❖ Introduction to Scripting Languages

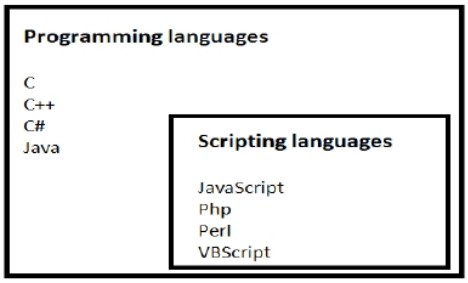
- The scripting language is basically a language where instructions are written for a run time environment.
- They do not require the compilation step and are rather interpreted.
- It brings new functions to applications and glue complex system together.

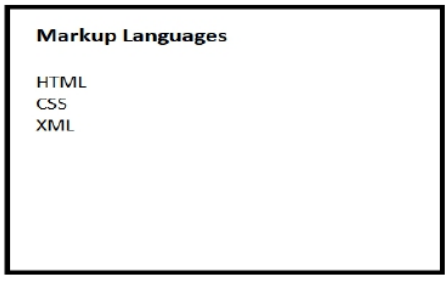
❖ Advantages of scripting languages:

- **Easy learning:** The user can learn to code in scripting languages quickly, not much knowledge of web technology is required.
- **Fast editing:** It is highly efficient with the limited number of data structures and variables to use.
- **Interactivity:** It helps in adding visualization interfaces and combinations in web pages. Modern web pages demand the use of scripting languages. To create enhanced web pages, fascinated visual description which includes background and foreground colors and so on.
- **Functionality:** There are different libraries which are part of different scripting languages. They help in creating new applications in web browsers and are different from normal programming languages.

❖ Application of Scripting Languages

- Scripting languages are used in web applications. It is used in server side as well as client side**. Server side scripting languages are:** JavaScript, PHP, Perl etc. **and client side scripting languages are:** JavaScript, AJAX, jQuery etc.
- Scripting languages are used in system administration. For example: Shell, Perl, Python scripts etc.
- It is used in Games application and Multimedia.
- It is used to create plugins and extensions for existing applications.





1 Client Side Scripting [JavaScript]

1.1 Introduction

- JavaScript is a lightweight, cross-platform, single-threaded, and interpreted compiled programming language.
- It is also known as the scripting language for webpages.
- It is well-known for the development of web pages, and many non-browser environments also use it.
- JavaScript can be used for Client-side developments as well as Server-side developments.
- JavaScript is both an imperative and declarative type of language.
- JavaScript contains a standard library of objects, like Array, Date, and Math, and a core set of language elements like operators, control structures, and statements.
- **Client-side:** It supplies objects to control a browser and it's Document Object Model (DOM). Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation. Useful libraries for the client side are AngularJS, ReactJS, VueJS, and so many others.
- **Server-side:** It supplies objects relevant to running JavaScript on a server. For if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is node.js.
- **Brendan Eich** first developed JavaScript, a computer language, in about ten days in **May 1995**. The language, formerly known as Mocha, later modified to LiveScript, and is now known simply as JavaScript, was created to be used on the client-side of websites, enabling the addition of dynamic and interactive components to static HTML texts.
- JavaScript was initially implemented in Netscape Navigator, which was the most popular browser at the time. The language was quickly adopted by Microsoft for use in Internet Explorer. Due to its simplicity of usage and the fact that it was the only client-side scripting language available at the time, JavaScript quickly gained popularity among web developers.
- JavaScript gained popularity during the ensuing years and was used to develop a wide range of web applications, such as online games, dynamic menus, and form validation. ECMAScript 4, a new version of the language, was planned in 2002, however, it was ultimately abandoned because of conflicts among the various browser vendors.
- With an estimated 95% of websites utilizing it in some capacity, JavaScript is currently one of the most popular programming languages in use worldwide. It is employed not only in web development but also in the creation of server-side applications, desktop and mobile applications, and even the programming of robots and other hardware.
- Since it was first released in **1995**, the JavaScript language has undergone a number of revisions, each of which has contributed new features and syntax.

1.2 Need of Client Side Scripting Language

Client-side scripting languages play a crucial role in web development by enabling the creation of dynamic and interactive user interfaces. These languages are executed on the user's web browser, allowing developers to enhance the functionality and user experience of web applications. Here are several reasons why client-side scripting languages are essential:

- **Enhanced User Experience:**
  - **Interactivity**: Client-side scripting allows developers to create interactive and responsive user interfaces. Users can experience dynamic content, real-time updates, and seamless interactions without the need to reload the entire web page.
  - **User Feedback:** Through client-side scripting, developers can provide instant feedback to users, such as form validation, error messages, and visual cues, improving the overall user experience.

- **Reduced Server Load:**
  - **Offloading Processing:** By handling certain tasks on the client side, such as form validation and data manipulation, the server load can be significantly reduced. This results in faster response times and improved scalability for web applications.
- **Asynchronous Operations:**
  - **AJAX (Asynchronous JavaScript and XML):** Client-side scripting enables the use of AJAX, allowing asynchronous data exchange between the client and server. This enhances the user experience by enabling partial page updates without requiring a full page reload.
- **Cross-Browser Compatibility:**
  - **Standardization:** Client-side scripting languages help in achieving a consistent user experience across different web browsers. Modern frameworks and libraries often abstract away browser-specific inconsistencies, making it easier for developers to write cross-browser compatible code.
- **Caching and Local Storage:**
  - **Local Storage:** Client-side scripting allows for the storage of data on the user's device, reducing the need to fetch the same data repeatedly from the server. This can improve the speed and efficiency of web applications, especially when dealing with large datasets.
- **Responsive Design:**
  - **Media Queries:** Client-side scripting is essential for implementing responsive web design, where the layout and presentation of a website adapt to different screen sizes and devices. Media queries in CSS, often triggered and manipulated by client-side scripts, enable this flexibility.
- **Rich User Interfaces:**
  - **Frameworks and Libraries:** Client-side scripting languages, when used in conjunction with frameworks and libraries like React, Angular, or Vue.js, allow developers to build complex and feature-rich user interfaces. These tools provide reusable components, state management, and other abstractions that streamline development.
- **Reduced Server Round Trips:**
  - **Form Handling:** Client-side scripting can be used to validate form inputs before submitting them to the server. This minimizes the need for server round trips to handle validation errors, resulting in a more efficient and responsive application.

