JavaScript

for Beginners

Urvashi Halpati

⇒ Introduction :

- JavaScript is used to create client-side dynamic pages.
- JavaScript is a lightweight, cross-platform, and interpreted compiled programming language which is also known as the scripting language for webpages.
- Pure JavaScript is also called Vanilla Js.
- It also known as object-based scripting language.
- JavaScript was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser.
- It is well-known for the development of web pages, many non-browser environments also use it.

⇒ History :

- In 1993, Mosaic, the first popular web browser, came into existence. In the year 1994, Netscape was founded by **Marc Andreessen**. He realised that the web needed to become more dynamic content, So he provided to designers and programmers to make easy designing a 'glue language'.
- In 1995, company recruited **Breadan Eich** to intending to implement and embed scheme programming language to the browser.
- In 1995, Marc Andreessen give a first name to JavaScript was **mocha**. Then renamed **LiveScript**. Due to some trademark or other reason name changed to **"JavaScript"**.

- JavaScript is not a compiled language, but it is translated language.
- 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.

⇒ JavaScript features :

- JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
- JavaScript is a lossy type language.
- JavaScript is a object oriented language.
- JavaScript is a case-sensitive language.
- it is written in <script> tag.
- <script> tag is written between <head> or <body> tag.
- Example,

```
<script>
        document.write("Hello world");
</script>
```

- where a Document object represents the HTML document that is displayed in that window. write() is a function to display your content.

JavaScript version

- JavaScript was invented by Brendan Eich in 1995, and become an ECMA standard in 1997.
- ECMAscript is an official name of the language.
- ECMAscript starting editions are known as ES1 to ES6, then it is known as Yearly edition ECMAscript 2016 to as on.

There are three types to add JavaScript in file

- Inline - add using event
- Internal - in <script> tag in <body> or <head> tag
- External - create new file using .js extension and add with <script> tag in html file

(1) inline

Examples,

```
<body>
    <!-- script inline -->
    <button onclick="alert('hello user')">Click here</button>
</body>
```

(2) internal
Example,

```
<head>
    <title>Document</title>
    <!-- script internal -->
    <script>
        document.write("Hello")
    </script>
</head>

<body>
    <script>
        document.write("world")
    </script>
</body>
```

(3) external
Example,

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <!-- link an external file -->
    <script src="script.js"></script>
</head>
```

```
<body>
    <!-- run in both tag -->
    <script src="script.js"></script>
</body>

</html>
```

Js file :

```
script.js
document.write("Hello world");
```

<noscript> tag

- use to know your browser js supported or not.

```
<body>

    <!-- use to know our browser js support or not -->
    <noscript>Your Browser does not support javascript</noscript>
</body>
```

Console

- Console is a command prompt of the browser.

```
<body>
    <!-- console like a command prompt of a browser
        used to know errors of js.
    -->
    <script>
        // we are print a value i.e. string
        console.log("Hello")
        // print number
        console.log(123);
    </script>
</body>
```

Keyword

- Keyword is a reserve word in a programming language.
- For ex,

let	var	const	do	while	for	if	else	switch
case	foreach	of	in	true	false			

Variable

- Variable is a container to store data/value or variable is a name of memory location.
- There are 3 keywords are use to create a variable :
  (1) var (2) let (3) const.
- valid name : myvar, MYVAR, MyVar, myVar, my_var, myvar1, myvar_1
- invalid name : my var, my-var, 1myvar etc
- Syntax :

  Keyword variable_name = value;

  For ex. var a = 5;

- var keyword allowed redeclaration and reassigning value.
- var is a global keyword.
- Example,

```
<script>
        var a = 10;
        document.writeln(a)

        var a = 14;
        document.writeln(a)
</script>
```

- let and const keyword are update in 2015 version.

- let keyword allowed reassigning but not redeclaration.
- let is a block keyword.
  Example,

```
 <script>
        let b = 13;
        document.writeln(b)

        b = 55;
        document.writeln(b)
</script>
```

- const means constant which has fixed value.
- const keyword not allowed redeclaration or reassigning value.
- If use const keyword so it has compulsory to declare value. We can not set undefined value using const.
  Example,

```
<script>
        const c = 12;
        document.writeln(c)

        // c = 14;
        // document.write(c)
</script>
```

Datatype

- There is two type of datatype in javascript.
  (1) Primitive datatype
  (2) Non-primitive(reference) datatype

- Primitive datatype means predefine datatype.

Datatype	Description
String	Represent characters. Ex. “Hello world”
Number	Represent numeric value. Ex. 12 or 12.22
Boolean	Represent boolean value either true or false.
Undefined	Represent undefine value
Null	Represent null means no value at all

- Non-primitive datatype.

Datatype	Description
Object	Represent instance through which we can access members.
Array	Represent group of similar values.
RegExp	Represent regular expression

operator

- Operator means which is operate data.

(1) Arithmetic Operator

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modules

++	Increment
--	Decrement

For example,
```
<body>
    <script>
        let n1 = 20;
        let n2 = 5;

        // here (+) is concatenation operator
      document.write("1st Number is : " + n1 + "<br>2nd Number is : " + n2)
        document.write("<br>---------------------------------")
        document.write("<br>Addition is : " + (n1 + n2))
        document.write("<br>Substraction is : " + (n1 - n2))
        document.write("<br>Multiply is : " + (n1 * n2))
        document.write("<br>Division is : " + (n1 / n2))
        document.write("<br>Modules is : " + (n1 % n2))

    </script>
</body>
```

- Increment = means add 1 value in declared value.
  There is 2 type of increment.
  (1) Pre-increment => for ex, ++a.
  (2) Post-increment => for ex. a++
      Here, a is a variable name.
- Decrement = means remove 1 value from declared value.
  There is 2 type of decrement.
  (1) Pre-decrement => for ex, --a.
  (2) Post-decrement => for ex. a--
      Here, a is a variable name.

For example,
```
<body>
    <script>
        var a = 1;
        console.log(a++); // print than increment  => 1
        console.log(a); // => 2
        console.log(++a); // increment than print => 3
        a++; // => 4
        console.log(a);

        var b = 10;
        console.log(b--); // print than decrement => 10
        console.log(b); // => 9
        console.log(--b); // decrement than print => 8
        --b; // => 7
        console.log(b);
    </script>
</body>
```

(2) Assignment Operator

Operator	Description
=	Assign
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modules and assign

For example,

