Client-side Technologies

Eng. Niveen Nasr El-Den
SD & Gaming CoE
iTi

Day 4

JavaScript Objects

JavaScript Objects

JavaScript Objects fall into 4 categories:

1. **Custom Objects (User-defined)**
   - Objects that you, as a JavaScript developer, create and use.
2. **Built - in Objects (Native)**
   - Objects that are provided with JavaScript to make your life as a JavaScript developer easier.
3. **BOM Objects "Browser Object Model" (Host)**
   - It is a collection of objects that are accessible through the global objects window. The browser objects deal with the characteristic and properties of the web browser.
4. **DOM Objects "Document Object Model"**
   - Objects provide the foundation for creating dynamic web pages. The DOM provides the ability for a JavaScript script to access, manipulate, and extend the content of a web page dynamically.

JavaScript built-in Objects

JavaScript Built-in Objects

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

- String
- Number
- Array
- Date
- Math
- Boolean
- RegExp
- Error
- Function
- Object

String Object

- Enables us to work with and manipulate strings of text.

- String Objects have:
  - Property
    - length : gives the length of the String.
  - Methods that fall into three categories:
    - Manipulate the contents of the String
    - Manipulate the appearance of the String
    - Convert the String into an HTML element

- To create a String Object
  - var str = new String('hello');

Methods Manipulating the contents of the String Object

var myStr = "Let's see what happens!";

Method	Example	Returned value
charAt	myStr.charAt(0)	L
charCodeAt	myStr.charCodeAt(12)	97// unicode of a=97
split	myStr.split(“ “,3)	["Let's", "see", "what"]
indexOf	myStr.indexOf("at")	12
lastIndexOf	myStr.lastIndexOf("a")	16
substring	myStr.substring(0, 7)	Let's s
concat	myStr.concat(“ now”);	Let's see what happens! now
replace	myStr.replace(/e/,”?”)	L?t's see what happens!
	myStr.replace(/e/g,”?”);	L?t's s?? what happ?ns!

Methods Manipulating the appearance of the String Object

Name	Example	Returned value
big	“hi".big()	<BIG>hi</BIG>
bold	"hi".bold()	<B>hi</B>
fontcolor	"hi".fontcolor("green")	<FONT COLOR="green“>hi</FONT>
fontsize	"hi".fontsize(1)	<FONT SIZE="1">hi</FONT>
italics	"hi".italics()	<I>hi</I>
small	"hi".small()	<SMALL>hi</SMALL>
strike	"hi".strike()	<STRIKE>hi</STRIKE>
sup	"hi".sup()	^{hi}

Other Useful Methods

Method name
toLowerCase()
toUpperCase()
endsWith()
startsWith()
includes()
repeat()
search()
trim()
trimRight()
trimLeft()

Number Object

- **Number** objects are not primitive objects, but if you use a number method on a primitive number, the primitive will be converted to a Number object behind the scenes and the code will work.
  - It is an object wrapper for primitive numeric values.
- Example:
  - var n = 123;
  - typeof n;
    → "number“
  - n.toString()
    → "123“
  - n.toString(16)
    → “7b"

Number Object

- To create a Number Object

  → var n = new Number(101);

  OR

  → n = new Number();

  // if not assigned a value initially n = 0

  → n=10;

  // value changed to n=10

- Number class has a set of *Constant values* & object methods.

Number Object Constants

1. Class Constants

Properties	Description
Number.MAX_VALUE	A constant property (cannot be changed) that contains the maximum allowed number. →1.7976931348623157e+308
Number.MIN_VALUE	The smallest number you can work with in JavaScript. →5e-324
Number.NaN	Contains the Not A Number number.
Number.POSITIVE_INFINITY	Contains the Infinity number. It is read-only.
Number.NEGATIVE_INFINITY	Has the value -Infinity.