1.3 Formatting and Coding Conventions

Formatting and coding conventions are essential aspects of writing maintainable and readable JavaScript code. Consistent and well-organized code not only makes it easier for developers to collaborate but also enhances code readability and reduces the likelihood of introducing errors. Here are some commonly accepted formatting and coding conventions for JavaScript:

- Naming and declaration rules for variables and functions.
- Rules for the use of white space, indentation, and comments.
- Programming practices and principles
- Coding conventions secure quality: which may include improves code readability and make code maintenance easier.
- Naming Conventions:
  - Always use the same naming convention for all your code. For example:
    - Variable and function names written as camelCase
    - Global variables written in UPPERCASE (We don't, but it's quite common) Constants (like PI) written in UPPERCASE
- Hyphens in HTML and CSS
  - HTML5 attributes can start with data- (data-quantity, data-price).
  - CSS uses hyphens in property-names (font-size).

- Hyphens are not allowed in JavaScript names.

➢ Underscores:
- Many programmers prefer to use underscores (date_of_birth), especially in SQL databases.
- Underscores are often used in PHP documentation.

➢ File Extensions
- HTML files should have a .html extension (not .htm).
- CSS files should have a .css extension.
- JavaScript files should have a .js extension

➢ Use Lower Case File Names
- If you use a mix of upper and lower case, you have to be extremely consistent.
- If you move from a case insensitive, to a case sensitive server, even small errors can break your web site.
- To avoid these problems, always use lower case file names (if possible).

➢ Performance
- Coding conventions are not used by computers. Most rules have little impact on the execution of programs. ‣ Indentation and extra spaces are not significant in small scripts.
- For code in development, readability should be preferred. Larger production scripts should be minified.

➢ Script Best Practices:
- Avoid global variables, avoid new, avoid ==, avoid eval()

➢ Avoid Global Variables
- Minimize the use of global variables.
- This includes all data types, objects, and functions.
- Global variables and functions can be overwritten by other scripts.

➢ Always Declare Local Variables
- All variables used in a function should be declared as local variables.
- Local variables must be declared with the var keyword or the let keyword, otherwise they will become global variables.
- Strict mode does not allow undeclared variables.

➢ Declarations on Top
- It is a good coding practice to put all declarations at the top of each script or function. This will:
  - Give cleaner code
  - Provide a single place to look for local variables
  - Make it easier to avoid unwanted (implied) global variables
  - Reduce the possibility of unwanted re-declarations.

➢ By default, JavaScript moves all declarations to the top.
- Initialize Variables
- It is a good coding practice to initialize variables when you declare them. This will:
  - Give cleaner code
  - Provide a single place to initialize variables
  - Avoid undefined values

1.4 JavaScript Comments

➢ JavaScript comments can be used to explain JavaScript code, and to make it more readable.

➢ JavaScript comments can also be used to prevent execution, when testing alternative code. Single Line Comments

✓ **Single line comments**
  - Single line comments start with //.

- Any text between // and the end of the line will be ignored by JavaScriptt (will not be executed)

✓ **Multi-line Comments**

- Multi-line comments start with /* and end with */
- Any text between /* and */ will be ignored by JavaScript.

- 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

The following example shows how to use comments in JavaScript.

```
<script language="javascript" type="text/javascript">
<!--

// This is a comment. It is similar to comments in C++

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

1.5 <noscript> tag in JavaScript

- The <noscript>tag defines an alternate content for users that have disabled scripts in their browser or have a browser that doesn't support script.
- The content inside the <noscript> element will be displayed if scripts are not supported, or are disabled in the user's browser.

```
<html>
<body>
<script language="javascript" type="text/javascript">
    <!--
     document.write ("Hello World!")
    //-->
</script>
<noscript>
    Sorry...JavaScript is not enabled.
</noscript>
</body>
</html>
```

1.6 Embedding JavaScript in HTML

- There is a flexibility given to include JavaScript code anywhere in an HTML document between <script> and </script> tag.
- However the most preferred ways to include JavaScript in an HTML file are as follows:
  - Script in <head>...</head> section.
  - Script in <body>...</body> section.
  - Script in <body>...</body> and <head>...</head> sections.
  - Script in an external file and then include in <head>...</head> section.
- Old JavaScript examples may use a type attribute.
- The type attribute is not required. JavaScript is the default scripting language in HTML.

```
<html>
<head>
<script>
    function sayHello()
    {
        alert("Hello World")
    }
</script>
</head>
<body>
    Click here for the result
    <input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

❖ JavaScript in External File

- As you begin to work more extensively with JavaScript, you will be likely to find that there are cases where you are reusing identical JavaScript code on multiple pages of a site.
- You are not restricted to be maintaining identical code in multiple HTML files. The script tag provides a mechanism to allow you to store JavaScript in an external file and then include it into your HTML files.
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag.
- External JavaScript **Advantages:**
  - It separates HTML and code
  - It makes HTML and JavaScript easier to read and maintain.
  - Cached JavaScript files can speed up page loads.
- We can add several script files to one page. Use several script tags.

```
<script src="filename1.js"></script>
<script src="filename2.js"></script>
```

- External References:
  External scripts can be referenced with a full URL or with a path relative to the current web page

```
<script src="https://www.nareshpd.com.np/js/filename.js"></script>
```

1.7 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 us 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 also defines two trivial data types, **null** and **undefined**, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as **object**.
- **Note**: Java does not make a distinction between integer values and floating-point values. All numbers in JavaScript are represented as floating-point values.