```
<body>
    <script>
        var a = 10;
        var b = 5;

        a += b  //a = a + b;
        // b = a + b;
        console.log(a); // => 15

        a -= b //a = a - b;
        console.log(a); // => 10

        a *= b //a = a * b
        console.log(a); // => 50

        a /= b
        console.log(a); // => 10

        a %= b
        console.log(a); // => 0
    </script>
</body>
```

(3) Comparison (Relational) Operator

Operator	Description
<	Less than

<=	Less than equal to
>	Greater than
>=	Greater than equal to
==	Is equal to
!=	Is not equal to
===	Is identical equal to
!==	Is not identical equal to

For example,

```
<body>
    <script>
        var x = 50, res;
        console.log("x is : " + x);

        res = (x > 50)
        console.log("(x > 50) : " + res);
        res = (x < 50)
        console.log("(x < 50) : " + res);
        res = (x >= 50)
        console.log("(x >= 50) : " + res);
        res = (x <= 50)
        console.log("(x <= 50) : " + res);
        res = (x == 50)
        console.log("(x == 50) : " + res);
        res = (x != 50)
        console.log("(x != 50) : " + res);

        // check also datatype
        res = (x === "50")
        console.log("(x === 50) : " + res);
        res = (x !== "50")
        console.log("(x !== 50) : " + res);
    </script>
</body>
```

(4) Logical Operator

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

For ex,

```
<body>
    <script>
        var x = 10, res;

        res = (x > 25 && x < 20)
        document.write("(x > 25 && x < 20) : " + res)

        res = (x > 25 || x < 20)
        document.write("<br>(x > 25 || x < 20) : " + res)

        res = !(x > 25 || x < 20)
        document.write("<br>!(x > 25 || x < 20) : " + res)
    </script>
</body>
```

- Remember there is two condition can be compulsory to apply logical condition.

How to logical work :

- If use logical AND

Condition1	Condition2	Result
True	True	True
False	True	False
True	False	False
False	False	False

- If use logical OR

Condition1	Condition2	Result
True	True	True
False	True	True
True	False	True
False	False	False

(5) Bitwise Operator

- Bitwise operator give output of 32 bit number conversion and give js answer.
  For ex,

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Left shift
>>	Right shift

For ex,

```
<body>
    <script>
        var res;
        res = (5 & 3)
        console.log(res); // 1
        res = (5 | 3)
        console.log(res); // 7
        res = (5 ^ 3)
        console.log(res); // 6
        res = ~(10)
        console.log(res); // -11
        res = (5 << 2)
        console.log(res); // 20
        res = (5 >> 2)
        console.log(res); // 1
    </script>
</body>
```

(6) Type operator :

Operator	Description
typeof	Return type of variable
Instanceof	Returns true if an object is instance of an object type

For example,

```
<body>
    <script>
        console.log(typeof "hello User");  // string
        console.log(typeof 123); // number
        console.log(typeof 123.33); // number
        console.log(typeof true); // boolean
        console.log(typeof null); // object
        console.log(typeof a); // undefined
        console.log(typeof [123, 456, 789]); // object
        console.log(typeof { firstname: "urvashi" }); // object
        // The instanceof operator returns true if an
        // object is an instance of the specified object
        var a = [56, 34, 33];
        console.log("================");
        console.log(a instanceof Number); // false
        console.log(a instanceof String); // false
        console.log(a instanceof Array); // true
        console.log(a instanceof Object); // true
    </script>
</body>
```

(7) Concatenation operator :

- It is use to separate variable or string at the time of print value.
- Use (+) symbol

For example,

```
<script>
        console.log("Hello" + "World")
</script>
```

(8) Void operator

- It used to evaluate an expression which does not return any value. It an unary operator that accepts the single operand, which many be of any type.
- It evaluate an expression and return undefined.

For ex,

```
<body>
    <a href="javascript:void(0)">Click Here</a>
    <a
href="javascript:void(document.body.style.backgroundColor='#32CD32');">Click
to change background color</a>
</body>
```

(9) Ternary operator :

- It also known as conditional operator.
- It perform better approach to expressing if condition.

  Syntax:

  Condition ? trueExpression : falseExpression

Conditional statement (Decision Making)

- There are 2 type of Conditional statement.

If statement

- The JavaScript if-else statement is use to execute the code whether condition is true or false.
- There are 3 forms of if statement.

1. If statement
    - It execute when condition/expression is true.
    - Syntax :

```
if(expression){
    statement
}
```

2. If else statement
    - It execute when condition/expression is true otherwise false.
    - In syntax else section always execute if condition are false.
    - Syntax :

```
if(expression){
    statement
}else{
    statement
}
```

3. If else if statement
    - it execute when condition/expression is true from multiple/several expression, otherwise false.
    - Syntax :

```
if(expression 1){
    statement
}else if(expression 2){
    statement
}else if(expression 3){
    statement
}else{
    statement
}
```

4. Nested if / Nested if else statement

Switch statement

- The JavaScript switch statement is used to execute one code from multiple expression.
- Syntax :

```
switch(expression){
    case value1:
        statement;
        break;
    case value2:
        statement;
        break;
    ...
    case valueN:
        statement;
        break;
    default:
        statement;
        break;
}
```

Loop

- Loop is count number of iteration.
- There is 6 type of loop in js.

(1) While loop :

- Js while loop iterates the elements for the infinite number of times.
- It should be used if number of iteration is not known.
- Syntax :

```
while(condition){
    statement;
}
```

(2) Do..while loop :

- The js do while loop iterates the elements for the infinite number of times like while loop.
- It code is executed atleast once whether condition is true or false.
- Syntax :

```
do{
    statement;
}while(condition);
```

(3) For loop :

- The js for loop iterates the elements for the fixed number of times.
- Syntax :

```
for(initialization; condition; increment/decrement){
    statement
}
```

(4) For.. of loop :

- Js for of loop use when array is created.
- This statement loops through the values of an iterable object

- Syntax :

```
for (const iterator of object) {

}
```

(5) For.. in loop :

- Js for in loop use when object is created.
- Syntax :

```
for (const key in object) {
    if (Object.hasOwnProperty.call(object, key)) {
        const element = object[key];
        for (const key in object) {
            statement;
        }
    }
}
```

(6) Foreach loop :

- foreach() method calls a function for each element in an array.
- It not executed for empty elements.
- Syntax :

