Introduction to JavaScript

Internet Programming I: Chapter 4 – Part I

ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY

Department of Software Engineering

Main Source: https://www.w3schools.com/js/default.asp

Introduction

- JavaScript is a lightweight, interpreted programming language
- Designed for creating network-centric applications
- Complementary to and integrated with Java
- Complementary to and integrated with HTML
- Open and cross-platform
- Advantages
    - **Less server interaction** – You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server
    - **Immediate feedback to the visitors** – They don't have to wait for a page reload to see if they have forgotten to enter something
    - **Increased interactivity** – You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard
    - **Richer interfaces** – You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors

Adding JavaScript

- In HTML, JavaScript code is inserted between <script> and </script> tags in the <body>, or <head> section of an HTML page, or in both
- JavaScript can also be added external

JavaScript in <head>

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML =
"Paragraph changed.";
}
</script>
</head>
<body>
<h2>Demo JavaScript in Head</h2>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Tr
y it</button>
</body>
</html>
```

JavaScript in <body>

```
<!DOCTYPE html>
<html>
<body>
<h2>Demo JavaScript in Body</h2>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myF
unction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").
innerHTML = "Paragraph changed.";
}
</script>
</body>
</html>
```

Adding JavaScript cont'd

- Scripts can also be placed in external files using **.js** extension
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag in the head or body section of HTML document
- 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
  - To add several script files to one page - use several script tags

Example: adding external JavaScript file

```
<script src="myScript.js"></script>
```

External File: myScript.js

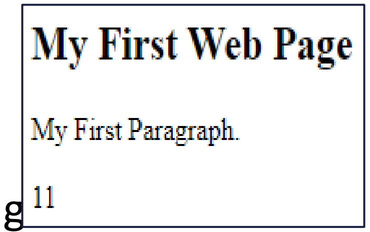
```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

JavaScript Display Possibilities

- JavaScript can "display" data in different ways:
  - Writing into an HTML element, using innerHTML
  - Writing into the HTML output using document.write()
  - Writing into an alert box, using window.alert().
  - Writing into the browser console, using console.log()
- You can call the window.print() method in the browser to print the content of the current window
  - E.g. <button onclick="window.print()">Print this page</button>

Using innerHTML

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Pag
<p>My First Paragraph</p>
<p id="demo"></p>
<script>
document.getElementById("demo").inne
rHTML = 5 + 6;
</script>
….
```



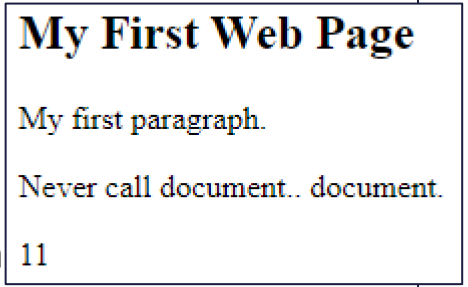
My First Web Page

My First Paragraph.

11

Using document.write()

```
<!DOCTYPE html>
<html>
<body>
<h2>My First Web Page</h
<p>My first paragraph.</p>
<p>Never call document.. document.</p>
<script>
document.write(5 + 6);
</script>..
```



My First Web Page

My first paragraph.

Never call document.. document.

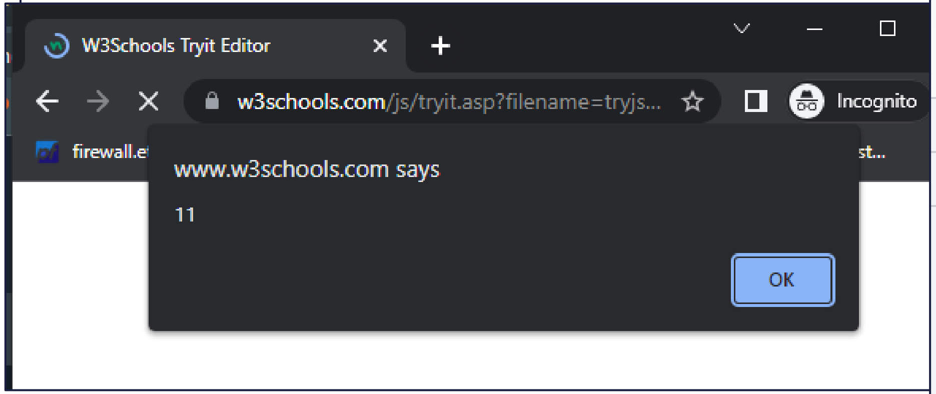
11

JavaScript Display Possibilities cont’d

Using window.alert()

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

</body>
```