1.8 JavaScript Variables

- Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. We can place data into these containers and then refer to the data simply by naming the container.
- Before we use a variable in a JavaScript program, we must declare it. Variables are declared with the **var** keyword.
- We can also declare multiple variables with the same **var** keyword.
- Storing a value in a variable is called variable initialization. We can do variable initialization at the time of variable creation or at a later point in time when we need that variable.

```
<script>
    var name = "Ram";
    var money;
    money = 2000.50;
</script>
```

- **Note**: Use the **var** keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.
- JavaScript is **untyped** language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.
- 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 variable names are case-sensitive. For example, **Name** and **name** are two different variables.

❖ JavaScript Reserved Words

➢ A list of all the reserved words in JavaScript is 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	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

1.9 Operators in JavaScript

- Java Script includes operators same as other languages. Operators are symbols that are used to perform operations on operands.
- For example, var sum=10+20;
  Here, + is the arithmetic operator and = is the assignment operator.

❖ JavaScript Arithmetic Operators

- Arithmetic operators are used to perform arithmetic between variables and/or values. Given that y=5, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

❖ JavaScript Assignment Operators

- Assignment operators are used to assign values to JavaScript variables. Given that x=10 and y=5, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

❖ The + Operator Used on Strings

- The + operator can also be used to add string variables or text values together. To add two or more string variables together, use the + operator

```
txt1="What a very";
txt2="nice      day";
txt3=txt1+txt2;
```

- After the execution of the statements above, the variable txt3 contains "What a verynice day".

❖ Comparison Operators

- Comparison operators are used in logical statements to determine equality or difference between variables or values. Given that x=5, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

❖ Logical Operators

➢ Logical operators are used to determine the logic between variables or values. Given that x=6 and y=3, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

❖ Bitwise Operators

Name	Operator	Syntax	Example
Bitwise AND	&	output = var1&var2;	Output = 2&3;;
Bitwise OR	|	Output = var1|var2;	Output = 3|2;
Bitwise XOR	^	Output = var1^var2;	Output = 3^2;
Bitwise NOT	~	Output = ~var2;	Output = ~5;
Left Shift	<<	Output = var1<<var2;	Output = 5<<1;
Right Shift	>>	Output = var1>>var2;	Output = 4>>1;

❖ Conditional Operator

variablename=(condition)?value1:value2

➢ JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

❖ The typeof Operator

➢ The typeof operator returns the type of a variable, object, function or expression:

```
typeof "ram"          // Returns string
typeof  3.57          // Returns number
```

❖ The delete Operator

➢ The delete operator deletes a property from an object:

➢
```
var person = {firstName:"John",lastName:"Doe", age:45}
delete person.age;     // or delete person["age"];
```

❖ The in Operator

➢ The in operator returns true if the specified property is in the specified object, otherwise false.

Special Operators

NAME	OPERATOR	DESCRIPTION
Property access	.	Appends an object, method, or property to another object
Array index	[]	Accesses an element of an array
Function call	()	Calls up functions or changes the order in which individual operations in an expression are evaluated
Comma	,	Allows you to include multiple expressions in the same statement
Conditional expression	?:	Executes one of two expressions based on the results of a conditional expression
Delete	delete	Deletes array elements, variables created without the var keyword, and properties of custom objects
Property exists	in	Returns a value of true if a specified property is contained within an object
Object type	instanceof	Returns true if an object is of a specified object type
New object	new	Creates a new instance of a user-defined object type or a predefined JavaScript object type
Data type	typeof	Determines the data type of a variable
Void	void	Evaluates an expression without returning a result

1.10 Control Structure in JavaScript

- JavaScript control Structure is used to control the execution of a program based on a specific condition. If the condition meets then a particular block of action will be executed otherwise it will execute another block of action that satisfies that particular condition.
- **Types of Control Statements in JavaScript**
  - **Conditional Statement:** These statements are used for decision-making; a decision is made by the conditional statement based on an expression that is passed. Either YES or NO.
  - **Iterative Statement:** This is a statement that iterates repeatedly until a condition is met. Simply said, if we have an expression, the statement will keep repeating itself until and unless it is satisfied.
- In JavaScript we have the following conditional statements:
  - **if statement** - use this statement if you want to execute some code only if a specified condition is true
  - **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
  - **if...else if else statement** - use this statement if you want to select one of many blocks of code to be executed
  - **switch statement** - use this statement if you want to select one of many blocks of code to be executed
- In JavaScript we have the following Iterative statement(loops):
  - **for** - loops through a block of code a specified number of times
  - **while** - loops through a block of code while a specified condition is true
  - **do...while**- Loop is a variant of the while loop. This loop will always execute a block of code ONCE, and then it will repeat the loop as long as the specified condition is true. This loop will always be executed at least once, even if the condition is false, because the code is executed before the condition is tested.

1.11 DOM (Document Object Model)

- The Document Object Model (DOM) is a programming interface for HTML (HyperText Markup Language) and XML (Extensible markup language) documents. It defines the logical structure of documents and the way a document is accessed and manipulated.
- DOM is a way to represent the webpage in a structured hierarchical way so that it will become easier for programmers and users to glide through the document. With DOM, we can easily access and manipulate tags, IDs, classes, Attributes, or Elements of HTML using commands or methods provided by the Document object. Using DOM, the JavaScript gets access to HTML as well as CSS of the web page and can also add behavior to the HTML elements.
- According to W3C - "The W3C 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."

❖ Why DOM is required?

