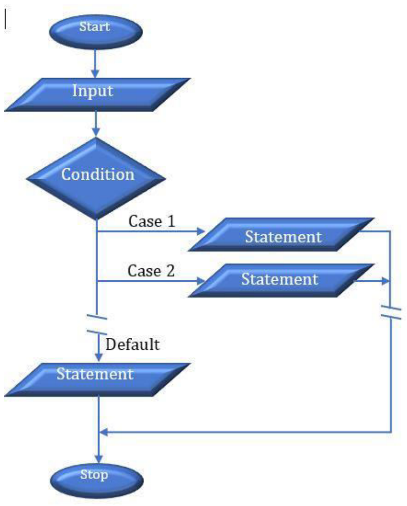
```
<script type = "text/javascript">
        var num = 1;
        if(num <0 1)
          document.write("<b>Number is negative</b>");
        else if(num>0)
          document.write("<b>Number is positive</b>");
        else
          document.write("<b>Number is zero</b>");
</script>
```

- **switch…… case**

The objective of a switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

Syntax

```
switch (expression) {
  case condition 1: statement(s)
  break;
  case condition 2: statement(s)
  break;
  ...
  case condition n: statement(s)
  break;
  default: statement(s)
}
```



The break statements indicate the end of a particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

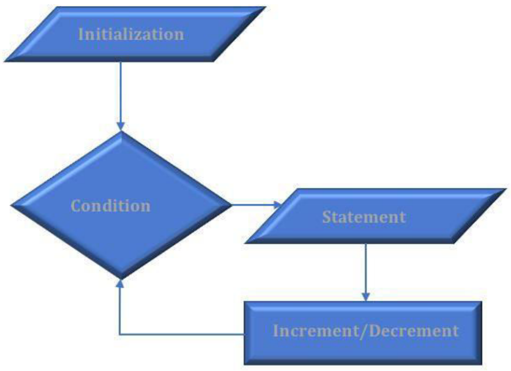
Example

```
<script type = "text/javascript">
   var day = 1;
   switch (day) {
     case 1:
               document.write("Sunday<br />");
               break;
   case 2:
               document.write("Monday<br />");
               break;
   case 3:
               document.write("Tuesday<br />");
               break;
   case 4:
               document.write("Wednesday<br />");
               break;
   case 5:
               document.write("Thursday<br />");
               break;
   case 6:
               document.write("Friday<br />");
               break;
   case 7:
               document.write("Saturday<br />");
               break;
    default:
               document.write("Unknown day name<br />");
               break;
   }
</script>
```

Iteration

- **For loop**

  The 'for' loop is the most compact form of looping. It three important parts: **initialization, test statement, increase or decrease**. You must put all the three parts in a single line separated by semicolons.



Syntax

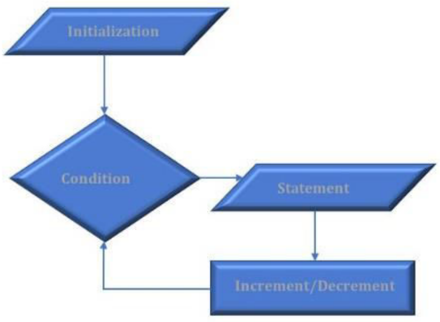
```
for (initialization; condition; increment)
{
   code to be executed
}
```

Example

```
<script>
        var i;
        for (i=1; i<=5; i++)
        {
                document.write(i + "<br/>")
        }
</script>
```

- **While Loop**

  The most basic loop in JavaScript is the while loop. The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.



Syntax

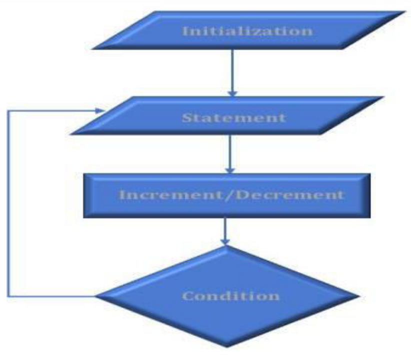
```
while (condition)
{
   code to be executed
}
```

Example

```
<script>
        var i=11;
        while (i<=15)
        {
                document.write(i + "<br/>");
                i++;
        }
</script>
```

- **Do while loop**

The **JavaScript do while loop** *iterates the elements for the infinite number of times* like while loop. But, code is *executed at least* once whether condition is true or false.