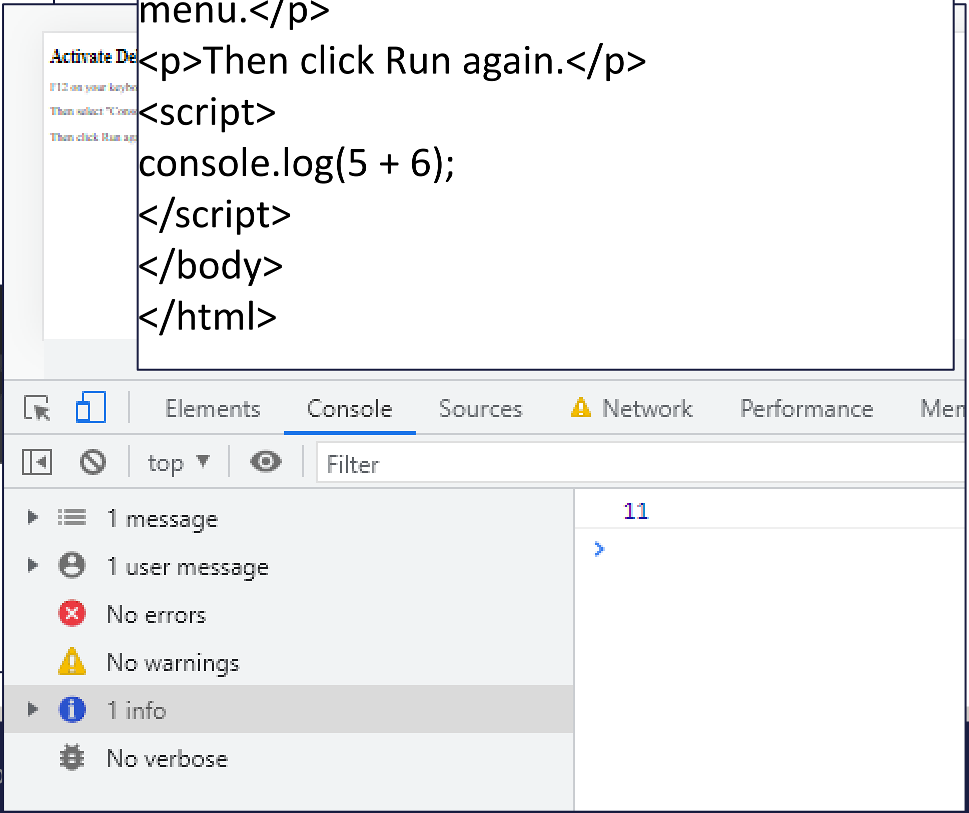
W3Schools Tryit Editor

w3schools.com/js/tryit.asp?filename=tryjs...

```
www.w3schools.com says
11
```

Using console.log()

```
<!DOCTYPE html>
<html>
<body>
<h2>Activate Debugging</h2>
<p>F12 on your keyboard will activate debugging.</p>
<p>Then select "Console" in the debugger menu.</p>
<p>Then click Run again.</p>
<script>
console.log(5 + 6);
</script>
</body>
</html>
```



```
1 message
1 user message
No errors
No warnings
1 info
No verbose

11
```

JavaScript Comments

- Single Line Comments (//)

```
// Change heading:
dgetElementById("myH").innerHTML = "My First Page";

// Change paragraph:
dgetElementById("myP").innerHTML = "My first paragraph.";
```

- Multi-line Comments (/*..*/)

```
/*
The code below will change the heading with id = "myH"
and the paragraph with id = "myP“ in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

JavaScript Variables

- JavaScript variables can be declared using var, let, const, or nothing
- If you want a general rule: always declare variables with const
- If you think the value of the variable can change, use let
- variables defined with let cannot be redeclared

Variable using var

```
var x = 5;
var y = 6;
var z = x + y;
```

Variable using let

```
let x = 5;
let y = 6;
let z = x + y;
```

Variable using nothing

```
x = 5;
y = 6;
z = x + y;
```

With let you can not do this:

```
let x = "John Doe";
let x = 0;
// SyntaxError: 'x' has already been declared
```

With var you can:

```
var x = "John Doe";
var x = 0;
```

Variable using const

```
const price1 = 5;
const price2 = 6;
let total = price1 + price2;
```

JavaScript Variables cont'd

Variables defined with let have Block Scope