- HTML is used to structure the web pages and JavaScript is used to add behavior to our web pages.
- When an HTML file is loaded into the browser, the JavaScript cannot understand the HTML document directly. So it interprets and interacts with the Document Object Model (DOM), which is created by the browser based on the HTML document.
- DOM is basically the representation of the same HTML document but in a tree-like structure composed of objects.
- JavaScript interprets DOM easily, using it as a bridge to access and manipulate the elements i.e JavaScript cannot understand the tags(<h1>H</h1>) in HTML document but can understand object h1 in DOM. Now, Javascript can access each of the objects (h1, p, etc) by using different functions.
- **The Document Object Model (DOM) is essential in web development for several reasons:**
  - **Dynamic Web Pages:** It allows you to create dynamic web pages. It enables the JavaScript to access and manipulate page content, structure, and style dynamically which gives interactive and responsive

web experiences, such as updating content without reloading the entire page or responding to user actions instantly.

- **Interactivity:** With the DOM, you can respond to user actions (like clicks, inputs, or scrolls) and modify the web page accordingly.
- **Content Updates:** When you want to update the content without refreshing the entire page, the DOM enables targeted changes making the web applications more efficient and user-friendly.
- **Cross-Browser Compatibility:** Different browsers may render HTML and CSS in different ways. The DOM provides a standardized way to interact with page elements.
- **Single-Page Applications (SPAs):** Applications built with frameworks such as React or Angular, heavily rely on the DOM for efficient rendering and updating of content within a single HTML page without reloading the full page.

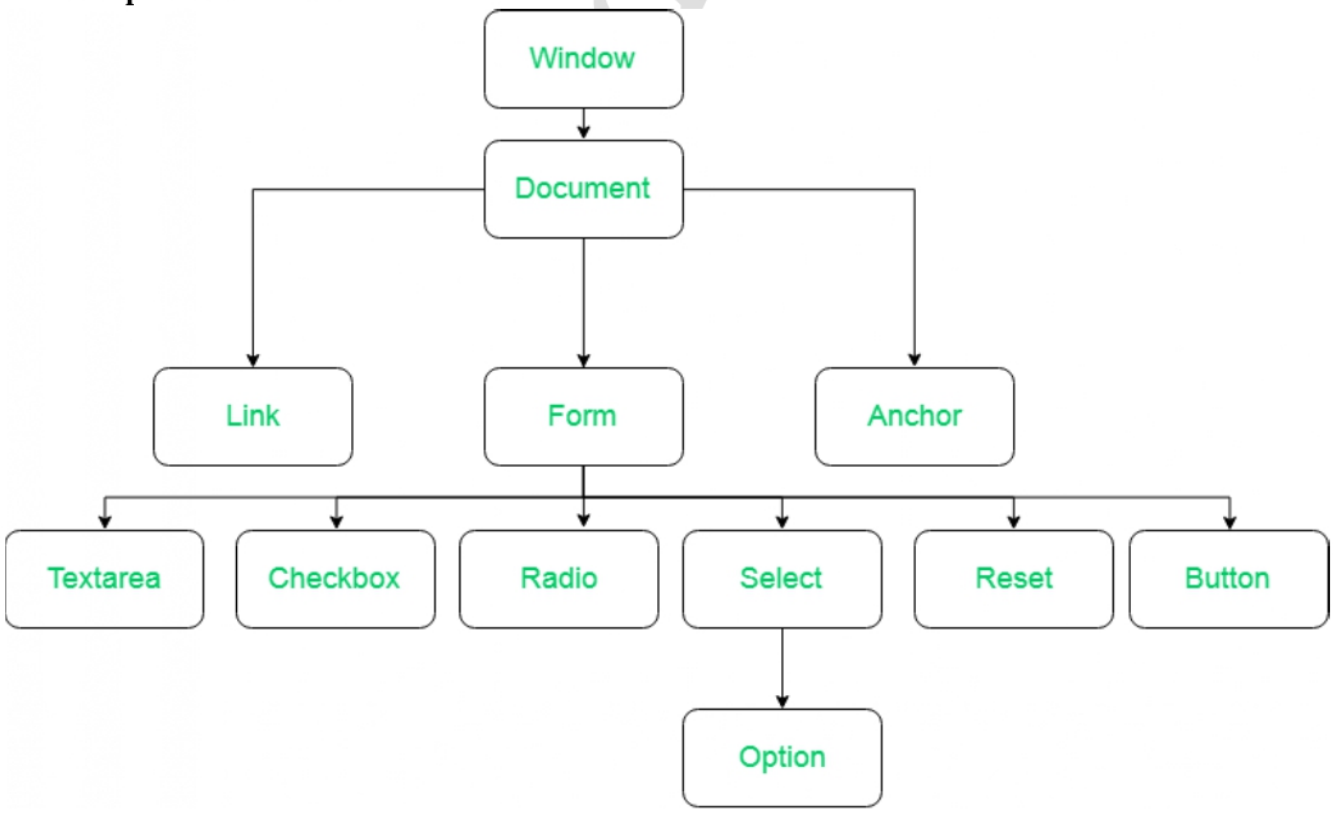
❖ Structure of DOM

- DOM can be thought of as a Tree or Forest (more than one tree). The term structure model is sometimes used to describe the tree-like representation of a document. Each branch of the tree ends in a node, and each node contains objects Event listeners can be added to nodes and triggered on an occurrence of a given event.
- One important property of DOM structure models is structural isomorphism: if any two DOM implementations are used to create a representation of the same document, they will create the same structure model, with precisely the same objects and relationships.

❖ Why DOM is called an Object Model?

- Documents are modeled using objects, and the model includes not only the structure of a document but also the behavior of a document and the objects of which it is composed like tag elements with attributes in HTML.