```
array.forEach(element => {

});
```

Events

- The change in the state of an object is known as an **Event**.
- Js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**.
- Js handles the HTML events via **Event Handlers**.
- There is main 4 type of event :

(1) mouse events :

Event	Event handler	Description
click	onclick	When mouse click on an element
dblclick	ondblclick	When mouse double click on an element.
mouseover	onmouseover	When the cursor of the mouse comes over the element.
mouseout	onmouseout	When the cursor of the mouse leaves an the element.
mousedown	onmousedown	When the mouse button is pressed over the element.

mouseup	onmouseup	When the mouse button is released over the element.
mousemove	onmousemove	When the mouse movement takes place.

(2) keyboard events :

Event	Event handler	Description
keydown	onkeydown	When user press down the key
keypress	onkeypress	When user press down the key
keyup	onkeyup	When user released the key

(3) form events :

Event	Event handler	Description
submit	onsubmit	When user submit the form.
blur	onblur	When element lost the focus from element
change	onchange	When user modifies or changes the value of a form element.
focus	onfocus	When the user focuses on an element.

(4) window events :

Event	Event handler	Description
resize	onresize	When the visitor resize the window.
load	onload	When the browser finishes the loading of the webpage.
scroll	onscroll	When user scroll the window.
unload	onunload	When visitor leaves the current webpage.

Function

- JavaScript functions are used to perform operations.
- We can call JavaScript function several times using make your reusable code.
- Syntax :

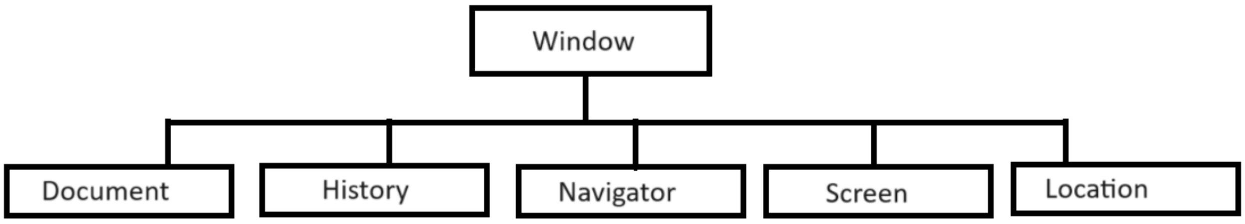
```
function functionName([arg1,arg2,...,argN]){
    statement
}
```

- **JavaScript function Method**:

Method	Description
apply()	It is used to call a function contains this value and a single array of argument.
bind()	It is used to create a new function.
call()	It is used to call function contains this value and an argument list.
toString()	It returns the result in the form of string.

JavaScript BOM

- **Browser Object Model** is used to interact with browser.



Browser objects

(1) Window object

- It represent a window in browser.
- Method of window object :

Method	Description
alert()	Display the alert popup box with containing message with ok button
confirm()	Display the confirm popup box with containing message with ok and cancel button
prompt()	Display the prompt popup box to take input from user.
open()	Open new window
close()	Close window which is open using js.

Alert example :

```
    <button onclick="alertMsg()">show alert</button>
    <script>
        function alertMsg() {
            alert("hello user")
        }
    </script>
```

Confirm example :

```
<button onclick="confirmBox()">Show confirm</button>

<script>
    function confirmBox() {
        var msg = confirm("click on any button")
        if (msg == true) {
            alert("you click on ok")
        } else {
            alert("you click on cancel")
        }
    }
</script>
```

Prompt example :

```
<button onclick="promptBox()">Show prompt</button>

<script>
    function promptBox() {
        var msg = prompt("Enter your name")
        if (msg == "") {
            alert("Enter your name")
        } else {
            alert(msg)
        }
    }
</script>
```

Open and close example :

```
<button onclick="openpage()">open</button>
<button onclick="closepage()">close</button>

<script>
   var openlink = "https://www.google.com/";
   var closelink;

   function openpage() {
       closelink = window.open(openlink, "_blank");
   }
   function closepage() {
       closelink.window.close()
   }
</script>
```

(2) History object

- There are 3 object of history.

Method	Description
go()	loads the given number page
back()	load previous page
forward()	load next page

Example,

```
<button onclick="backFun()">Back</button>
<button onclick="goFun()">Go</button>
<script>
   function backFun() {
     window.history.back()
   }
   function goFun() {
     window.history.go(-2)
   }
</script>
```

Example,

```
<button onclick="forwardFun()">Forward</button>
<script>
    function forwardFun() {
        window.history.forward()
    }
</script>
```

example,

```
<button onclick="getLength()">get length</button>
<hr>
<script>
    function getLength() {
       let len = history.length
       alert(len)
    }
</script>
```

(3) Navigator object

- Used for browser detection.
- Used to get browser information.

Example,

```
  <script>

      console.log(window.navigator.appCodeName);
      console.log(window.navigator.appName);
      console.log(window.navigator.appVersion);
      console.log(window.navigator.userAgent);
      console.log(window.navigator.platform);
      console.log(window.navigator.language);
   </script>
```

- Navigator object method :

Example,

```
<script>
    var a = navigator.javaEnabled()
    console.log(a);
</script>
```

(4) Screen object :

- Used to know information of browser screen.

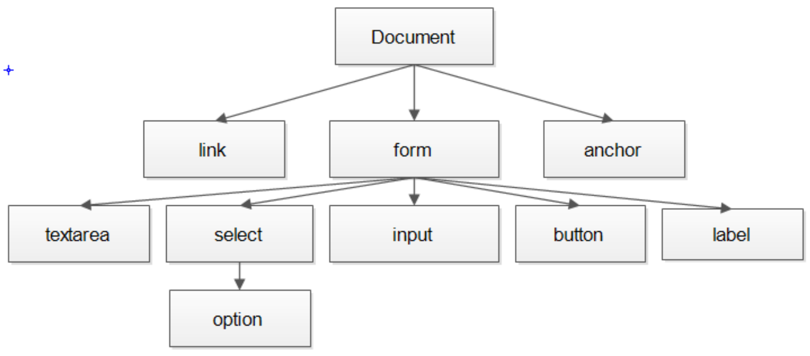
Example,

```
    <script>
        console.log(window.screen.availHeight);
        console.log(window.screen.availWidth);
        console.log(window.screen.height);
        console.log(window.screen.width);
        console.log(window.screen.pixelDepth);
        console.log(window.screen.colorDepth);
    </script>
```

(5) Document object

- Document Object Model (DOM) represents the whole html document.
- When html document is loaded in browser it become a document object.
- It is root element represent html document.
- Using this property and method to we can make dynamic content on webpage.

DOM



Method of document object :

Method	Description
write()	Write string/number in document
writeln()	Write string/number in document with one space.
getElementById()	Element get by element id
getElementByClassName()	Element get by element class name.
getElementByTagName()	Element get by element tag name.
getElementByName()	Element get by element name attribute.

Property of document object :

Property	Description
innerHTML	Property can be used to write dynamic html code in js.
innerText	Property can be used to write dynamic text in js.

JavaScript Selector

(1) getElementById() :

- method return the element of specified id.

Example,

```
<!-- get element by id -->
<p id="p1"></p>
<p id="p2"></p>
<script>
    document.getElementById("p1").innerText = "<b>Hello world</b>";
    document.getElementById("p2").innerHTML = "<b>Hello world</b>";
</script>
```

(2) getElementByClassName() :

- use for getting or selecting the elements through their classname value.

Example,