```
{
  let x = 2;
}
// x can NOT be used here
```

Variables declared with the var keyword can NOT have block scope

```
{
  var x = 2;
}
// x CAN be used here
```

Redeclaring a variable using the var keyword can impose problems

```
var x = 10;
// Here x is 10
{
  var x = 2;
  // Here x is 2
}
// Here x is 2
```

Redeclaring a variable using let keyword inside a block will not redeclare the variable outside the block:

```
let x = 10;
// Here x is 10
{
let x = 2;
// Here x is 2
}
// Here x is 10
```

A const variable cannot be reassigned:

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

JavaScript const variables must be assigned a value when they are declared:

Correct

```
const PI = 3.14159265359;
```

Incorrect

```
const PI;
PI = 3.14159265359;
```

JavaScript Operators

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

Assignment Operators

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

JavaScript Operators

Shift Assignment Operators

Operator	**Example**	**Same As**
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y

Bitwise Assignment Operators

Operator	**Example**	**Same As**
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y

Logical Assignment Operators

Operator	Example	Same As
&&=	x &&= y	x = x && (x = y)
||=	x ||= y	x = x || (x = y)
??=	x ??= y	x = x ?? (x = y)

JavaScript Datatypes

Data Type	**Example**

String	`let color = "Yellow";`
	`let lastName = "Johnson";`

Number	let length = 16;
	let weight = 7.5;

Booleans	`let x = true;`
	`let y = false;`

Object	`const person = {firstName:"John", lastName:"Doe"};`
	`// Array object:`
	`const cars = ["Saab", "Volvo", "BMW"];`
	`// Date object:`
	`const date = new Date("2022-03-25");`

Bigint	let x = BigInt("123456789012345678901234567890");

Boolean	let x = 5;
	let y = 5;
	let z = 6;
	(x == y) // Returns true
	(x == z) // Returns false

Undefined	Example let car; // Value is undefined, type is undefined

JavaScript Functions

- JavaScript functions are used to perform operations
- Main advantage of function is Code reusability, Less coding etc..
- Syntax

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

        //code to be executed
}
```

Example

```
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()"
 value="call function"/>
```

Example with argument

```
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcu
be(4)"/>
</form>
```

Example with Return Value

```
<script>
function getInfo(){
return "hello javatpoint! How r u?";
}
</script>
<script>
document.write(getInfo());
</script>
```

JavaScript If-else

- There are three forms of if statement in JavaScript.
  - If Statement, If else statement and if else if statement

If statement

Syntax

```
if(expression){
   //content to be evaluated
}
```

Example

```
<script>
var a=20;
if(a>10){
document.write("value of a i
s greater than 10");
}
</script>
```

If...else Statement

Syntax

```
if(expression){
//content to be evaluated if condition is true
}  else{
//content to be evaluated if condition is false
}
```

Example

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}  else{
document.write("a is odd number");
}
</script>
```

JavaScript If-else cont’d

JavaScript If...else if statement Syntax

```
if(expression1){
//content to be evaluated if expre
ssion1 is true
} else if(expression2){
//content to be evaluated if expre
ssion2 is true
} else if(expression3){
//content to be evaluated if expre
ssion3 is true
} else{
//content to be evaluated if no ex
pression is true
}
```

JavaScript If...else if statement Example

```
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
} else if(a==15){
document.write("a is equal to 15");
} else if(a==20){
document.write("a is equal to 20");
} else{
document.write("a is not equal to 10
, 15 or 20");
}
</script>
```

JavaScript Switch

Syntax

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

Example

```
<script>
var grade='B';
var result;
switch(grade){
case 'A':
  result="A Grade";
  break;
case 'B':
  result="B Grade";
  break;
case 'C':
  result="C Grade";
  break;
default:
  result="No Grade";
}
document.write(result);
</script>
```

JavaScript Loops

- There are four types of loops in JavaScript
  - for loop, while loop, do-while loop, for-in loop

For loop

Syntax

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

Example

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

while loop

Syntax

while (condition) { code to be executed }

Example

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

do while loop

Syntax

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

Example

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

JavaScript Loops cont’d

For In Loop Over Objects

Syntax

```
for (key in object) {
 // code block to be executed
}
```

Example

```
const person = {fname:"John",  lname:"Doe", age:25};
let text = "";
for (let x in person) {
 text += person[x];
}
```

For In Over Arrays

Syntax