❖ Properties of DOM

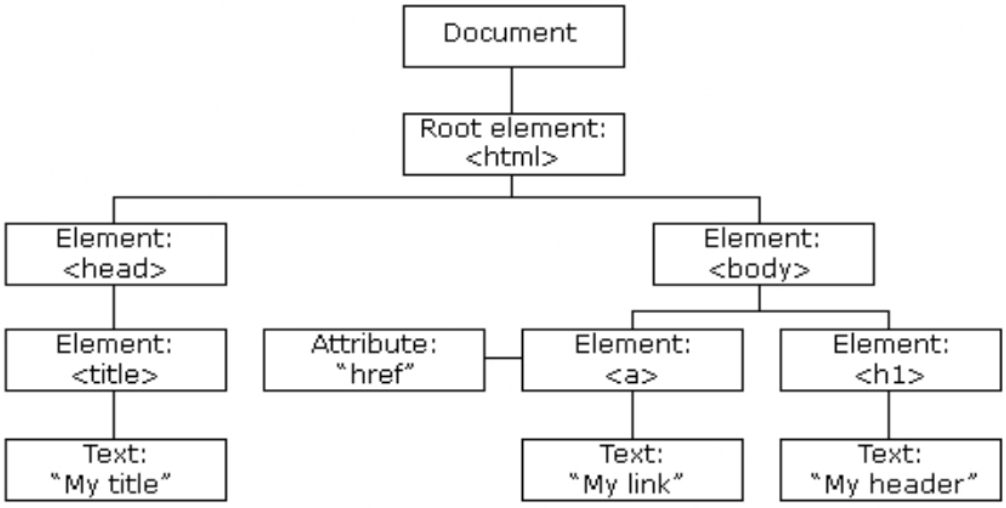


- **Window Object:** Window Object is object of the browser which is always at top of the hierarchy. It is like an API that is used to set and access all the properties and methods of the browser. It is automatically created by the browser.
- **Document object:** When an HTML document is loaded into a window, it becomes a document object. The 'document' object has various properties that refer to other objects which allow access to and modification of the content of the web page. If there is a need to access any element in an HTML page, we always start with accessing the 'document' object. Document object is property of window object.
- **Form Object:** It is represented by form tags.
- **Link Object:** It is represented by link tags.
- **Anchor Object:** It is represented by a href tags.
- **Form Control Elements:** Form can have many control elements such as text fields, buttons, radio buttons, checkboxes, etc.

❖ What is the 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.

❖ 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

- HTML DOM **methods** are actions you can perform (on HTML Elements).
- HTML DOM **properties** are values (of HTML Elements) that you can set or change.

```
<p id="demo"></p>
<script>
        document.getElementById("demo").innerHTML = "Hello Wolrd!";
        var x = document.getElementsByTagName("p");

</script>
```

- **Finding HTML Elements**

Method	Description
document.getElementById(id)	//Find an element by element id
document.getElementsByTagName(name)	// Find element by tag name
Document.getElementsByClassName(name)	//Find element by class name

- **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

- **Adding Events Handlers**

Method	Description
document.getElementById(id).onclick = function(){code}	//Adding event handler code to an onclick event

1.12 JavaScript Functions

- A function (also known as a method) is a self-contained piece of code that performs a particular "function". We can recognize a function by its format - it's a piece of descriptive text, followed by open and close brackets. A function is a reusable code-block that will be executed by an event, or when the function is called.
- To keep the browser from executing a script when the page loads, you can put your script into a function. A function contains code that will be executed by an event or by a call to that function.
- We may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
- Example:

```
<html>
<head>
<script>
    function sayHello()
    {
        alert("Hello World")
    }
</script>
</head>
<body>
    Click here for the result
    <input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

- If the line: alert("Hello world") in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before the user hits the button. We have added an onClick event to the button that will execute the function sayHello() when the button is clicked.
- **Define and Invocation function in Javascript:**
    - JavaScript functions are defined with the function keyword.
    - Function Declarations:

```
function functionName (parameters){
    //code to be executed
}
```

```
var x = function(a,b){return a*b};

var z = x(4,3);

var myFun = function (a,b){return a*b};
var y = myFun(4,3);
```

- A **self-executing function** is a function in JavaScript that doesn't need to be called for its execution it executes itself as soon as it is created in the JavaScript file. This function does not have a name and is also called an **anonymous function**. This function is initialized inside a set of round brackets and the parameters can be passed through the round brackets at the end.

```
(function (){
    alert("Hello anonymous self-invoking function");
})();
```

1.13 JavaScript Objects

- A JavaScript object is an entity having state and behavior (properties and method).
- JavaScript is an object-based language. Everything is an object in JavaScript.
- JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.
- **Creating Objects in JavaScript**
  - By object literal
  - By creating instance of Object directly (using new keyword)
  - By using an object constructor (using new keyword)

❖ JavaScript Object by object literal

- The syntax of creating object using object literal is given below:

```
object={property1:value1,property2:value2.....propertyN:valueN}
```

- Property and value is separated by : (colon).

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

❖ By creating instance of Object

The syntax of creating object directly is given below:

```
var objectname=new Object();
```

Here, **new keyword** is used to create object.

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

❖ By using an Object constructor

- Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.
- The this keyword refers to the current object.

- The example of creating object by object constructor is given below.

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

❖ Defining method in JavaScript Object

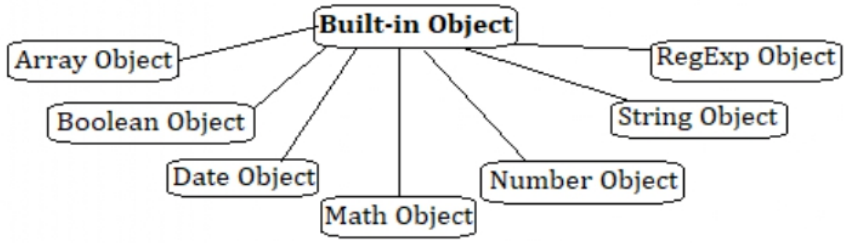
- We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.
- The example of defining method in object is given below.

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;

this.changeSalary=changeSalary;
function changeSalary(otherSalary){
this.salary=otherSalary;
}
}
e=new emp(103,"Sonoo Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changeSalary(45000);
document.write("<br>"+e.id+" "+e.name+" "+e.salary);
</script>
```

1.14 JavaScript Built-In Objects

- JavaScript has several built-in objects that provide various functionalities and capabilities.
- Here are some of the built-in objects in JavaScript:



1.15 Array Object

- The Array object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
- In JavaScript, an array can hold different types of data types in a single slot, which implies that an array can have a string, a number or an object in a single slot.