```
<!-- get element by classname -->
<p class="mypara">example 1</p>
<p class="mypara">example 2</p>
<script>
    var setcolor = document.getElementsByClassName("mypara");
    for (var i = 0; i < setcolor.length; i++) {
        setcolor[i].style.color = "red"
    }
</script>
```

(3) getElementByTagName() :

- return all the element specified tag name.

example,

```
<!-- get element by Tag name -->
<button onclick="showLen()">click here</button>
<div>Div 1</div>
<div>Div 2</div>
<script>
    var tag = document.getElementsByTagName("div")
    function showLen() {
        alert(tag.length)
    }
</script>
```

(4) getElementByName() :

- return all the element specified name.

example,

```
<!-- get element by name -->
<button onclick="showLenName()">click here</button>
select :
<input type="radio" name="gen" id="male"> Male
<input type="radio" name="gen" id="female"> Female
<input type="radio" name="gen" id="other"> Other
<script>
    var element = document.getElementsByName("gen");
    function showLenName() {
        alert(element.length)
    }
</script>
```

(5) querySelector() :

- it is return first element selection.
- Selector : id, class, attribute, element etc.

Example,

```
<!-- query Selector -->
<p id="p3"></p>
<p class="newpara"></p>
<h6></h6>
<script>
    document.querySelector("#p3").innerText = "id selector"
    document.querySelector(".newpara").innerText = "class selector"
    document.querySelector("h6").innerText = "element selector"
</script>
```

(6) querySelectorAll() :

- return all the matching value of the specified css selector or group selector.

example,

```
<!-- query selector all -->
<div class="main">
    <div class="box"></div>
    <div class="box"></div>
    <div class="box"></div>
</div>

<script>
    let main = document.querySelector(".main")
    let boxes = main.querySelectorAll(".box")
    for (var i = 0; i < boxes.length; i++) {
        boxes[i].style.backgroundColor = "pink"
    }
</script>
```

JavaScript ClassList Property

- allows for styling the CSS classes of an element.
- It is read-only property that returns the names of the CSS classes.
- classList return the name of the classes that we have used in the CSS file.
- It return the name of the classes in form of an array.

JavaScript ClassName Property

- It set the class name of the element.
- Use to add css class but it replace existing class.

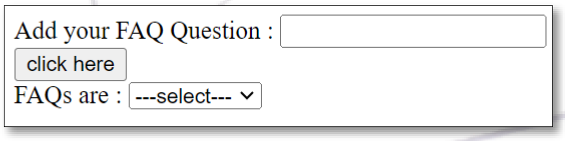
add() method

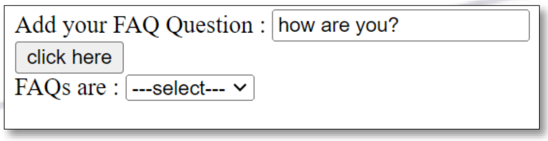
- used to add value.
- You can use to add css or add any element value.

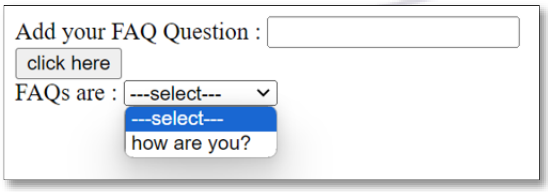
Example : 1, **dynamically add element.**

```
<body>
    <div>
        <label for="">Add your FAQ Question : </label>
        <input type="text" name="" id="faq">

        <div>
            <button onclick="addQues()">click here</button>
        </div>

        <div>
            FAQs are :
            <select name="" id="ques">
                <option value="">---select---</option>
            </select>
        </div>
    </div>

    <script>
        function addQues() {
            let ques = document.getElementById("faq").value;
            let list = document.getElementById("ques");
            let option = document.createElement("option")
            option.text = ques;
            list.add(option);
            document.getElementById("faq").value = "";
        }
    </script>
</body>
```

Output :



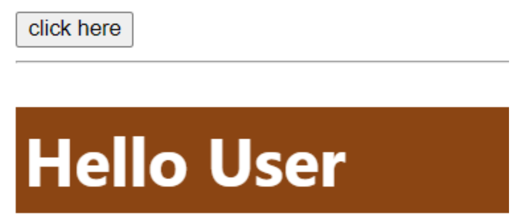




Example : 2, **add css using add method.**

```
<!DOCTYPE html>
<html lang="en">

<head>
    <style>
        .head1 {
            font-size: 40px;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            color: white;
            background-color: saddlebrown;
            padding: 5px;
        }
    </style>
</head>

<body>
    <button onclick="myFunc()">click here</button>
    <hr>
    <h1 class="heading1">Hello User</h1>

    <script>
        function myFunc() {
            let h1 = document.querySelector(".heading1")
            h1.classList.add("head1")
        }
    </script>
</body>

</html>
```

Output :



remove() method

- remove() method are used to remove selected element from browser.
- Also used to remove css classes as per requirement.

Example : 1 **dynamically remove element.**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <style>
        .div1 {
            height: 100px;
            width: 100px;
            background-color: crimson;
        }
    </style>
</head>
<body>
    <button onclick="remVal()">Remove Element</button>
    <hr>
    <div id="div1" class="div1"></div>

    <script>
        function remVal() {
            let div = document.getElementById("div1");
            div.remove()
        }
    </script>
</body>
</html>
```

Example : 2, dynamically remove css from element.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <style>
        .mydiv {
            height: 100px;
            width: 100px;
            border: 1px solid black;
        }

        .divcolor {
            background-color: crimson;
        }
    </style>
</head>
<body>
    <button onclick="remVal()">Remove Element</button>
    <hr>
    <div id="div1" class="mydiv divcolor"></div>

    <script>
        function remVal() {
            let div = document.getElementById("div1");
            div.classList.remove("divcolor")
        }
    </script>
</body>
</html>
```

toggle() method

- Use to add or remove element or css class on webpage.

Example : 1

```
<!DOCTYPE html>
<html lang="en">
<head>
    <style>
        .mydiv {
            height: 100px;
            width: 100px;
            border: 1px solid black;
        }
        .divcolor {
            background-color: crimson;
        }
    </style>
</head>
<body>
    <button onclick="remVal()">Remove Element</button>
    <hr>
    <div id="div1" class="mydiv divcolor"></div>

    <script>
        function remVal() {
            let div = document.getElementById("div1");
            div.addEventListener
            div.classList.toggle("divcolor")
        }
    </script>
</body>
</html>
```

contains() method

- It is used to check item in exist or not in your collection of data/class/element.
- If exist return true otherwise false.

Example,