```
for (variable in array) {
  code
}
```

Example

```
const numbers = [45, 4, 9, 16, 25];
let txt = "";
for (let x in numbers) {
  txt += numbers[x];
}
```

JavaScript Loops cont’d

Array.forEach()
The forEach() method calls a function (a callback function) once for each array element

Example

```
const numbers = [45, 4, 9, 16, 25];

let txt = "";
numbers.forEach(myFunction);

function myFunction (value, index, array) {
  txt += value;
}
```

Example using value only

```
const numbers =
[45, 4, 9, 16, 25];

let txt = "";

numbers.forEach(myFunction);

function myFunction(value) {
  txt += value;
}
```

JavaScript Loops cont'd

- **The For Of Loop:** loops through the values of an iterable object
  - It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more...

Syntax

```
for (variable of iterable) {
  // code block to be executed
}
```

Example: Looping over an Array

```
const cars =
["BMW", "Volvo", "Mini"];

let text = "";
for (let x of cars) {
  text += x;
}
```

Example: Looping over a String

```
let language = "JavaScript";

let text = "";
for (let x of language) {
text += x;
}
```

JavaScript Objects

- A javaScript object is an entity having state and behavior (properties and method
- 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

Syntax

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

Example

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

By creating instance of Object

Syntax

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

Example

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

JavaScript Objects cont’d

- **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

Example

```
<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>
```

JavaScript Object Methods

Methods	**Description**
Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
Object.create()	This method is used to create a new object with the specified prototype object and properties
Object.defineProperty()	This method is used to describe some behavioral attributes of the property.
Object.defineProperties()	This method is used to create or configure multiple object properties.
Object.entries()	This method returns an array with arrays of the key, value pairs.
Object.freeze()	This method prevents existing properties from being removed.
Object.getOwnPropertyDescriptor()	This method returns a property descriptor for the specified property of the specified object.
Object.getOwnPropertyDescriptors()	This method returns all own property descriptors of a given object.
Object.getOwnPropertyNames()	This method returns an array of all properties (enumerable or not) found

JavaScript Object Methods cont'd

Methods	Description
Object.getOwnPropertySymbols()	This method returns an array of all own symbol key properties
Object.getPrototypeOf()	This method returns the prototype of the specified object
Object.is()	This method determines whether two values are the same value
Object.isExtensible()	This method determines if an object is extensible
Object.isFrozen()	This method determines if an object was frozen
Object.isSealed()	This method determines if an object is sealed
Object.keys()	This method returns an array of a given object's own property names
Object.preventExtensions()	This method is used to prevent any extensions of an object
Object.seal()	This method prevents new properties from being added and marks all existing properties as non-configurable.
Object.setPrototypeOf()	This method sets the prototype of a specified object to another object
Object.values()	This method returns an array of values.

JavaScript Array

- There are 3 ways to construct array in JavaScript
  - By array literal,
  - By creating instance of Array directly (using new keyword), and
  - By using an Array constructor (using new keyword)

JavaScript array literal

Syntax

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

Example

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

JavaScript Array directly (new keyword)

Syntax

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

Example

```
<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)

Example

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

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.

JavaScript Array Methods cont'd

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.

JavaScript String

- There are 2 ways to create string in JavaScript
  - By string literal
  - By string object (using new keyword)

String literal (using double qoutes)

Syntax

```
var stringname="string value";
```

Example

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

By string object (using new keyword)

Syntax

```
var stringname=new String("string literal");
```

Example

```
<script>
var stringname=new String("hello javascript string");
 document.write(stringname);
</script>
```

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.

JavaScript String Methods cont'd

Methods	Description
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.

JavaScript String Methods example

JavaScript String indexOf(str) Method

```
<script>
var s1="javascript from javatpoint indexof";
var n=s1.indexOf("from");
document.write(n);
</script>
```

JavaScript String charAt(index) Method

```
<script>
var str="javascript";
document.write(str.charAt(2));
</script>  v
```

JavaScript String lastIndexOf(str) Method

```
<script>
var s1="javascript from javatpoint indexof";
var n=s1.lastIndexOf("java");
document.write(n);
</script>
```

JavaScript String concat(str) Method

```
<script>
var s1="javascript ";
var s2="concat example";
var s3=s1.concat(s2);
document.write(s3);
</script>
```

JavaScript Date Object

- You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.
  - Date()
  - Date(milliseconds)
  - Date(dateString)
  - Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript Date Example

Example

```
Current Date and Time: <span id="txt"></span>
<script>
var today=new Date();
document.getElementById('txt').innerHTML=today;
</script>
```