❖ JavaScript array literal

The syntax of creating array using array literal is given below:

```
var arrayname=[value1,value2.....valueN];
```

As you can see, values are contained inside [ ] and separated by , (comma).

Let's see the simple example of creating and using array in JavaScript.

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

❖ JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

```
var arrayname=new Array();
```

Here, **new keyword** is used to create instance of array.

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

❖ **JavaScript array constructor (new keyword)**

- Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.
- The example of creating object by array constructor is given below.

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

1.15.1 JavaScript Array Properties

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

1.15.2 JavaScript Array Methods

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modified array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops through these keys.

lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the new array
of()	It creates a new array from a variable number of arguments, holding any type of argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
reduce(function, initial)	It executes a provided function for each value from left to right and reduces the array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
toLocaleString()	It returns a string containing all the elements of a specified array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

1.16 Date Object

- At times when user needs to access the current date and tie and also past and future date and times. JavaScript provides support for working with dates and time through the Date Object.
- The Date object provides a system-independent abstraction of dates and times.
- Date object can creates as:

```
var today = new Date();
```

- Dates may be constructed from a year, month, day of the month, hour, minute, and second, and those six components, as well as the day of the week, may be extracted from a date.

- Dates may also be compared and converted to a readable string form. A Date is represented to a precision of one millisecond.
- Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.
- Syntax
  We can use any of the following syntaxes to create a Date object using Date() constructor.

```
new Date( )
new Date(milliseconds)
new Date(datestring)
new Date(year,month,date[,hour,minute,second,millisecond ])
```

- **Note** – Parameters in the brackets are always optional.
- Here is a description of the parameters –
  - **No Argument** – With no arguments, the Date() constructor creates a Date object set to the current date and time.
  - **milliseconds** – When one numeric argument is passed, it is taken as the internal numeric representation of the date in milliseconds, as returned by the getTime() method. For example, passing the argument 5000 creates a date that represents five seconds past midnight on 1/1/70.
  - **datestring** – When one string argument is passed, it is a string representation of a date, in the format accepted by the Date.parse() method.
  - **7 arguments** – To use the last form of the constructor shown above. Here is a description of each argument –
    - **year** – Integer value representing the year. For compatibility (in order to avoid the Y2K problem), you should always specify the year in full; use 1998, rather than 98.
    - **month** – Integer value representing the month, beginning with 0 for January to 11 for December.
    - **date** – Integer value representing the day of the month.
    - **hour** – Integer value representing the hour of the day (24-hour scale).
    - **minute** – Integer value representing the minute segment of a time reading.
    - **second** – Integer value representing the second segment of a time reading.
    - **millisecond** – Integer value representing the millisecond segment of a time reading.

1.16.1 JavaScript Date Method

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.

getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.
getUTCDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time.
getUTCDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time.
getUTCFullYears()	It returns the integer value that represents the year on the basis of universal time.
getUTCHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of universal time.
getUTCMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time.
getUTCMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of universal time.
getUTCSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of universal time.
setDate()	It sets the day value for the specified date on the basis of local time.
setDay()	It sets the particular day of the week on the basis of local time.
setFullYears()	It sets the year value for the specified date on the basis of local time.
setHours()	It sets the hour value for the specified date on the basis of local time.
setMilliseconds()	It sets the millisecond value for the specified date on the basis of local time.

setMinutes()	It sets the minute value for the specified date on the basis of local time.
setMonth()	It sets the month value for the specified date on the basis of local time.
setSeconds()	It sets the second value for the specified date on the basis of local time.
setUTCDate()	It sets the day value for the specified date on the basis of universal time.
setUTCDay()	It sets the particular day of the week on the basis of universal time.
setUTCFullYears()	It sets the year value for the specified date on the basis of universal time.
setUTCHours()	It sets the hour value for the specified date on the basis of universal time.
setUTCMilliseconds()	It sets the millisecond value for the specified date on the basis of universal time.
setUTCMinutes()	It sets the minute value for the specified date on the basis of universal time.
setUTCMonth()	It sets the month value for the specified date on the basis of universal time.
setUTCSeconds()	It sets the second value for the specified date on the basis of universal time.
toDateString()	It returns the date portion of a Date object.
toISOString()	It returns the date in the form ISO format string.
toJSON()	It returns a string representing the Date object. It also serializes the Date object during JSON serialization.
toString()	It returns the date in the form of string.
toTimeString()	It returns the time portion of a Date object.
toUTCString()	It converts the specified date in the form of string using UTC time zone.
valueOf()	It returns the primitive value of a Date object.

1.17 Math Object

- The math object provides you properties and methods for mathematical constants and functions. Unlike other global objects, Math is not a constructor. All the properties and methods of Math are static and can be called by using Math as an object without creating it.
- Thus, we refer to the constant pi as **Math.PI** and we call the sine function as **Math.sin(x)**, where x is the method's argument.

1.17.1 JavaScript Math Methods

Methods	Description
abs()	It returns the absolute value of the given number.
acos()	It returns the arccosine of the given number in radians.
asin()	It returns the arcsine of the given number in radians.
atan()	It returns the arc-tangent of the given number in radians.
cbrt()	It returns the cube root of the given number.
ceil()	It returns a smallest integer value, greater than or equal to the given number.
cos()	It returns the cosine of the given number.
cosh()	It returns the hyperbolic cosine of the given number.
exp()	It returns the exponential form of the given number.
floor()	It returns largest integer value, lower than or equal to the given number.
hypot()	It returns square root of sum of the squares of given numbers.
log()	It returns natural logarithm of a number.
max()	It returns maximum value of the given numbers.
min()	It returns minimum value of the given numbers.

pow()	It returns value of base to the power of exponent.
random()	It returns random number between 0 (inclusive) and 1 (exclusive).
round()	It returns closest integer value of the given number.
sign()	It returns the sign of the given number
sin()	It returns the sine of the given number.
sinh()	It returns the hyperbolic sine of the given number.
sqrt()	It returns the square root of the given number
tan()	It returns the tangent of the given number.
tanh()	It returns the hyperbolic tangent of the given number.
trunc()	It returns an integer part of the given number.