Syntax

```
do{
   code to be executed
}while (condition);
```

Example

```
<script>
var i=21;
do{
document.write(i + "<br/>");
i++;
}while (i<=25);
</script>
```

Input Methods

In JavaScript, we use the prompt() function to ask the user for input. As a parameter, we input the text we want to display to the user. Once the user presses "ok," the input value is returned. We typically store user input in a variable so that we can use the information in our program.

```
<script>
        var name = prompt("What is your name?");
        var num = prompt("What is your favorite number? ");
        document.write("Hello " + name + "!");
        document.write(num + "?! That's my favorite number too!");

</script>
```

Output Methods

JavaScript can "display" data in different ways:

- Writing into an HTML element, using **innerHTML**.
- Writing into the HTML output using **document.write()**.
- Writing into an alert box, using **window.alert()**.
- Writing into the browser console, using **console.log()**.

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

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
```

```
<p>My First Paragraph</p>
<p id="demo"></p>
<script>
        document.getElementById("demo").innerHTML = 5 + 6;
</script>
</body>
</html>
```

Using document.write()

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

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
document.write(5 + 6);
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<button type="button" onclick="document.write(5 + 6)">Try it</button>
</body>
</html>
```

Using document.write() after an HTML document is loaded, will delete all existing HTML:

Using window.alert()

You can use an alert box to display data:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
```

```
        window.alert(5 + 6);
</script>
</body>
</html>
```

We can skip the window keyword.

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
        alert(5 + 6);
</script>
</body>
</html>
```

Example 1

```
<!DOCTYPE html>
<html>
    <head>
            <title>Bulb OnOff</title>
      </head>
      <body>
            <h2>What Can JavaScript Do?</h2>

            <p>JavaScript can change HTML attribute values.</p>
            <p>JavaScript changes the value of the src (source) attribute of an image.</p>
            <button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">
                  Turn on the light
            </button>
            <img id="myImage" src="pic_bulboff.gif" style="width:100px">
            <button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">
                 Turn off the light
            </button>

      </body>
</html>
```

Example 2

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

<u>Function</u>

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Syntax

```
<script type = "text/javascript">
    function functionname(parameter-list)
    {
                Statements
  }
</script>
```

Example

```
<script type = "text/javascript">
    function sayHello() {
      alert("Hello we are DA");
    }
   </script>
```

Example

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello() {
        document.write ("Hello we are DA");
      }
    </script>

  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Hello to DA">
    </form>
    <p>Use different text in write method and then try...</p>
  </body>
</html>
```

Example

Function with parameter

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello(a,b) {
                var sum;
                sum = a+b;
                document.write (sum);
      }
    </script>
  </head>
  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello(5,7)" value = "SUM">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

Example
Function with parameter

```
<html>
  <head>
    <script type = "text/javascript">
      function concatenate(name1, name2) {
        var full;
        full = name1 + name2;
        return full;
      }
      function secondFunction() {
        var result;
        result = concatenate('DAYA', 'AJAY');
        document.write (result);
      }
    </script>
  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "secondFunction()" value = "Call Function">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

Some worked out examples:

WAP to find area and circumference of circle.

```
<script>
        var r, a, c;
        r=parseInt(prompt("Enter radius value: "));
        a=3.14*r*r;
        c=2*3.14*r;
        document.write("Area of circle= "+a.toFixed(2));
        document.write("<BR>"+"Circumference of circle : "+c.toFixed(2));
</script>
```

WAP to find the square root of a number given by the user, use Math.sqrt function.

```
<script>
var num, sqr;
 num=parseInt(prompt("Enter any number : "));
 sqr=Math.sqrt(num);
 document.write ("Answer = "+sqr);
</script>
```

Write a program to find the greatest number among three numbers.

```
<script>
        var num1,num2,num3;
        num1=parseInt(prompt("Enter first number : "));
        num2=parseInt(prompt("Enter second number : "));
        num3=parseInt(prompt("Enter third number : "));
        if(num1>num2 && num1>num3)
                document.write (" Greatest number =  "+num1);
        else if(num2>num3)
                document.write (" Greatest number =  "+num2);
        else
                document.write (" Greatest number =  "+num3);
</script>
```