Number Object Constants

- Class Constant Methods

Methods	Example
Number.isInteger()	Number.isInteger(11.2)//false
Number.isFinite()	Number.isFinite(123)//true
Number.isNaN()	Number.isNaN(“aa12”)//true
Number.parseInt()	Number.parseInt(“123”)//123
Number.parseFloat ()	Number.parseFloat (“123.2”)//123.2

Number Object Methods

var n = new Number(10)

Methods	Description	Example

toFixed(x)	Fixed-point representation of a number object as a string. Rounds the returned value.	n = 34.8896;
		n.toFixed(6);
		//34.889600

toExponential(x)	Exponential notation of a number object as a string. Rounds the returned value.	n = 56789;
		n.toExponential(2);
		// "5.68e+4"

toPrecision(x)	Formats any number so it is of "x" length	n = 34.8896; n.toPrecision (3); //34.9

Other Methods

var n = new Number(10)

Methods	Description	Example
toString()	Converts from decimal system to any other system when passing its base as parameter	var x=n.toString(16); //a
	Returns a string representing the Number object.	var numStr = n.toString(); //"10"
valueOf()	returns the primitive value of a Number object as a number data type.	var x = 5 + n.valueOf() ; //15
toLocaleString()	returns a string representing the number with the equivalent language sent as function parameter.	(123). toLocaleString('ar-EG'); //١٢٣

Math Object

- Allows you to perform common mathematical tasks.
- The Math object is a *static object*.
- Math is a little different from other built in objects because it cannot be used as a constructor to create objects.
- Its just a collection of functions and constants

Math Object

- Math object has:
  - I- Properties (constant values)
  - II- Methods

- Example:

```
var circleArea = Math.PI * radius * radius;
```

Math Object Properties

Name	Returned value
Math.E	Returns Euler's constant
Math.PI	Return the value of π (PI)
Math.SQRT2	Returns the square root of 2
Math.SQRT1_2	Returns the square root of 0.5
Math.LN2	Returns the natural logarithm of 2
Math.LN10	Returns the natural logarithm of 10
Math.LOG2E	Returns the log base -2 of E
Math.LOG10E	Returns the log base -10 of E

Math Object Methods

Name	Example	Returned value
max	Math.max(1 , 700)	700
min	Math.min(1 , 700)	1
sqrt	Math.sqrt(9801)	99
pow	Math.pow(6, 2)	36
random	Math.random()	.7877896
round	Math.round(0.567)	1
floor	Math.floor(0.567)	0
ceil	Math.ceil(0.567)	1
sin	Math.sin(Math.PI)	0
cos	Math.cos(Math.PI)	-1
tan	Math.tan(1.5 * Math.PI)	5443746451065123

Math Object Methods

Name	Example	Returned value
abs	Math.abs(-6.5)	6.5
acos	Math.acos(.5)	1.04719755119659763
asin	Math.asin(1)	1.5707963267948966
atan	Math.atan(.5)	0.4636476090008060935
sqrt	Math.sqrt(9801)	99
exp	Math.exp(8)	2980.957987041728302
log	Math.log(5)	1.6094379124341002

Array Object

- Array is actually a special type of object
- It has length property:
  - gives the length of the array
  - It is one more than the highest index in the array
- To declare an array use
  - new keyword
  - array literal notation

Array Object

- Using new operator:

→ var colorArray = new Array();

colorArray [0]="red";

colorArray [1]="blue";

colorArray [2]="green";

OR

→ var colorArray = new Array(3);
colorArray [0]="red";
colorArray [1]="blue";
colorArray [2]="green";

OR

→ var colorArray = new Array("red","blue","green");

//this is called dense array where array is populated at the time it is declared

- Use array literal notation

→ var arr = ["apple", "banana", "grapes"];
→ var arr = [, 1, , , "a"];

Array Object Methods

var arr1=new Array(“A”,”B”,”C”);

var arr2 = new Array(1,2,0);

Name	Example	Result
concat	arr1.concat(arr2);	A,B,C,1,2,0 //neither arr1 nor arr2 changed
join	arr1.join() arr1.join(“*”)	A,B,C A*B*C //arr1 not changed
reverse	arr1.reverse()	C,B,A
pop	arr1.pop()	C // and arr1.length becomes 2
push	arr1.push(“D”);	4 // 4 → Length of the array // resulting in : arr1[3]=“D”

Array Object Methods

```
var  arr1=new Array("A","B","C");

var arr2 = new Array(4,2,3,0);
```

Name	Example	Result
shift	arr1.shift();	Returns: A arr1[0] ="B" & arr[1]="C"
unshift	arr1.unshift("D");	arr1[0]="D" //length become 4
slice	arr1.slice(1); arr1.slice(2);	B,C C //arr1 not changed
sort (according to Unicode)	arr2.sort()	0,2,3,4