1.18 String Object

- The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.
- As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.
- Syntax
  Use the following syntax to create a String object
- var val = new String(string);
- The String parameter is a series of characters that has been properly encoded.

1.18.1 JavaScript String Methods

Methods	Description
charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.

concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
toLocaleLowerCase()	It converts the given string into lowercase letter on the basis of host?s current locale.
toUpperCase()	It converts the given string into uppercase letter.
toLocaleUpperCase()	It converts the given string into uppercase letter on the basis of host?s current locale.
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.

1.19 RegExp(Regular Expression)

- A regular expression is an object that describes a pattern of characters.
- A regular expression is a sequence of characters that forms a search pattern.
- When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of text search and text replaces operations.
- Syntax:
  new RegExp(pattern, modifier);
  or
  /pattern/modifiers;
- Regular Expression Modifiers can be used to perform multiline searches which can also be set to case-insensitive matching:

Expressions	Descriptions
g	Find the character globally (find all the occurrences of the pattern instead of stopping after the first match)
i	Find a character with case-insensitive matching
m	Find multiline matching

- Regular Expression **Brackets** can Find characters in a specified range

Expressions	Description
[abc]	Find any of the characters inside the brackets
[^abc]	Find any character, not inside the brackets
[0-9]	Find any of the digits between the brackets 0 to 9
[^0-9]	Find any digit not in between the brackets
(x | y)	Find any of the alternatives between x or y separated with |

- Regular Expression **Metacharacters** are characters with a special meaning:

Metacharacter	Description
\.	Search single characters, except line terminator or newline.
\w	Find the word character i.e. characters from a to z, A to Z, 0 to 9
\d	Find a digit
\D	Search non-digit characters i.e all the characters except digits
\s	Find a whitespace character
\S	Find the non-whitespace characters.
\b	Find a match at the beginning or at the end of a word
\B	Find a match that is not present at the beginning or end of a word.
\0	Find the NULL character.
\n	Find the newline character.
\f	Find the form feed character
\r	Find the carriage return character
\t	Find the tab character
\v	Find the vertical tab character

Metacharacter	Description
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxxx

- Regular Expression **Quantifiers** are used to define quantities occurrence

Quantifier	Description
n+	Match any string that contains at least one n
n*	Match any string that contains zero or more occurrences of n
n?	Match any string that contains zero or one occurrence of n
m{X}	Find the match of any string that contains a sequence of m, X times
m{X, Y}	Find the match of any string that contains a sequence of m, X to Y times
m{X,}	Find the match of any string that contains a sequence of m, at least X times
m$	Find the match of any string which contains m at the end of it
^m	Find the match of any string which contains m at the beginning of it
?!m	Find the match of any string which is not followed by a specific string m.

- Regular Expression Object Properties:

Property	Description
constructor	Return the function that created the RegExp object's prototype

Property	Description
global	Specify whether the **"g"** modifier is set or not
ignorecase	Specify whether the **"i"** modifier is set or not
lastindex	Specify the index at which to start the next match
multiline	Specify whether the **"m"** modifier is set or not
source	Return the text of RegExp pattern

- Regular Expression Object **Methods**:

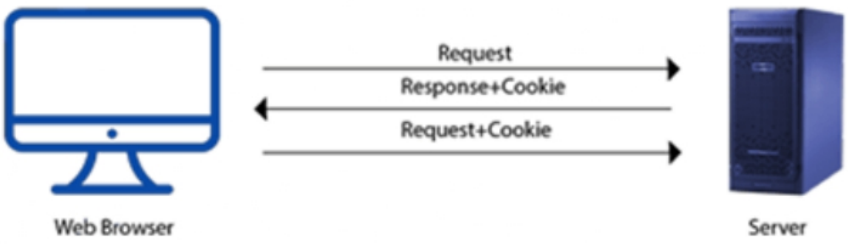
Method	Description
compile()	Used to compile the regular expression while executing of script
exec()	Used to test for the match in a string.
test()	Used to test for a match in a string
toString()	Return the string value of the regular expression

1.20 Cookies

- Web Browsers and Servers use HTTP protocol to communicate and HTTP is a stateless protocol. But for a commercial website, it is required to maintain session information among different pages. For example, one user registration ends after completing many pages. But how to maintain users' session information across all the web pages.
- Cookies are data, stored in small text files, on your computer.
- When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.
- Cookies were invented to solve the problem "how to remember information about the user":
- When a user visits a web page, his/her name can be stored in a cookie.
- Next time the user visits the page, the cookie "remembers" his/her name.
- Cookies are saved in name-value pairs like:
    - username = John Doe

❖ How Cookies Works?

- When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.
- So, to recognize the old user, we need to add the cookie with the response from the server. Browser at the client-side.
- Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users.



❖ How to create a Cookie in JavaScript?

- In JavaScript, we can create, read, update and delete a cookie by using **document.cookie** property.
    - With JavaScript, a cookie can be created like this:
        - document.cookie = "username=John Doe";
    - You can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed:
        - document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";
    - With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.
        - document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";

❖ Read a Cookie with JavaScript

- With JavaScript, cookies can be read like this:
    - let x = document.cookie;

❖ Change a Cookie with JavaScript

- With JavaScript, you can change a cookie the same way as you create it:

- document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
- The old cookie is overwritten.