WAP to find even or odd number.

```
<script type="text/javascript">
        var num;
        num=parseInt(prompt("Enter any number : "));
        if(num%2==0)
                document.write (" Even number ");
        else
                document.write ("Odd number ");
</script>
```

WAP to input your name and age using form and know whether he/she is eligible to vote or not. [Use function and onclick event]

```
<form name="myForm">
 <input type="text" name="uname" placeholder="Enter name">
 <br><br>
 <input type="text" name="age" placeholder="Enter age">
 <br><br>
 <input type="button" value="Click here"  onclick="test()">
</form>

<script>
        function test() {
        var naam=document.myForm.uname.value;
        var a=document.myForm.age.value;
        if(a>18)
                document.write(naam+" You are eligible to vote ");
        else
                document.write(naam+" Sorry! You are not eligible to vote");
        }
</script>
```

WAP to Grade of a student and show "you have obtained grade A+/A/B+/B/C+/C/D", using switch case.

```
<script>
        var grade=prompt(" Enter your grade : ");
        switch(grade)
        {
                case "A+":
                        document.write( " <br> A+");
                        break;
                case "A":
                        document.write( " <br> A");
                        break;
                case "B+":
                        document.write( " <br> B+");
                        break;
                case "B":
                        document.write( " <br> B");
                        break;
                default:
                        document.write(" <br> Other Grade");
        }
</script>
```

WAP to print 1, 3, 5 ,....100 using a while loop.

```
<input type="button" onclick="oddNumbers()" value="start">
<script>
        function oddNumbers() {
                var i=1;
                while(i<=100)
                {
                        document.write(i +"  ");
                        i=i+2;
                }
        }
</script>
```

WAP to print factorial value of a number.

```
<script>
        var num,i,f=1;
        num=parseInt(prompt("Enter any number: "));
        for(i=1;i<=num;i++)
        {
                f=f*i;
        }
        document.write("Factorial is "+f);
</script>
```

WAP to print multiplication table of a number. Use function.

```
<input type="text" id="num" placeholder="Enter a number"><br><br>
<input type="button" onclick="mtable()" value="multiplication table">
<script>
        function mtable(){
                var num,i;
                num=document.getElementById("num").value;
                for(i=1;i<=10;i++)
                {
                        document.write(num+"x"+i+"="+num*i+"<br>");
                }
        }
</script>
```

WAP to print factors of a number. Use function.

```
<input type="text" id="num" placeholder="Enter a number"><br><br>
<input type="button" onclick="factors()" value="factors">
<script>
        function factors(){
                var num,i;
                num=document.getElementById("num").value;
                        for(i=1; i<=num; i++)
                        {
                                if((num%i)==0)
                                document.write(i +"<BR>");
                        }
        }
</script>
```

WAP to reverse a string. [use loop from its length-1 to 0 and print it].

```
<script>
        var da, i;
        da=prompt("Enter any sting : ");
        for(i=da.length-1;i>=0;i--)
        {
                document.write(da[i]);
        }
</script>
```

An array contains multiple strings. Print them in reverse order. [use array.reverse() method].

```
<script>
        var days = ["Sun","Mon","Tues","Wed","Thur","Fri","Sat"];
        document.write (days.reverse());
</script>
```

WAP to insert a string “We are learning JS” via form object getElementById() method. Use paragraph with id.

```
<p id="result"> </p>
<script type="text/javascript">
        document.getElementById("result").innerHTML ="We are learning Java Script";
</script>
```

WAP to use date and time.

```
<script>
        var today=new Date();
        var h=today.getHours();
        var m=today.getMinutes();
        var s=today.getSeconds();
        var y=today.getFullYear();
        var m=today.getMonth();
        var d=today.getDate();
        document.getElementById('txt').innerHTML=h+":"+m+":"+s+"-"+y+"-"+m+"-"+d;
</script>
```

Programs for string implementation

WAP to check the given string is palindrome or not.

```
<script>
        function palin() {
                var a, st1,st, i;
                orgSt=document.myform.str1.value;
                splSt = orgSt.split('');
                revSt = splSt.reverse();
                revJoin = revSt.join('');
                if(orgSt==revJoin){
                        alert("String is palindrome.");
                }else{
                        alert("String is not palindrome.");
                }
        }
</script>
<form name="myform">
        <input type="text" name="str1" placeholder="Palindrome">
        <button onclick="palin()">Palindrome Check</button>
</form>
```