```
<body>
    <button onclick="checkClass()">check</button>
    <hr>
    <div class="mydiv" id="div1"></div>

    <script>
        function checkClass() {
            let div = document.getElementById("div1");
            if (div.classList.contains("mydiv") == true) {
                alert("class exist")
            } else {
                alert("class not exist")
            }
        }
    </script>
</body>
```

For create element dynamically

- createElement() is used to create element dynamically to HTML element node with specified name via Js.
- This method takes the name of the elements as a parameter and create element node.

- After create element we can use appendChild() and insertBefore() method to insert the created element in the document.
- Use removeChild() to remove node.
- Use replaceChild() to replace node.
- createTextNode() use to add text on that particular element.

Example : 1

```
<body>
    <button onclick="addElementVal()">add list</button>
    <ul>
        <li>List 1</li>
    </ul>
    <script>
        function addElementVal() {
            let list = document.createElement("li");
            let listtext = document.createTextNode("MynewList");
            list.appendChild(listtext);
            document.querySelector("ul").appendChild(list)
        }
    </script>
</body>
```

For set attribute

- setAttribute() method is used to set or add attribute to a particular element.
- If attribute already exist, it is only set or changes value of an attribute.
- We can also add style attribute using this method.
- To get attribute use getAttribute() method.
- To remove attribute use removeAttribute() method.
- createAttribute() method creates an attribute

For example,

```
<body>
    <button onclick="setAttr()">click to set class</button>
    <button onclick="removeAttr()">click to remove class</button>
    <button onclick="getAttr()">click to show image url</button>

    <hr>
    <div>
        <img src="https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQQyD446vRksw83BsovYbnyQGgYdWro8cLX8Q&us
qp=CAU"
            alt="" id="img1">
    </div>
    <div class="">Url : <span id="url"></span></div>
    <script>
        function setAttr() {
            let img = document.getElementById("img1");
            img.setAttribute("class", "image")
        }
        function removeAttr() {
            let img = document.getElementById("img1");
            img.removeAttribute("class", "image")
        }
        function getAttr() {
            let img = document.getElementById("img1");
            let url = img.getAttribute("src");
            document.getElementById("url").innerHTML = url;
        }
    </script>
</body>
```

addEventListner() method

- Used to attach an event handler in particular element.
- Target on direct element.
- Example,

```
<body>
    <button id="btn1">click here</button>

    <script>
        let btn = document.getElementById("btn1");
        btn.addEventListener("click", function () {
            alert("Hello User")
        })
    </script>
</body>
```

removeEventListner() method

- This event remove event / disabled from an element after being click once.
- This event remove click event from element

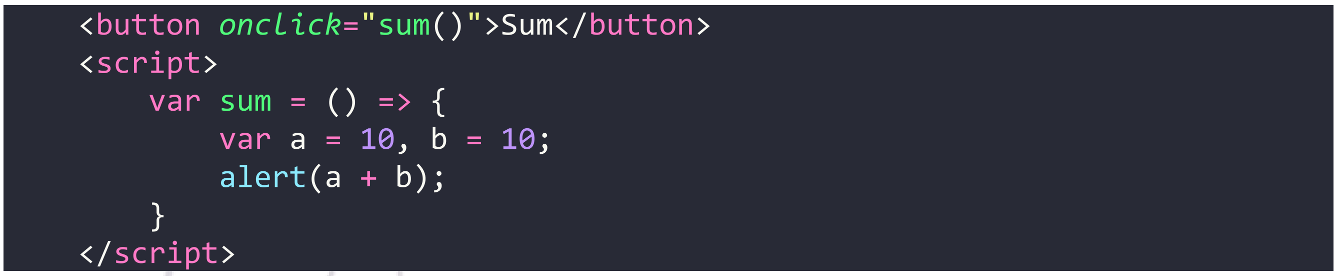
example :

```
<body>
    <button id="btn">click here</button>

    <script>
        let btn = document.getElementById("btn")
        btn.addEventListener("click", msg)
        function msg() {
            alert("remove click access after click on ok")
            btn.removeEventListener("click", msg)
        }
    </script>
</body>
```

arrow function

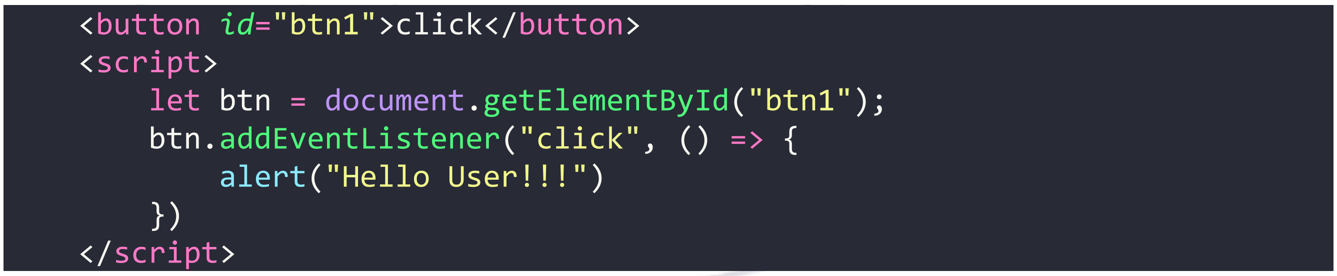
- arrow(=>) function updated in ES6.
- They allows us to write smaller function syntax.
- It makes your code more readable and structured.
- Arrow function are anonymous function. They don't return any value and can declare without the function keyword.
- It cant used as constructor.
- Also called Lambda function in different languages.

Example : 1