Associative Array

- The Arrays That Aren't
- Associative array is just like an ordinary array, except that instead of the indices being numbers, they're strings, which can be a lot easier to remember and reference.
- The key idea is that every JavaScript object is an associative array
- Can't be access the array using numeric indexes.
- Associative arrays let you specify key-value pairs.
- Although the keys for an associative array have to be strings, the values can be of any data type, including other arrays or associative arrays.
- **Syntax:**

```
var assocArray = new Array( );
 assocArray["one"] = "one";
assocArray[“1"] = "two";
assocArray[“Next Value"] = “Three";
assocArray[“new"] = 2;
```

Date Object

- To obtain and manipulate the day and time in a script.
- The information either takes the value from the user's computer or from a specified date and time
- To create date object:
  var varName = new Date([parameters])
  - Parameters are
    - Year, month, date of the month, hour, minute, second, and milliseconds
  - Example:
    var varName = new Date()
    var varName = new Date(milliseconds)
    var varName = new Date(datestring)
    var varName = new Date(yr, month, date [, hrs, min, sec, msec])

Date Object Number Conventions

Date Attribute	Numeric Range
seconds, minutes	0 - 59
hours	0 - 23
day	0 - 6 (0 = Sunday, 1 = Monday, and so on)
date	1 - 31
month	0 - 11 (0 = January, 1 = February, and so on)
year	0 + number of years since 1900

Date Object

- The Date object methods fall into these broad categories:

1. *"get"* methods

   → for getting date and time values from date objects

2. *"set"* methods

   → for setting date and time values in date objects

3. *"to"* methods

   → for returning string values from date objects.

Date Object “get” Methods

var now = new Date ("November 25,2009");

Name	Example	Returned Value
getDate	now.getDate()	25
getMonth	now.getMonth()	10
getYear	now.getYear()	109
getDay	now.getDay()	6
getHours	now.getHours()	0
getMinutes	now.getMinutes()	0
getSeconds	now.getSeconds()	0
getTime	now.getTime()	The internal, millisecond representation of a Date object similar to now.valueOf()

Date Object “set” Methods

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

Name	Example
setDate	someDate.setDate(6)
setHours	someDate.setHours(14)
setMinutes	someDate.setMinutes(50)
setMonth	someDate.setMonth(7)
setSeconds	someDate.setSeconds(7)
setTime	someDate.setTime(yesterday.getTime())
setYear	someDate.setYear(88)

Date Object “to” Methods

```
var   now = new Date ( "November 25,2009");
```

Name	Example	Returned value
toUTCString	now.toUTCString()	Tue, 24 Nov 2009 22:00:00 GMT //used instead of the deprecated toGMTSting()
toLocaleString	now.toLocaleString("ar-eg")	٢٥/١١/٢٠٠٩ ١٢:٠٠:٠٠ ص
	now.toLocaleString()	11/25/2009, 12:00:00 AM
toString	now.toString()	Wed Nov 25 2009 00:00:00 GMT+0200 (Egypt Standard Time)

Date Object

- Hours should be specified using a 24-hour clock.
- The month is always indexed from zero, so that November is month 10.
- The year can also be offset by 1900, so that you can use either of these two forms

```
var NovDate = new Date(90, 10, 23);
var NovDate = new Date(1990, 10, 23);
```

- For the year 2000 and beyond you must use the second form

```
var NovDate = new Date(2006, 10, 23);
```

- This form may optionally take an additional three integer arguments for the time, so that 1:05 PM on November 23, 1990 is

```
var NovDate2 = new Date(90, 10, 23, 13, 5, 0);
```

Boolean Object

- The Boolean object is used to convert a non-Boolean value to a Boolean value (true or false).
- Everything in the language is either “truthy” or “falsy”
- The rules for truthiness:
  - 0 , ”” , NaN , null , and undefined → falsy
  - Everything else → truthy
- You can convert any value to it’s boolean equivalent by applying “!!” preceding the value
  - Example:

    !!”” ➔ false

    !!123 ➔true
- To create Boolean Object
  - var b = new Boolean(); → false // typeof is Object
  - B = false → false // typeof “boolean”