Output:

```
Current Date and Time: Sat Jan 28 2023 16:54:12 GMT+0300 (East
Africa Time)
```

Example: print date/month/year

```
<script>
var date=new Date();
var day=date.getDate();
var month=date.getMonth()+1;
var year=date.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+year);
</script>
```

Output:

```
Date is: 28/1/2023
```

Example: Current Time

```
Current Time: <span id="txt"></span>
<script>
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
document.getElementById('txt').innerHT
ML=h+":"+m+":"+s;
</script>
```

Output:

```
Current Time: 16:54:12
```

JavaScript Date Methods

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.

JavaScript Date Methods cont'd

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.

JavaScript Math

- The JavaScript math object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors
- **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.

JavaScript Math Methods

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.

JavaScript Math example

Example Math.sqrt(n)

```
Square Root of 17 is: <span id="p1"></span>
<script>
document.getElementById('p1').innerHTML=Math.sqrt(17);
 </script>
```

Output:

```
Square Root of 17 is: 4.123105625617661
```

Example Math.random()

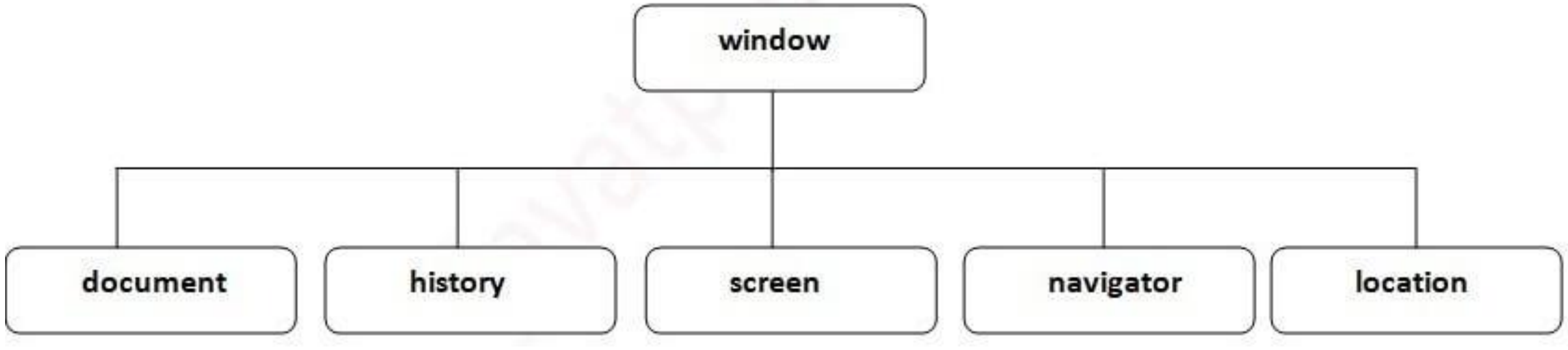
```
Random Number is: <span id="p2"></span>
<script>
document.getElementById('p2').innerHTML=Math.random();
</script>
```

Output:

```
Random Number is: 0.928270942604742
```

Browser Object Model (BOM)

- The Browser Object Model (BOM) is used to interact with the browser
- The default object of browser is window means you can call all the functions of window by specifying window or directly.
- For example: **window.alert("hello javatpoint");** is same as: **alert("hello javatpoint");**
- You can use a lot of properties (other objects) defined underneath the window object like:
  - document, history, screen, navigator, location, innerHeight, innerWidth



Window Object

- The window object represents a window in browser. An object of window is created automatically by the browser

Window Object Methods

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

Window Object example

Example of alert() in javascript

```
<script type="text/javascript">
function msg(){
 alert("Hello Alert Box");
}
</script>
<input  type="button"  value="click"
onclick="msg()"/>
```

Example of confirm() in javascript

```
<script type="text/javascript">
function msg(){
 var v= confirm("Are u sure?");
 if(v==true){  alert("ok");
 }  else {  alert("cancel");  }
}
</script>
<input  type="button"  value="delet
e record" onclick="msg()"/>
```

Example of prompt() in javascript

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
alert("I am "+v);
}
</script>
<input  type="button"  value="click"
onclick="msg()"/>
```

Example of setTimeout() in javascript

```
<script type="text/javascript">
function msg(){
setTimeout(function(){  alert("Welcome after 2 seconds") },2000);
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