```
    <button onclick="sum()">Sum</button>
    <script>
        var sum = () => {
            var a = 10, b = 10;
            alert(a + b);
        }
    </script>
```

Example : 2

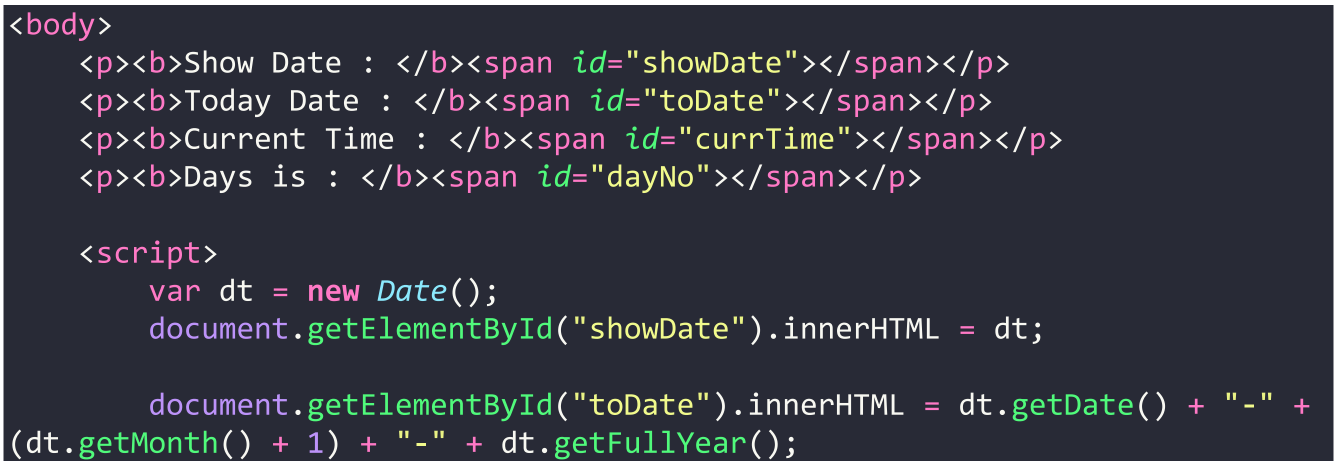


```
    <button id="btn1">click</button>
    <script>
        let btn = document.getElementById("btn1");
        btn.addEventListener("click", () => {
            alert("Hello User!!!")
        })
    </script>
```

Date() function

- JavaScript date object can be used to get year, month and day.
- Use different Date constructor to create date object.
- It provides methods to get and set day, month, year, hour, minute and seconds.

Example : 1



```
<body>
    <p><b>Show Date : </b><span id="showDate"></span></p>
    <p><b>Today Date : </b><span id="toDate"></span></p>
    <p><b>Current Time : </b><span id="currTime"></span></p>
    <p><b>Days is : </b><span id="dayNo"></span></p>

    <script>
        var dt = new Date();
        document.getElementById("showDate").innerHTML = dt;

        document.getElementById("toDate").innerHTML = dt.getDate() + "-" +
(dt.getMonth() + 1) + "-" + dt.getFullYear();
```

```
        document.getElementById("currTime").innerHTML = dt.getHours() + ":"
+ dt.getMinutes() + ":" + dt.getSeconds();

        document.getElementById("dayNo").innerHTML = dt.getDay()
    </script>
</body>
```

setTimeout() & clearTimeout() method

- setTimeout() Use to execute function after waiting for the specified time interval.
- setTimeout() return numeric value that represents the ID value of the timer.
- setTimeout() executes the function only once.
- clearTimeout() method clears a timer set with the setTimeout() method.

Example,

```
<body>
    <p id="p1"></p>
    <script>
        var settime = setTimeout(() => {
            document.getElementById("p1").innerHTML = "Welcome, we learn
JS"
        }, 3000);
        function stopmsg() {
            clearTimeout(settime)
        }
    </script>
</body>
```

setInverval() & clearInterval() method

- setInverval() used to repeat a specified function at every given time-interval.
- setInverval() evaluates an expression or calls a function at given intervals.
- setInverval() invokes the function multiple times.
- setInverval() method can be written with or without the window prefix.
- setInterval() continues the calling of function until the window is closed or the clearInterval() method is called.

example,

```
<body>
    <script>
        var a = setInterval(msg, 3000);
        function msg() {
            alert("Welcome user, to learn js");
            clearInterval(a, 1000)
        }
    </script>
</body>
```

Number Object / Method

- This JavaScript number object enables you to represent a numeric value.
- It may be integer or floating-point.
- Using Number() constructor create number object in JS.
  var n = Number(value);
- If value can't be converted into number, it return NaN(Not a Number) that can be checked by isNaN() method.

- JavaScript Number Method :

Method	Description
isFinite()	It determines whether given value is a finite number.
isInteger()	It determines whether given value is an integer number.
parseFloat()	It converts given string into floating number.
parseInt()	It coverts given string into integer number.
inExponential()	It returns the string represents exponential notation of given number.
toFixed()	It returns the string that represents a number with exact digits after a decimal points.
toPrecision()	It return the string representing a number of specified precision.
toString()	It return given number return in form of String.

Example,

```
<body>
    <script>
        var x = 1 // integer
        console.log(x);
        var y = 10.55 // floating number
        console.log(y);
        var z = 1e4; // exponental number
        console.log(z);
        var n = new Number(16); // integer value by number object
        console.log(n);
        var str = "hello"

        console.log(Number.isFinite(x));
        console.log(Number.isFinite(y));

        console.log(Number.isInteger(x));
        console.log(Number.isInteger(y));
        console.log(Number.parseFloat(x));
        console.log(Number.parseFloat(y));
        console.log(Number.parseFloat(str));

        console.log(Number.parseInt(x));
        console.log(Number.parseInt(y));
        console.log(Number.parseInt(str));

        var exp = 1234567890;
        console.log(exp.toExponential(2));
        console.log(exp.toExponential(5));
        console.log(exp.toExponential(3));

        console.log(y.toFixed());
        console.log(y.toFixed(1));
        console.log(y.toPrecision(2));

        console.log(x.toString());
    </script>
</body>
```

String Object / Method

- Represent sequence of character.
- There 2 ways to creating string :

(1) by string literal

Example,

```
<body>
    <script>
        var str = "hello world";
        console.log(str);
    </script>
</body>
```

(2) by string object

Represent array form

Example,

```
<body>
    <script>
        var str = new String("Hello world");
        console.log(str);
    </script>
</body>
```

- String method

Method	Description
charAt()	Returns the character at the specified index.
charCodeAt()	Returns the Unicode value of the character at the specified location.
concat()	Returns a string that contains the concatenation of two or more strings.
indexOf()	Returns the position of the first occurrence of a substring.
lastIndexOf()	Returns the last occurrence of a substring in the string.
search()	Finds the first substring match in a regular expression search.
match()	Matches a string with a regular expression, and returns an array containing the results of that search.
replace()	Replaces text in a string, using a regular expression or search string.
substr()	Gets a substring beginning at the specified location and having the specified length.
substring()	Returns the substring at the specified location within a string object.
slice()	Return a section of a string.
toLowerCase()	Coverts all the alphabetic characters in a string to lowercase.
toLocalLowerCase()	Converts all alphabetic characters to lowercase, taking into account the host environment's current locale.
toUpperCase()	Coverts all the alphabetic characters in a string to uppercase.
toLocalUpperCase()	Converts all alphabetic characters to uppercase, taking into account the host environment's current locale.
toString()	Return string representation of a string.
valueOf()	Return primitive value of the specified object.
split()	Split a string into substrings using the specified separator and return them as an array.
trim()	Removed the leading and trailing white space and line terminator character from string.

Example,