❖ Delete a Cookie with JavaScript

- Deleting a cookie is very simple.
- We don't have to specify a cookie value when you delete a cookie.
- Just set the expires parameter to a past date:
  - document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

❖ JavaScript Cookie Example

- In the example to follow, we will create a cookie that stores the name of a visitor.
- The first time a visitor arrives to the web page, he/she will be asked to fill in his/her name. The name is then stored in a cookie.
- The next time the visitor arrives at the same page, he/she will get a welcome message.
- For the example we will create 3 JavaScript functions:
  - A function to set a cookie value
  - A function to get a cookie value
  - A function to check a cookie value

```
<html>
<head>
<script>
function setCookie(cname,cvalue,exdays) {
  const d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  let expires = "expires=" + d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
```

```
function getCookie(cname) {
  let name = cname + "=";
  let decodedCookie = decodeURIComponent(document.cookie);
  let ca = decodedCookie.split(';');
  for(let i = 0; i < ca.length; i++) {
    let c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}
```

```
function checkCookie() {
  let user = getCookie("username");
  if (user != "") {
    alert("Welcome again " + user);
  } else {
     user = prompt("Please enter your name:","");
     if (user != "" && user != null) {
       setCookie("username", user, 30);
     }
  }
}
</script>
</head>
<body onload="checkCookie()">
</body>
</html>
```

1.21 Window Object

- The window object in JavaScript stands for the current web page that is being seen in a browser window. It gives access to the browser's methods and attributes as the global object in the browser environment. The window object is the global object in the web browser environment. It represents the current window or frames that the JavaScript code is running in.
- A browser window is represented by the window object. The browser produces one window object for the HTML content and one extra window object for each frame when a document contains frames (iframe> elements).
- The global object of JavaScript in the web browser is the window object. It means that all variables and functions declared globally with the var keyword become the properties and methods of the window object.
- Properties of window object are as follows:

Property Name	Description
window.document	The HTML document that is shown in the window is represented by the Document object, which is referred to by the document property.
window.console	The console gives the window's Console Object back.
window.location	The location attribute makes reference to the Location object,

	which has data about the page's current URL.
window.defaultStatus	It is now Abandoned.
window.closed	If a window is closed, it returns a true boolean value.
window.frameElement	It gives the window's current frame back.
window.frame	returns every window item currently active in the window.
window.history	Retrieves the window's History object.
window.length	It gives the number of iframe> elements currently present in the window.
window.localStorage	provides the ability to store key/value pairs in a web browser. stores data without a time.
window.innerWidth and window.innerHeight	Without including the toolbars and scrollbars, these characteristics describe the width & height of the browser window.
window.opener	It returns a pointer to the window that created the window in the opener function.
window.outerHeight	You can use outerHeight to return the height of the browser window, including toolbars and scrollbars.
window.outerWidth	You can outerWidth to get the width of the browser window, including toolbars and scrollbars.
window.name	Returns or sets a window's name.
window.parent	Brings up the current window's parent window.
window.sessionStorage	Provides the ability to store key/value pairs in a web browser. Contains data for a single session.

window.scrollX	It is a pageXOffset alias.
window.scrollY	It is a pageYOffset alias.
window.self	It provides the window's current state.
window.status	It is now Deprecated. Don't employ it.
window.top	It provides the top-most browser window back.
window.screen	The screen attribute makes reference to the Screen object, which stands in for the screen that the browser is shown on.
window.history	The History object, which includes details about the current page's browsing history, is the subject of the history property.
window.pageXOffset	The number of pixels that the current document has been scrolled horizontally.
window.pageYOffset	The number of pixels that the current document has been scrolled vertically.
window.screenLeft:	The x-coordinate of the current window relative to the screen.
window.screenTop	The y-coordinate of the current window relative to the screen.
window.screenX	The x-coordinate of the current window relative to the screen (deprecated).
window.screenY	The y-coordinate of the current window relative to the screen (deprecated).
window.navigator	An object representing the browser and its capabilities

- Method of window object are as follows:

Property Name	Description
window.open()	This method opens a new browser window or tab.
window.close()	This method closes the current window or tab.
window.alert()	This method displays an alert message to the user.
window.prompt()	This method displays a prompt message to the user and waits for their input.
window.confirm()	This method displays a confirm message to the user and waits for their response.window.focus: brings the current window or tab to the front.
window.blur()	Sends the current window or tab to the back.
window.postMessage()	Sends a message to the window or frame that is targeted by the specified WindowProxy object.
window.scrollTo()	Scrolls the window or frame to a specific position.
window.scrollBy()	Scrolls the window or frame by a specific number of pixels.
window.resizeTo()	Resizes the window to a specific size.
window.resizeBy()	Resizes the window by a specific number of pixels.
window.atob()	A base-64 encoded string is decoded via atob().
window.btoa()	Base-64 encoding is done with btoa().
window.clearInterval()	A timer set with setInterval() is reset.
window.clearTimeout()	The function clearTimeout() resets a timer specified with setTimeout().

window.focus()	It switches the focus to the active window.
window.getComputedStyle()	This function returns the element's current computed CSS styles.
window.getSelection()	It provides a Selection object corresponding to the user-selected text selection range.
window.matchMedia()	The provided CSS media query string is represented by a MediaQueryList object created by the matchMedia() function.
window.moveBy()	Relocates a window with respect to its present location.
window.moveTo()	Relocates a window to the given location.
window.print()	Displays what is currently displayed in the window.
window.requestAnimationFrame()	Before the subsequent repaint, the browser is asked to invoke a function to update an animation using the requestAnimationFrame() method.
window.setInterval()	At predetermined intervals, setInterval() calls a function or evaluates an expression (in milliseconds).
window.setTimeout()	When a certain amount of milliseconds have passed, the setTimeout() method calls a function or evaluates an expression.
window.stop()	It halts the loading of the window.