WAP to find the length of given string

```
<script>
     function strLength(){
          orgSt=document.myform.str1.value;
          document.write(orgSt.length);
     }
</script>
<form name="myform">
     <input type="text" name="str1" placeholder="String">
     <button onclick="strLength()">String Length</button>
</form>
```

WAP to get sub string form given string

```
<script>
        function subString(){
                orgSt=document.myform.str1.value;
                strt=document.myform.strt.value;
                stend=document.myform.stend.value;
                part = orgSt.substr(strt, stend);
                alert("Substring is: "+part);
        }
</script>
<form name="myform">
     <input type="text" name="str1" placeholder="String">
     <input type="text" name="strt" placeholder="Start">
     <input type="text" name="stend" placeholder="End">
     <button onclick="subString()">Sub String</button>
</form>
```

WAP to convert in upper case

```
<script>
        function Uppercase(){
                orgSt=document.myform.str1.value;
                upr=orgSt.toUpperCase() ;
                alert("Upper case is: "+upr);
        }
</script>
<form name="myform">
     <input type="text" name="str1" placeholder="String">
     <button onclick="Uppercase()">Upper Case</button>
</form>
```

WAP to convert into lower case

```
<script>
        function Lowercase(){
                orgSt=document.myform.str1.value;
                lwr=orgSt.toLowerCase() ;
                alert("Lower case is: "+lwr);
        }
</script>
<form name="myform">
     <input type="text" name="str1" placeholder="String">
     <button onclick="Lowercase()">Lower Case</button>
</form>
```

WAP to concatenate two string

```
<script>
        function conct(){
                orgSt=document.myform.str1.value;
                strt=document.myform.str2.value;
                part = orgSt.concat(' ');
                part2=part.concat(strt);
                alert("Concat is: "+part2);
        }
</script>
<form name="myform">
     <input type="text" name="str1" placeholder="String">
     <input type="text" name="str2" placeholder="String">
     <button onclick="conct()">Concate String</button>
</form>
```

WAP to validate form using JavaScript.

```
<!DOCTYPE html>
<html>
<head>
 <title>JavaScript Validation</title>
  <style type="text/css">
  .error{
        color: #FFFFFF; font-size:18px;
  }
        input[type=text], [type=password]{
                height:50px; width:250px; border-radius:10px; font-size:16px;
                padding:5px;
        }
        input[type=submit], [type=reset]
        {
                height:50px; width:85px;  background-color:#05DDAB; color:#FFFFFF;
                font-weight:bold; font-size:18px; border-radius:10px;
        }
        .outer{
                background-color:#ABA7A7; border-radius:10px;
                margin:100px 0px 0px 300px; height:220px; width:250px; padding:50px;
        }
  </style>
        <script>
                function validate() {
                        var uname=document.myForm.user.value;
                        var pass=document.myForm.pass.value;
                        var err = " field is required";
                        if (uname==""){
                                document.getElementById("usererror").innerHTML="User
name "+err;
                                return false;
                        }else if(pass==""){
```

```
        document.getElementById("passerror").innerHTML="Password "+err;
                                    return false;
                    }
   }
   </script>
</head>
<body>
<div class="outer">
 <form name="myForm" method="post" onsubmit="return validate()">
    <input type="text" name="user" placeholder="User name">
                  <br>
    <span id="usererror" class="error"></span>
    <br>
    <input type="password" name="pass" placeholder="Enter password">
    <br>
          <span id="passerror" class="error"></span>
    <br>
    <input type="submit" name="submit" value="Login">
    <input type="reset" name="reset" value="Cancel">
 </form>
 </div>
</body>
</html>
```

WAP to show and hide a paragraph using jQuery

```
<!DOCTYPE html>
<html>
        <head>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
        </script>
        <script>
              $(document).ready(function(){
                     $(".sh").click(function(){
                            $(".content").show();
                     });
                     $(".hd").click(function(){
                            $(".content").hide();
                     });
              });
 </script>
</head>
 <body>
        <h2> Simple Example of jQuery </h2>
        <button class="sh">Show </button>
        <button class="hd">Hide </button >
        <p class="content"> This is a sample paragraph </p>
</body>
</html>
```