Boolean Object

- All the following lines of code create Boolean objects with an initial value of false:

```
var myBoolean=new Boolean()
var myBoolean=new Boolean(0)
var myBoolean=new Boolean(null)
var myBoolean=new Boolean(undefined)
var myBoolean=new Boolean("")
var myBoolean=new Boolean(false)
var myBoolean=new Boolean(NaN)
```

- And all the following lines of code create Boolean objects with an initial value of true:

```
var myBoolean=new Boolean(true)
var myBoolean=new Boolean(1)
var myBoolean=new Boolean("false“)
var myBoolean=new Boolean(“anyThing")
```

RegExp Object

- Regular expressions provide a powerful way to search and manipulate text.
- A Regular Expression is a way of representing a pattern you are looking for in a string.
- A Regular Expression lets you build *patterns* using a set of special characters. Depending on whether or not there's a match, appropriate action can be taken.
- People often use regular expressions for *validation* purposes.
  - In the validation process; you don't know what exact values the user will enter, but you do know the format they need to use.

RegExp Object

- Specified literally as a sequence of characters with forward slashes (/) or as a JavaScript string passed to the RegExp( ) constructor

- A regular expression consists of:
  - A pattern used to match text, Mandatory parameter.
  - Zero or more modifiers (also called flags) that provide more instructions on how the pattern should be applied, Optional parameter.

RegExp Object

- to create regular expression objects
  - **Explicitly using the RegExp object**
    - var searchPattern = new RegExp(“pattern” [, “flag”]);
    - var re = new RegExp("j.*t")
  - **Using literal RegExp**
    - var myRegExp = / pattern / [flag] ;
    - var re = /j.*t/;
- In the example above,
  - j.*t is the regular expression pattern. It means, "Match any string that starts with j, ends with t and has zero or more characters in between".
  - The asterisk * means "zero or more of the preceding";
  - the dot (.) means "any character“

RegExp Object

- Modifers can be passed as a second parameter in any combination of the following characters and in any order
  - "g" for global
  - "i" for ignoreCase
  - "m" for multiline
  - etc.

https://javascript.info/regexp-introduction

- Example:
  - var re = new RegExp('j.*t', 'gmi');
  - var re = /j.*t/ig;

RegExp Object Properties

- global:
  - If this property is false, which is the default, the search stops when the first match is found. Set this to true if you want all matches.
- ignoreCase:
  - Case sensitive match or not, defaults to false.
- multiline:
  - Search matches that may span over more than one line, defaults to false.
- lastIndex:
  - The position at which to start the search, defaults to 0.
- source:
  - Contains the regexp pattern.

Once set, the modifier cannot be changed

RegExp Methods

- test()
  - returns a boolean (true when there's a match, false otherwise)
  - Example:

    /j.*t/.test("Javascript")

    →false case sensitive

- exec()
  - returns an array of matched strings.
  - Example:

    /j.*t/i.exec("Javascript")[0]

    →"Javascript"

String Methods that Accept Regular Expressions as Parameters

- .match(regex)
  - returns an array of matches
- .search(regex)
  - returns the position of the first match
- .replace(regex, txt)
  - allows you to substitute matched text with another string
- .split(delimiter [, limit])
  - also accepts a RegExp when splitting a string into array elements

RegExp Syntax

Character	Description	Example
.	Any character	/a.*a/ matches "aa", "aba", "a9qa", "a!?_a",
^	Start	/^a/ matches “apple”, “abcde”..
$	End	/z$/ matches “abcz”, “az”..
|	Or	/abc|def|g/ matches lines with "abc", "def", or "g"
[ ]	Match any one character between the brackets	/[a-z]/ matches any lowercase letter
[^]	Match any one character not between the brackets	/[^abcd]/ matches any character but not a, b, c, or d