JavaScript History Object

- The JavaScript history object represents an array of URLs visited by the user
- By using this object, you can load previous, forward or any particular page
- Can be accessed by: window.history Or history
- Methods of JavaScript history object include forward(), back(), go()

Example of history object

```
history.back();//for previous page
history.forward();//for next page
history.go(2);//for next 2nd page
history.go(-2);//for previous 2nd page
```

JavaScript Navigator Object

- The JavaScript navigator object is used for browser detection
- It can be used to get browser information such as appName, appCodeName, userAgent etc.
- It can be accessed by: window.navigator   Or navigator

Property of JavaScript navigator object

Property	Description
appName	returns the name
appVersion	returns the version
appCodeName	returns the code name
cookieEnabled	returns true if cookie is enabled otherwise false
userAgent	returns the user agent
language	returns the language. It is supported in Netscape and Firefox only.
userLanguage	returns the user language. It is supported in IE only.
plugins	returns the plugins. It is supported in Netscape and Firefox only.
systemLanguage	returns the system language. It is supported in IE only.
mimeTypes[]	returns the array of mime type. It is supported in Netscape and Firefox only.
platform	returns the platform e.g. Win32.
online	returns true if browser is online otherwise false.

JavaScript Navigator Object example

AASTU

```
<script>
document.writeln("<br/>navigator.appCodeName: "+navigator.appCodeName);
document.writeln("<br/>navigator.appName: "+navigator.appName);
document.writeln("<br/>navigator.appVersion: "+navigator.appVersion);
document.writeln("<br/>navigator.cookieEnabled: "+navigator.cookieEnabled);
document.writeln("<br/>navigator.language: "+navigator.language);
document.writeln("<br/>navigator.userAgent: "+navigator.userAgent);
document.writeln("<br/>navigator.platform: "+navigator.platform);
document.writeln("<br/>navigator.onLine: "+navigator.onLine);
</script>
```

Out put

```
navigator.appCodeName: Mozilla
navigator.appName: Netscape
navigator.appVersion: 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/109.0.0.0 Safari/537.36
navigator.cookieEnabled: true
navigator.language: en-US
navigator.userAgent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/109.0.0.0 Safari/537.36
navigator.platform: Win32
navigator.onLine: true
```

JavaScript Screen Object

- The JavaScript screen object holds information of browser screen. It can be used to display screen width, height, colorDepth, pixelDepth etc.
- It can be accessed by: **window.screen** Or **screen**

Property of JavaScript Screen Object

Property	Description
width	returns the width of the screen
height	returns the height of the screen
availWidth	returns the available width
availHeight	returns the available height
colorDepth	returns the color depth
pixelDepth	returns the pixel depth.

JavaScript Screen Object example

```
<script>
document.writeln("<br/>screen.width: "+screen.width);
document.writeln("<br/>screen.height: "+screen.height);
document.writeln("<br/>screen.availWidth: "+screen.availWidth);
document.writeln("<br/>screen.availHeight:  "+screen.availHeight);
document.writeln("<br/>screen.colorDepth: "+screen.colorDepth);
document.writeln("<br/>screen.pixelDepth: "+screen.pixelDepth);
</script>
```

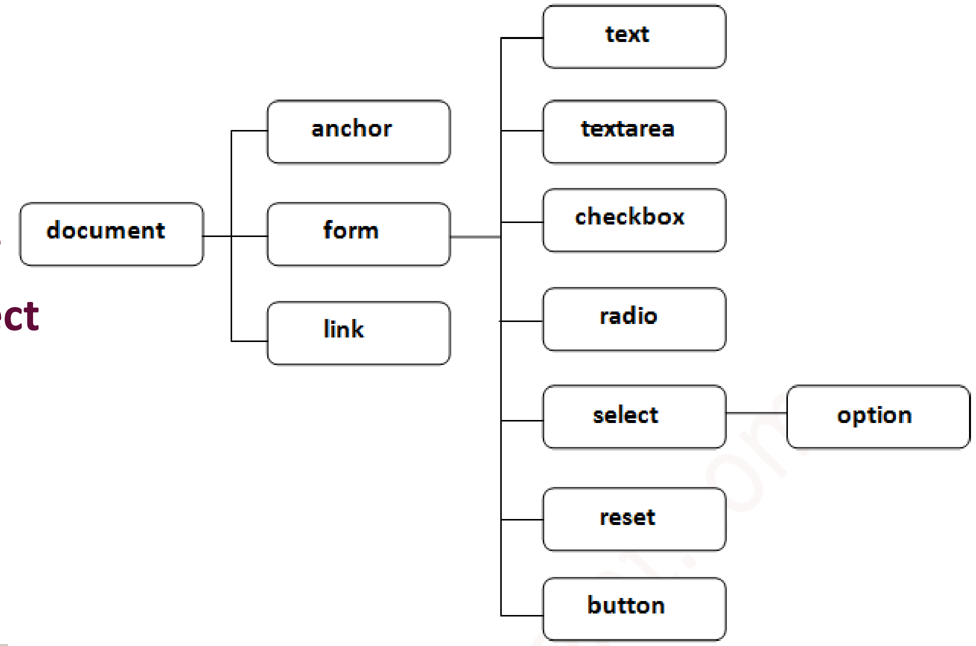
```
screen.width: 1366
screen.height: 768
screen.availWidth: 1366
screen.availHeight: 728
screen.colorDepth: 24
screen.pixelDepth: 24
```