```
<body>
    <script>
        var str = "Hello world, Hello User";
        console.log(str.charAt(2));
        console.log(str.charCodeAt(0));
        console.log(str.concat(" Welcome"));
        console.log(str.indexOf("Hello"));
        console.log(str.lastIndexOf("Hello"));
        console.log(str.search("l"));
        console.log(str.match("world"));
        console.log(str.replace("Hello", "New"));
        // start, length
        console.log(str.substr(2, 7));
        // start, end
        console.log(str.substring(2, 7));
        console.log(str.slice(2, 7));
        console.log(str.toLowerCase());
        console.log(str.toLocaleLowerCase());
        console.log(str.toUpperCase());
        console.log(str.toLocaleUpperCase());
        console.log(str.toString());
        console.log(str.valueOf());
        console.log(str.split(","));
        var str = "            user            "
        console.log(str.trim());
    </script>
</body>
```

Math Object / Method

- It provides several constant and methods to perform mathematical task/operation.

- Math method :

Method	Description
abs()	Return the absolute value of number.
acos()	Return the arc cosine of a number.
acosh()	Return the inverse hyperbolic cosine of a number.
asin()	Return the arcsine of a number.
asinh()	Returns the inverse hyperbolic sin of a number.
atan()	Returns the arc tangent of a number.
atan2()	Returns the angle (in radians) form the X axis to a point.
atanh()	Returns the inverse hyperbolic tangent of a number.
cbrt()	Returns an implementation-dependent approximation to the cube root of number.
ceil()	Returns the smallest integer greater than or equal to its numeric argument.
cos()	Return the cosine of a number.

cosh()	Return the hyperbolic cosine of a number.
exp()	Returns e (the base of natural logarithms) raised to a power.
floor()	Returns the greatest integer less than or equal to its numeric argument.
fround()	returns the nearest single precision float representation of a number.
hypot()	Returns the square root of the sum of squares of its arguments.
imul()	Returns the result of 32-bit multiplication of two numbers.
log()	Returns the natural logarithm (base e) of a number.
min()	Returns the smallest value from the expression.
max()	Returns the largest value from expression.
pow()	Returns the value of a base expression taken to a specified power.
random()	Returns a pseudo-random number between 0 to 1.
round()	Returns a supplied numeric expression rounded to the nearest integer.
sign()	Returns the sign of the x, indicating whether x is positive, negative or zero.
sin()	Returns the sine of a number.
sinh()	Returns the hyperbolic sine of a number.
sqrt()	Returns the square root of number.
tan()	Returns the tangent of a number.
tanh()	Returns the hyperbolic tangent of a number.
trunc()	Returns the integral part of the a numeric expression, x, removing any fractional digits. If x is already an integer, the result is x.

Example,

```
<body>
    <script>
        console.log(Math.abs(-15));
        console.log(Math.acos(1));
        console.log(Math.acosh(2));
        console.log(Math.asin(1));
        console.log(Math.asinh(1));
        console.log(Math.atan(1));
        console.log(Math.atan2(2, 1));
        console.log(Math.atanh(1));
        console.log(Math.cbrt(27));
        // display largest number
        console.log(Math.ceil(10.33));
        console.log(Math.cos(1));
        console.log(Math.cosh(1));
        console.log(Math.exp(1));
        // display smallest number
        console.log(Math.floor(10.33));
        console.log(Math.fround(12.222));
        console.log(Math.hypot(2, 3));
        console.log(Math.imul(2, 2));
        console.log(Math.log(2));
        console.log(Math.max(3, 6, 8, 2, 9));
        console.log(Math.min(3, 6, 8, 2, 9));
        console.log(Math.pow(2, 3));
        console.log(Math.random());
        console.log(Math.round(12.55));
        console.log(Math.sign(-4));
        console.log(Math.sin(1));
        console.log(Math.sinh(1));
        console.log(Math.sqrt(9));
        console.log(Math.tan(1));
        console.log(Math.tanh(1));
        console.log(Math.trunc(3.22));
    </script>
</body>
```

Array

- Array is a collection of similar type of data.
- It create in 3 ways:

  (1) By Array literal

  - Syntax :

  var arrname = [value1, value2,… valueN]

  (2) By creating instance of Array directly (using new keyword)

  - Syntax :

    var arrayname = new Array();

  (3) By using an Array Constructor (using new keyword)

Example : 1,

By using Array Literal (simple array)

```
<body>
    <script>
        // create an array
        var colors = ["red", "blue", "pink", "magenta", "yellow", "green"];
        console.log(colors);

        // to print single array value
        console.log(colors[0]);
        console.log(colors[3]);
        console.log(colors[2]);
        console.log(colors[4]);
        console.log("-----------------------");
        // to print all values using for of loop
        for (const col of colors) {
            console.log(col);
        }
        console.log("-----------------------");

        // using for loop
        for (var i = 0; i < colors.length; i++) {
            console.log(colors[i]);
        }
        console.log("-----------------------");

        // using foreach loop method
        colors.forEach(clr => {
            console.log(clr);
        });
    </script>
</body>
```

Example : 2,

By creating instance of Array directly (using new keyword)

```
<body>
    <script>
        var myarr = new Array();
        myarr[0] = "red";
        myarr[1] = "green"
        myarr[2] = "white"
        myarr[3] = "black"

        for (let i = 0; i < myarr.length; i++) {
            console.log(myarr[i]);
        }
    </script>
</body>
```

Example: 3,

By using an Array Constructor (using new keyword)

```
<body>
    <script>
        var colors = new Array("Red", "Green", "Blue", "Black", "White")
        colors.forEach(clr => {
            console.log(clr);
        });
    </script>
</body>
```

Object

- Object is an entity which have state and behaviour.
- It has key-pair value.
- Every thing an object in js.
- Js is a templated based not a class based. So we not need to create class for calling object.
- It create in 3 ways:

  (1) By Object literal

  - Syntax :

    objectname = {property1:value; property2:value,…, property:value}

  (2) By creating instance of object (using new keyword)

  - Syntax :

    var objname = new Object();

  (3) By using an Object Constructor (using new keyword)

  - Each argument value can be assigned in the current object by using this keyword.

Example : 1,

By using Object Literal (simple Object value calling)

```
<body>
    <script>
        var chair = {
            color: "Brown",
            size: "40.6 cm - 50.8 cm",
            brand: "cello",
            price: "Rs.500"
        }
        console.log(chair);

        // print single object value
        console.log("Brands : " + chair.brand);
        console.log("Color : " + chair.color);
        console.log("Size : " + chair.size);
        console.log("Price : " + chair.price);
        console.log("----------------------");

        // print all values at a time
        for (const ch in chair) {
            console.log(ch + " -> " + chair[ch]);
        }
        console.log("----------------------");
    </script>
</body>
```

Example : 2,

By creating instance of object.