RegExp Syntax

Character	Description	Example	
*	0 or more	/Goo*gle/ →"Gogle", "Google", "Gooogle", "Goooogle"	
+	1 or more	/Goo+gle/ →"Google", "Gooogle", "Goooogle"	
?	0 or 1	/Goo?gle/ →"Gogle", "Google",	
{min, max}	{min,} → min or more	{2,} 2 or more	/a(bc){2,4}/ →"abcbc", "abcbcbc", or "abcbcbcbc"
	{,max} → up to max	{,6} up to 6	
	{val} → exact value	{3} exactly 3	

https://regex101.com/tests
http://regexr.com/

Browser Object Model

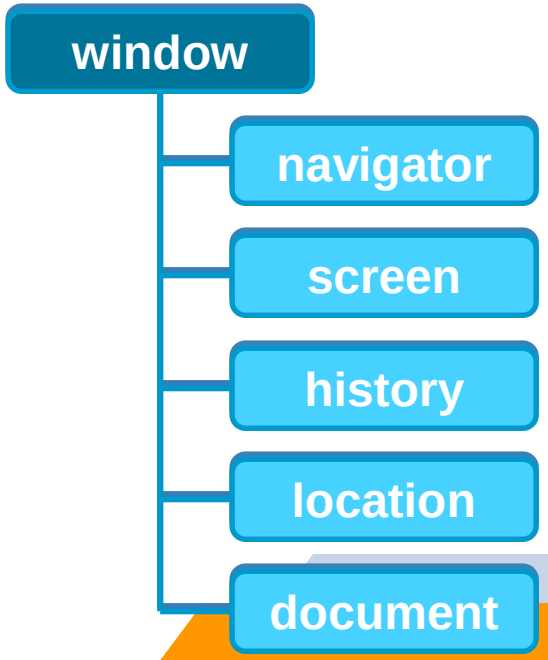
BOM

BOM

- BOM Stands for Browser Object Model.
- BOM covers objects which relate to the browser.
- At the top of the **BOM** hierarchy is **window** object. Below that comes the
  - **navigator** object,
  - **screen** object,
  - **history** object,
  - **location** object, and
  - **document** object
    - It is the top level of the **DOM** hierarchy.

  Each object below the window is of equal status. (comes in no particular order).

  They all relate directly to the window object.