Output

Document Object Model (DOM)

- The document object represents the whole html document
- When html document is loaded in the browser, it becomes a document object
- It is the root element that represents the html document
- It has properties and methods. By the help of document object, we can add dynamic content to our web page
- It can be accessed by: **window.document** or **document**

Properties of document object



Methods of document object

- We can access and change the contents of document by its methods

Method	Description
write("string")	writes the given string on the document
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name

Accessing value by document object

- In this example, we are going to get the value of input text by user
- Here, we are **using document.form1.name.value** to get the value of name field
  - Here, **document** is the root element that represents the html document
  - **form1** is the name of the form
  - **name** is the attribute name of the input text
  - **value** is the property, that returns the value of the input text

```
<script type="text/javascript">
function printvalue(){
 var name=document.form1.name.value;
 alert("Welcome: "+name);  }
</script>

<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

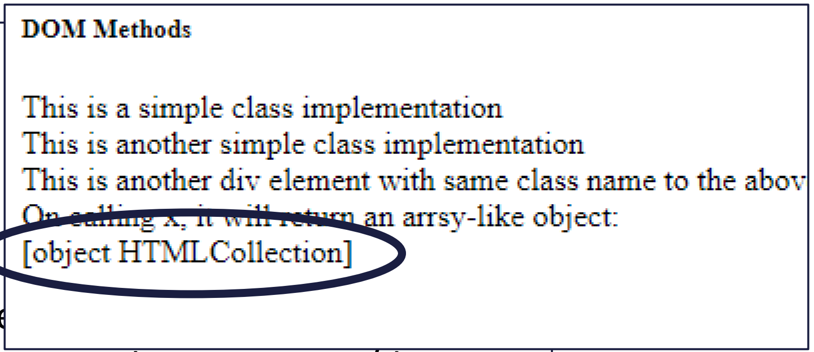
document.getElementById() method

```
<script type="text/javascript">
function getcube(){
 var number=document.getElementById("number").value;
 alert(number*number*number);
}
</script>
<form>
Enter No:<input type="text" id="number" name="number"/><br/>
<input type="button" value="cube" onclick="getcube()"/>  </form>
```

GetElementsByClassName()

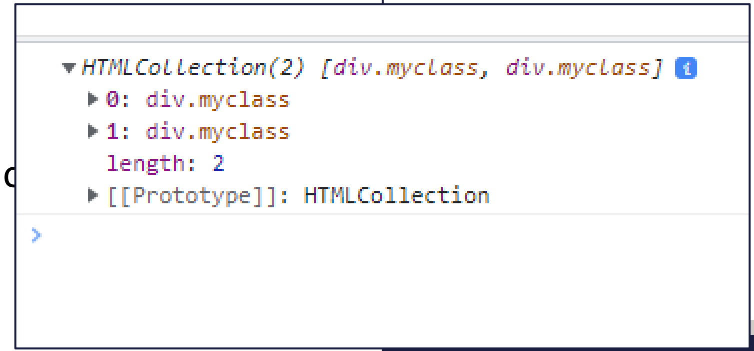
- The getElementsByClassName() method is used for selecting or getting the elements through their class name value
- This DOM method returns an array-like object that consists of all the elements having the specified classname
- Syntax: var ele=document.getELementsByClassName('name');