```
<body>
    <script>
        var chair = new Object();
        chair.id = "1";
        chair.brand = "cello";
        chair.price = "1200"
        console.log(`id : ${chair.id}, brand is : ${chair.brand} and price are :
${chair.price}`);
    </script>
</body>
```

Example : 3,

By using Object Constructor.

```
<body>
    <script>
        function employee(id, name, salary) {
            this.id = id;
            this.name = name;
            this.salary = salary;
        }
        emp = new employee(1, "Abhay", 25000);
        document.write(`id is : ${emp.id}, name is : ${emp.name} and salary is :
${emp.salary}`);
    </script>
</body>
```

Example,

How to implement object in structure.

```
<body>
    <table style="border-collapse: collapse" border="1">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Salary</th>
            </tr>
        </thead>
        <tbody id="tdata"></tbody>
    </table>
    <script>
        let data = document.getElementById("tdata")
        var emp = [
            {
                id: 1,
                name: "Poojan",
                salary: 35000,
            },
            {
                id: 2,
                name: "Raj",
                salary: 30000
            }
        ];
        let i;
        for (i = 0; i < emp.length; i++) {
            data.innerHTML += `
                <tr>
                    <td>${emp[i].id}</td>
                    <td>${emp[i].name}</td>
                    <td>${emp[i].salary}</td>
                </tr>
            `
            console.log(`${emp[i].id} | ${emp[i].name} | ${emp[i].salary}`);
        }
    </script>
</body>
```

this keyword

- It is a reference variable that refers to the current object.

Example,

```
var person = {
    name: "Urvashi",
    number: 8794563201,

    fullDetails: function () {
        return this.name + " " + this.number;
    }
}
var data = person.fullDetails();
console.log(data);
```

Example, variable declare outside of function.

```
var link = "mylink";
function myfunc() {
      document.write(this.link)
}
myfunc();
```

Cookies

- A cookies is an amount of an information which intermediate between server side and client side.
- Browser store information at the time of browsing.
- It contains the information as string.
- String contains key-value pair separated by semi-colons.
- When user send request to server, then each request is treated as new request sent by user.
- To recognize an old user, we need to add the cookie with the response from the server.

- So whatever a user send a request to the server, the cookie is added with that request automatically.
- So due to cookie server recognize the user.

Example, set cookie and get cookie using document.cookie property.

```
<body>
    <button onclick="setCookie()">set cookie</button>
    <button onclick="getCookie()">get cookie</button>
    <script>
        function setCookie() {
            document.cookie = "username = abc";
        }
        function getCookie() {
            if (document.cookie.length != 0) {
                alert(document.cookie)
            } else {
                alert("cookie not available")
            }
        }
    </script>
</body>
```

Cookie attribute :

No	Attribute	Description
1	expires	Maintain state of a cookie up to specified date and time.
2	max-age	Maintain state of a cookie up to specified date and time. Time is given in seconds.
3	path	Extends the scope of the cookie to all the pages of a website.
4	domain	Used to specify the domain for which cookie is valid.

Example,

```
<body>
    <button onclick="setCookie()">set cookie</button>
    <button onclick="getCookie()">get cookie</button>
    <script>
        function setCookie() {
            // document.cookie = "username=urvashi; expires=Sun, 03 Nov 2030
12:00:00 UTC"
            document.cookie = "username=urvashi; max-age=" + (60 * 60 * 24 *
365) + "; path = /;"
        }
        function getCookie() {
            if (document.cookie.length != 0) {
                alert(document.cookie)
            } else {
                alert("cookie not available")
            }
        }
    </script>
</body>
```

Regular Expression

- A regular expression is a pattern of characters.
- Use searching and replacing characters of string.

Regular expression object methods :

No	Method	description
1	test()	Tests for a match in a string. It returns true or false.
2	match()	Returns an array containing all of the matches, including capturing groups, or null if no match is found.
3	matchAll()	Returns an iterator containing all of the matches, including capturing groups.
4	replace()	Executes a search for a match in a string, and replaces the matched substring with a replacement substring.

Modifiers :

No	Expression	Description
1	g	Find character globally.
2	i	Find character with case-insensitive matches
3	m	Find multiline matching
4	d	Find start and end matching (new from ES2022)

Brackets :

No	Expression	Description
1	[abc]	Find any of the characters inside the brackets.
2	[0-9]	Find any of the digits between the brackets.
3	[^abc] or [^0-9]	Find digits / character not inside the brackets.

Metacharacters :

No	Expression	Description
1	\.	Search single characters, except newline
2	\w	Find the word character i.e. character from a-z, A-Z or 0-9
3	\W	Find non-word character.
4	\d	Find a digit.
5	\D	Find non-digit character.
6	\s	Find whitespace character.
7	\S	Find non-whitespace character.
8	\b	Find match at the beginning or at the end of word.
9	\B	Find match that the not present at the beginning or at the end of word.
10	\0	Find the NULL character.
11	\n	Find the newline character.
12	\f	Find the form feed character.
13	\r	Find the carriage return character.
14	\t	Find the tab character.
15	\v	Find a vertical tab character.

Quantifiers :

No	Expression	Description
1	n+	Match any string that contains at least one n
2	n*	Match any string that contains zero or more occurrences of n
3	n?	Match any string that contains zero or one occurrence of n
4	^n	Find the match of any string which contains n at the beginning of it
5	n$	Find the match of any string which contains n at the end of it

Example : 1 using test() method.

```
<body>
    <script>
        let word = "hello";
        let pattern = /^[a-z]*$/;
        if (pattern.test(word)) {
            console.log("match");
        } else {
            console.log("not match");
        }
    </script>
</body>
```

Example : 2 using match() method.

```
<body>
    <script>
        let word = 9865;
        let pattern = /^[0-9]*$/
        if (word.match(pattern)) {
            console.log("match");
        } else {
            console.log("not match");
        }
    </script>
</body>
```

Example : 3 replace() method.

```
<body>
    <script>
        let sentance = "Hello User, We learn Javascript";
        let replaceWord = sentance.replace(/[a]/gi, "000")
        console.log(sentance);
        console.log(replaceWord);
    </script>
</body>
```

Advanced Topic

eval() function

- It used evaluate the string expression.
- It is JS global function.
- It is evaluate the string as JS code and executes it.
- There is some limitation of eval() function, so we can that recommended to used it and it is slower and makes code unreadable.

Example,

```
var a = 10, b = 15, c = 20;
var sum = eval("a + b + c");
console.log(sum);
var minus = eval("a - b - c");
console.log(minus);
```