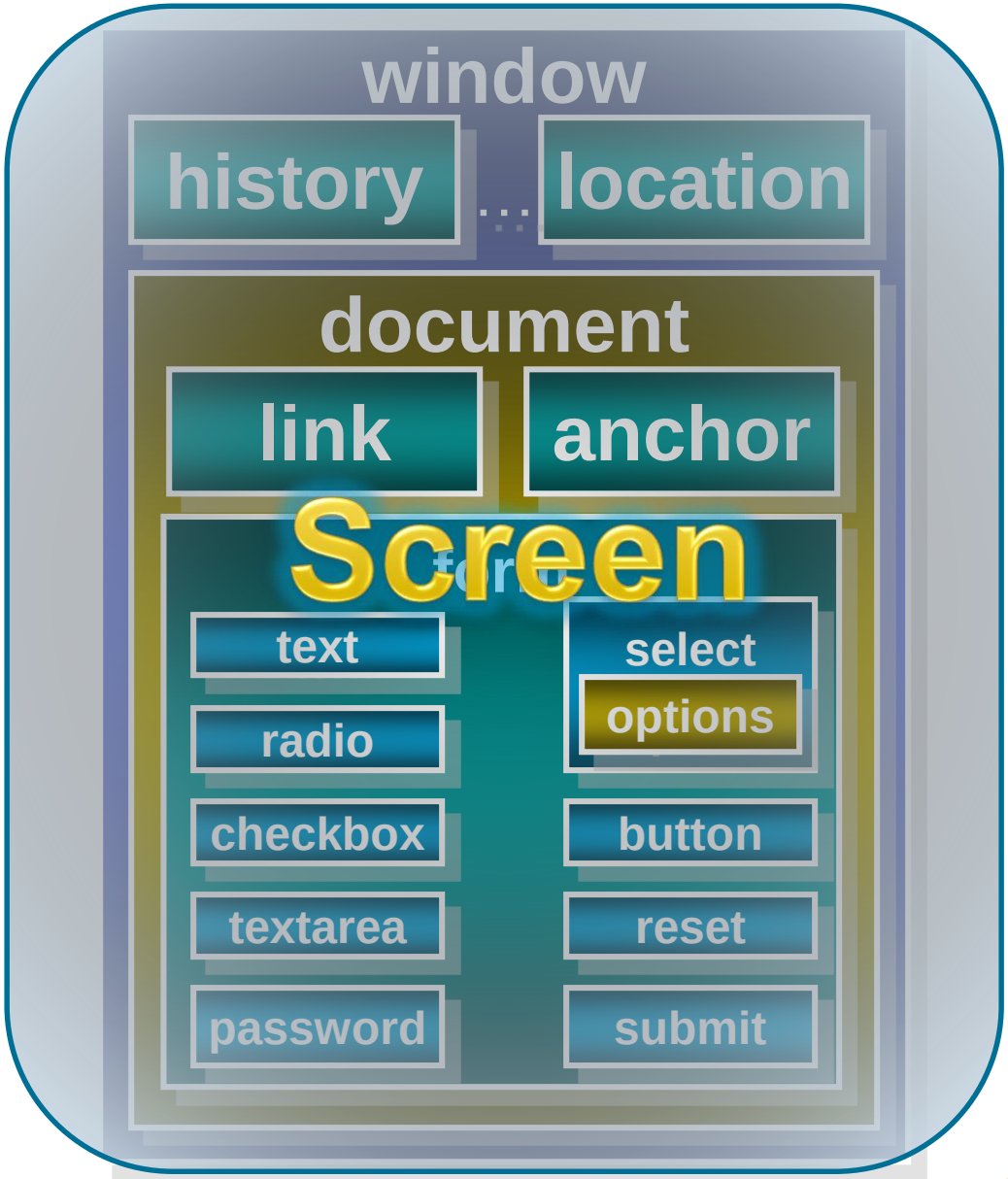
BOM

- Using the **BOM**, developers can move the window, change text in the status bar, and perform other actions that do not directly relate to the page content.
- For some reason, the **B**rowser **O**bject **M**odel is generally not referred to by its proper name. More often, it's usually wrapped up with the **DOM**.
- In actuality, the **DOM**, which relates to all things pertaining to the document, resides *within* the **BOM**.
- Because no standards exist for the BOM, each browser has its own implementation.