```
<html>
<head>
  <h5>DOM Methods </h5>
</head>
<body>
  <div class="myClass"> This is a simple class imple
  <div class="yourClass"> This is another simple class implementation</div>
  <div class="myClass">
    This is another div element with same class name to the above myClass
  </div>
  <script type="text/javascript">
    var x = document.getElementsByClassName('myClass');
    document.write("On calling x, it will return an arrsy-like c
    console.log(x)
  </script>
</body>
</html>
```



```
DOM Methods

This is a simple class implementation
This is another simple class implementation
This is another div element with same class name to the abov
On calling x, it will return an arrsy-like object:
[object HTMLCollection]
```



```
▾HTMLCollection(2) [div.myclass, div.myclass]
  ▸0: div.myclass
  ▸1: div.myclass
   length: 2
  ▸[[Prototype]]: HTMLCollection
>
```

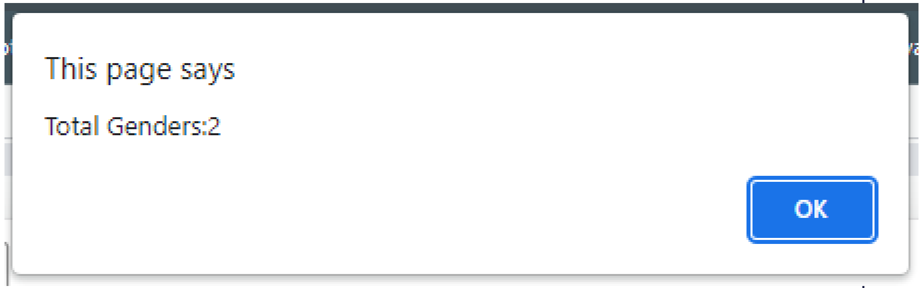
document.getElementsByName() method

- The document.getElementsByName() method returns all the element of specified name
- **Syntax**: document.getElementsByName("name")

Example of document.getElementsByName() method

```
<script type="text/javascript">
function totalelements()  {
 var allgenders=document.getElementsByName("gender");
 alert("Total Genders:"+allgenders.length);
}
</script>

<form>
Male:<input type="radio" name="gender" value="male">
Female:<input type="radio" name="gender" value="female">
<input type="button" onclick="totalelements()" value="Total Genders">
</form>
```


This page says

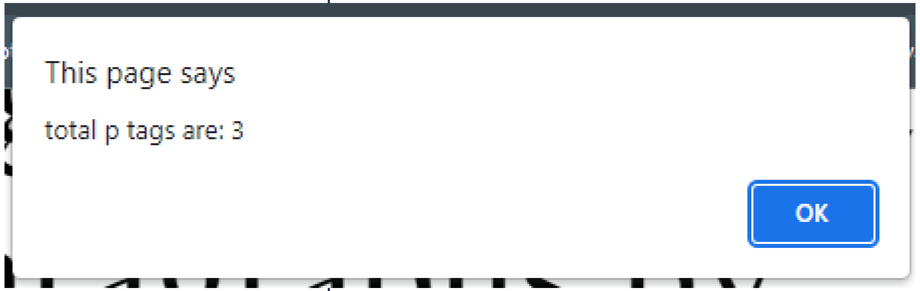
Total Genders:2

document.getElementsByTagName()

- The document.getElementsByTagName() method returns all the element of specified tag name
- Syntax: document.getElementsByTagName("name")

```
<script type="text/javascript">
function countpara(){
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);
 }
</script>


<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraph
s by getElementByTagName() method.</p>
<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button>
```



This page says

total p tags are: 3

Javascript - innerHTML

- The **innerHTML** property can be used to write the dynamic html on the html document.
- It is used mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.

innerHTML example

```
<script type="text/javascript" >
function showcommentform() {
var data="Name:<input type='text' name='name’>
<br>Comment:<br><textarea rows='5' cols='80'></textarea>
<br><input type='submit' value='Post Comment'>";
document.getElementById('mylocation').innerHTML=data;
}
</script>
<form name="myForm">
<input type="button" value="comment" onclick="showcommentform()">
<div id="mylocation"></div>
</form>
```

Javascript – innerText

- The innerText property can be used to write the dynamic text on the html document
- Here, text will not be interpreted as html text but a normal text
- It is used mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc

innerText example

```
<script type="text/javascript" >
function validate() {
var msg;
if(document.myForm.userPass.value.length>5){
msg="good";
}  else{  msg="poor";  }
document.getElementById('mylocation').innerText=msg;
 }
</script>
<form name="myForm">
<input type="password" value="" name="userPass" onkeyup="validate()">
Strength:<span id="mylocation">no strength</span>
</form>
```

?