WAP to show alert box using jQuery

```
<!DOCTYPE html>
<html>
      <head>
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
      </script>
      <script>
            $(document).ready(function(){
                  $("button").click(function(){
                        alert("You clicked");
                  });
            });
    </script>
<body>
        <button>Click here</button>
</body> </html>
```

JavaScript Events

click	The event occurs when the user clicks on an element	MouseEvent
dblclick	The event occurs when the user double-clicks on an element	MouseEvent
focus	The event occurs when an element gets focus	FocusEvent
focusin	The event occurs when an element is about to get focus	FocusEvent
focusout	The event occurs when an element is about to lose focus	FocusEvent
input	The event occurs when an element gets user input	InputEvent, Event
keydown	The event occurs when the user is pressing a key	KeyboardEvent
keypress	The event occurs when the user presses a key	KeyboardEvent
keyup	The event occurs when the user releases a key	KeyboardEvent
load	The event occurs when an object has loaded	UiEvent, Event
mousedown	The event occurs when the user presses a mouse button over an element	MouseEvent
mouseenter	The event occurs when the pointer is moved onto an element	MouseEvent
mouseleave	The event occurs when the pointer is moved out of an element	MouseEvent
mousemove	The event occurs when the pointer is moving while it is over an element	MouseEvent
mouseover	The event occurs when the pointer is moved onto an element, or onto one of its children	MouseEvent
mouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	MouseEvent

mouseup	The event occurs when a user releases a mouse button over an element	MouseEvent
submit	The event occurs when a form is submitted	Event

PHP

The PHP 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 started out as a small open-source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- 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.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible –

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

Example:

```
<html>
        <head>
                <title>This is my first php page. </title>
        </head>
        <body>
                <?php echo "Hello world"; ?>
        </body>
</html>
```

Control Statement

Control statements are conditional statements that execute a block of statements if the condition is correct. The statement inside the conditional block will not execute until the condition is satisfied.

Types

1. if..........else

```
if($var="DAYA")
{
        echo "Your name is DAYA";
}else{
        echo "You are not DAYA";
}
```

2. switch case

```
switch case($var)
{
        case 1:
                echo "Sunday";
                break;
        case 2:
                echo "Monday";
                break;
        default:
                echo "No case match";
```

```
    }
```
3. ternary
   ```
   ($var=="DAYA") ? "Your name is DAYA" : "You are not DAYA";
   ```

Looping/Iteration

Loops are mainly used to repeat the process until the conditions are met. Until the conditions are met the loops repeatedly execute. If the condition is not met, the loop will execute an infinite number of times.

Types

1. For loop
   ```
   for($var=1; $var<=10; $var++)
   {
           echo $var;
   }
   ```
2. While loop
   ```
   $var=1;
   while($var<=10)
   {
           echo $var;
           $var++;
   }
   ```
3. Do...... While loop
   ```
   $var=1;
   do
   {
           echo $var;
           $var++
   } while($var<10);
   ```
4. For each loop
   ```
   foreach(array as key => value)
   {
           statement;
   }
   ```

Operators

Operators are used to perform operations on variables and values.
PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

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
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

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	**Same as...**	**Description**
x = y	x = y	The left operand gets set to the value of the expression on the right
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

Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	**Name**	**Example**	**Result**
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y

<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.
The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1. $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1. = $txt2	Appends $txt2 to $txt1

Array Operators
The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x=== $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x! = $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x!== $y	Returns true if $x is not identical to $y

Conditional Assignment Operators
The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	Result
?:	Ternary	$x = *expr1*? *expr2*: *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

Session
A session is a temporary and interactive information interchange between two or more communicating devices, or between a computer and user. A session creates a file in a temporary directory on the server where registered session variables and their values are stored. Typically, one end point requests a connection with another specified end point and if that end point replies agreeing to the connection, the end points take turns exchanging commands and data. The session begins when the connection is established at both ends and terminates when the connection is ended commonly 30 minutes duration.

Cookies
An HTTP cookie (also called web cookie, Internet cookie, browser cookie, or simply cookie) is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing. These are small files which are stored on a user's computer. They are designed to hold a modest amount of data specific to a particular client and website, and can be accessed either by the web server or the client computer.
Cookies are text files stored on the client computer and they are kept of use tracking purpose. Server script sends a set of cookies to the browser. For example, name, age, or identification number etc. The browser stores this information on a local machine for future use.
When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.