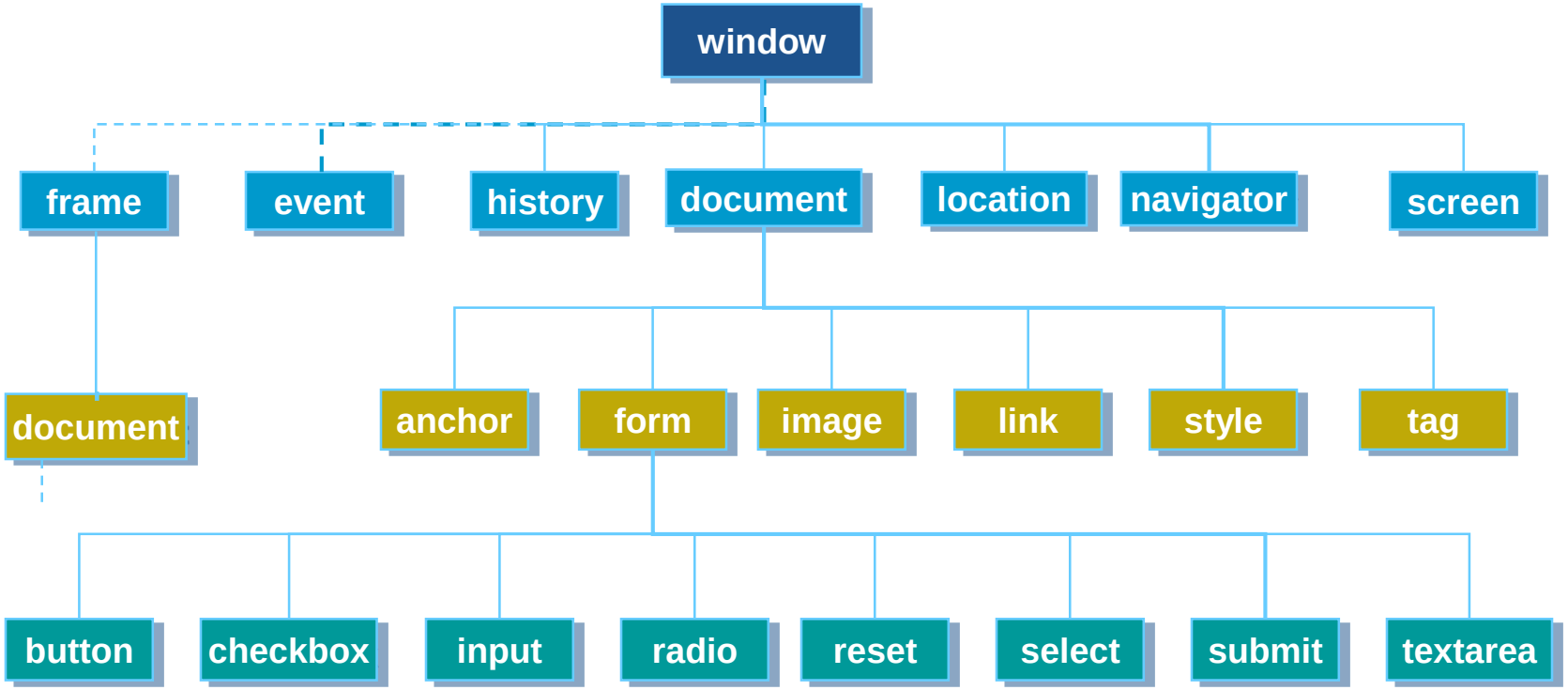
JavaScript Top Object Model Hierarchy

- Every page has the following objects:
  - window : the top-level object; has properties that apply to the entire window.
  - navigator : has properties related to the name and version of the Navigator being used.
  - document : contains properties based on the content of the document, such as title, background color, links, and forms.
  - location : has properties based on the current URL.
  - history : contains properties representing URLs the client has previously requested.
  - screen : contains information about the visitor's screen.

Browser Model



Model Hierarchy



BOM is a larger representation of everything provided by the browser including any other functionality the browser may expose to JavaScript.

Window

- Window is the top level object in the JavaScript client hierarchy.
- Window is the Global Object
- The Window object represents a browser window.
- Window object has a set of properties & methods.
- Object Model Reference:
  window
- To reference its properties & methods:
  - [window.]property
  - [window.]method

Window Properties

Name	Description	Syntax
opener	Returns a reference to the window object that created this current window Note: If the current window has no opener, this method returns NULL.	**window.opener**
closed	**Specifies whether or not a window has been closed. Returns true if the window is closed**	**window.closed**
innerHeight	Gets the height of the content area of the browser window including, if rendered, the horizontal scrollbar.	window.innerHeight
innerWidth	Gets the height of the content area of the browser window including, if rendered, the vertical scrollbar.	window.innerWidth
outerheight / outerwidth	These properties determine the dimensions, in pixels, of the outside boundary of browser window.	**window.outerheight** **window.outerwidth**

Window Properties

Name	Description	Syntax
document	Reference to the current document object.	window.document
frames	An array referencing all of the frames in the current window.	window.frames[i]
history	Reference to the History object of JavaScript	window.history
navigator	Reference to the browser application	window.navigator
location	Reference to the Location object of JavaScript	window.location

Window Methods

Name	Description	Syntax
alert()	Displays an alert box with a message and an OK button	window.alert(“Hello”)
confirm()	Displays a dialog box with a message and an OK, returning true, and a Cancel, returning false	Window.confrim(“Do you want to exit”)
prompt()	Displays a dialog box that prompts the user for input	name=prompt("Please enter your name","")
open()	Opens a new browser window	window.open(URL, name [, features])
close()	close a specified window	window.close()
blur()	Sets focus away from the window.	window.blur()
focus()	Set calling window object on top	window.focus()
print()	Print the contents of the specified window.	window.print()

Window Methods

Name	Description	Syntax
moveTo(h,v)	Moves the window to horizontal and vertical position relative top-left of screen	window. moveTo(,)
moveBy(h,v)	Moves the window by + or - horizontal and vertical pixels	window.moveBy(,)
resizeTo(h,v)	Changes the size of the window to horizontal and vertical number of pixels	window.resizeTo(,)
resizeBy(h,v)	Changes the size of the window by + or - horizontal and vertical pixels	window.resizeBy(,)
scrollTo(h,v)	Scrolls the document in the current window or frame to horizontal and vertical pixel postions from top of document	window.scrollTo(,)
scrollBy(h,v)	Scrolls the document in the current window or frame by + or - horizontal and vertical pixel from current position	window.scrollBy(,)

Assignment