Client-side Technologies

Eng. Niveen Nasr El-Den
iTi

Day 5

Basics of JavaScript

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

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

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

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
  - JavaScript has no pure associative array.
  - Associative array is just like an ordinary array, except that instead of the indices being numbers, they're strings, hence they do not have a length property.
    - The indices are replaced by user defined keys.
  - 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.
  - Associative array is simply a set of key-value pairs

- The key idea is that every JavaScript object is an associative array which is the most general sort of array you can invent - sometimes this is called a hash or map structure or a dictionary object.

Associative **Array**

- **Example:**

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

for (let i in assocArray)
    console.log(i+”:”+ assocArray[i]);
```

Objects are Associative arrays

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
getFullYear	now.getFullYear()	2009
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())
setFullYear	someDate.setFullYear(2008)

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
toString	now.toString()	'Wed Nov 25 2009 00:00:00 GMT+0200 (Eastern European Standard Time)'
toLocaleString	now.toLocaleString()	11/25/2009, 12:00:00 AM
	now.toLocaleString(‘ar-EG’)	’٢٥/١١/٢٠٠٩ ١٢:٠٠:٠٠ ص’
	now.toLocaleString(‘ar-EG’,arrDate)	11/25/2009, 12:00:00 AM
toLocaleDateString	now.toLocaleDateString()	'11/25/2009'
	now.toLocaleDateString(‘ar-EG’)	‘٢٥/١١/٢٠٠٩'

arrDate = [weekday: 'long', year: 'numeric’,
month: 'long', day: 'numeric’]

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”

Browser Object Model

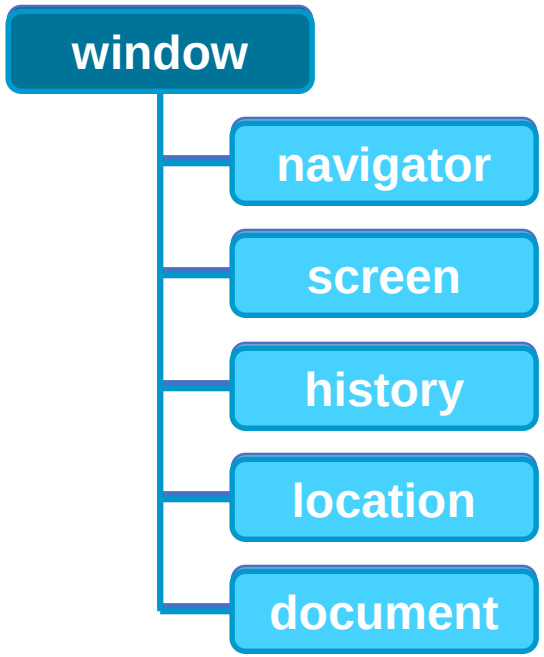
BOM

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.



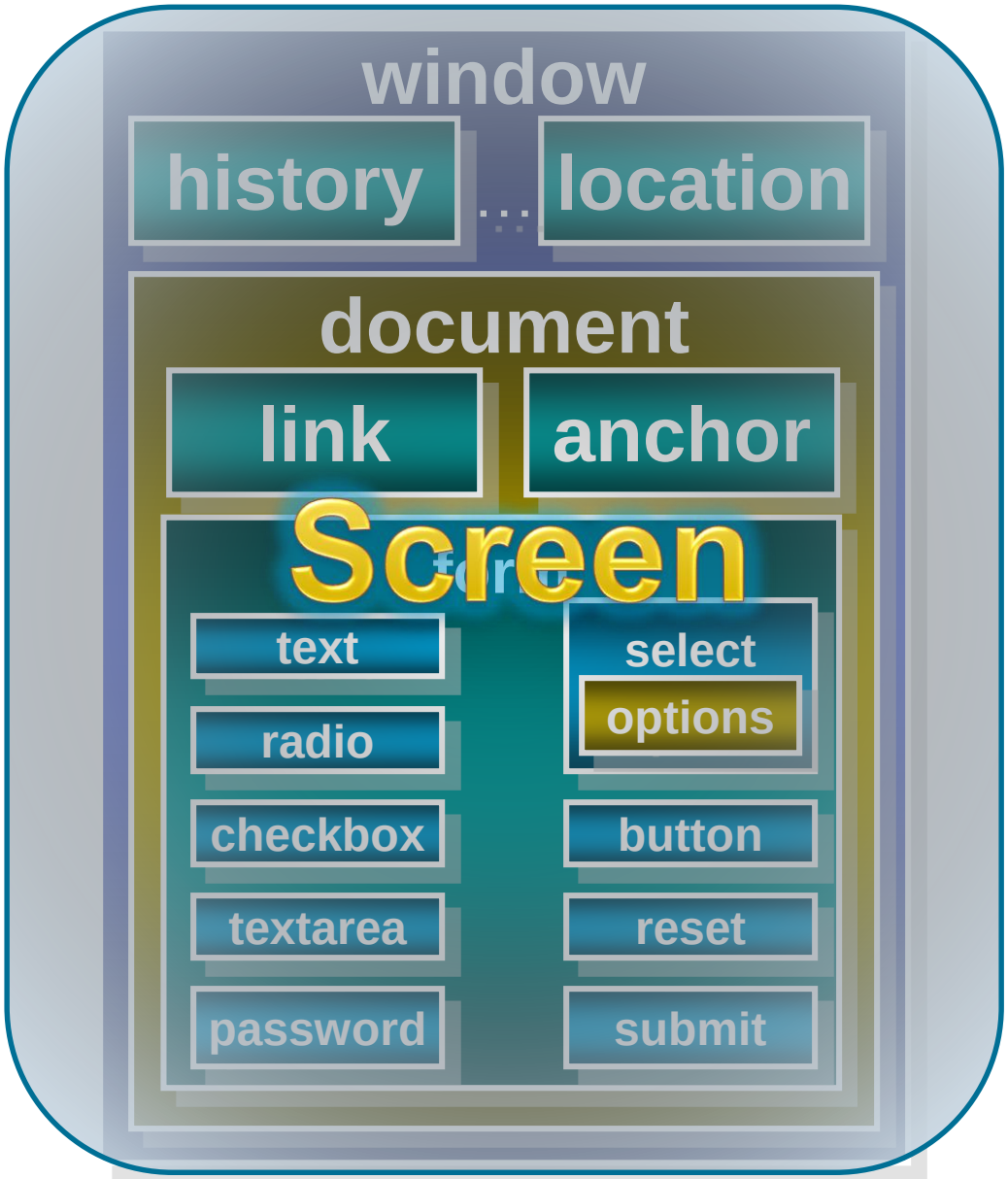
BOM

- Using the **BOM**, developers can move the window, 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

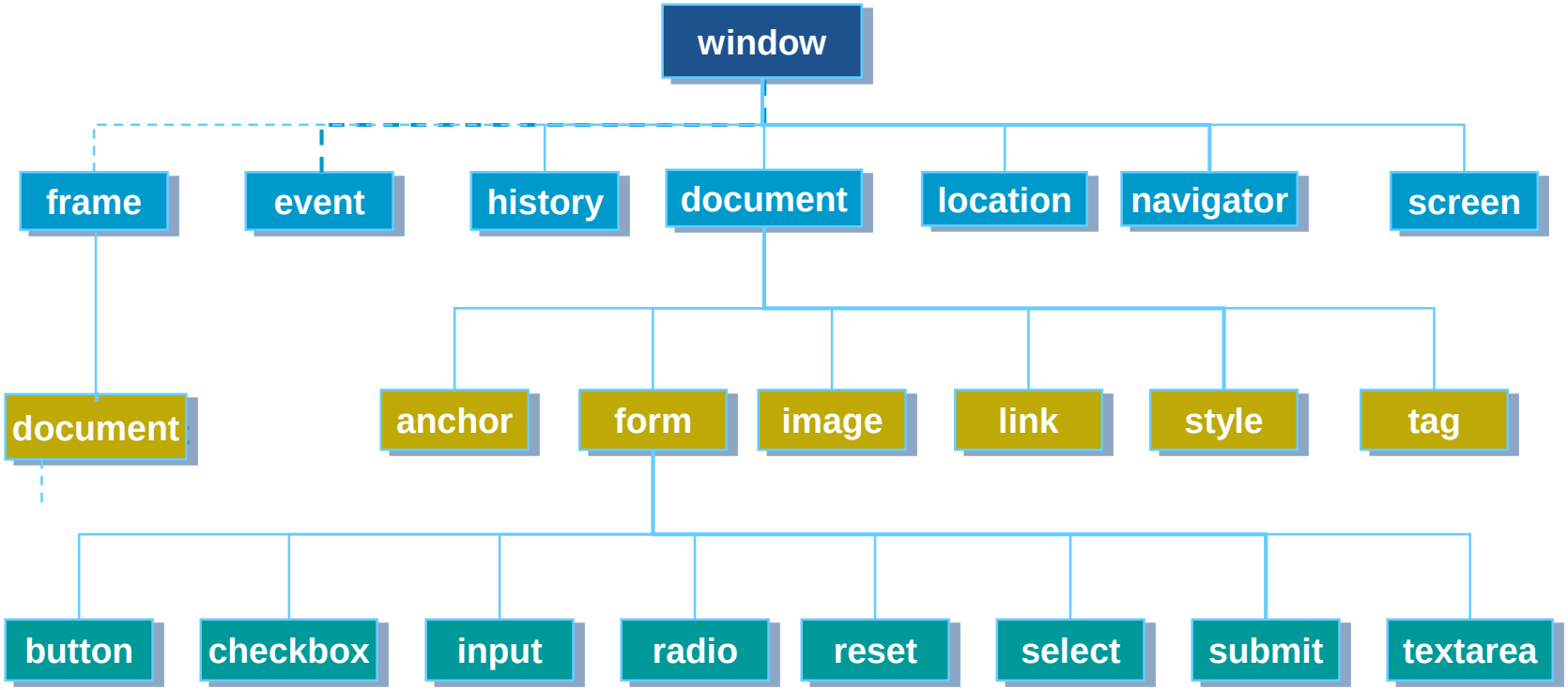
- **Every page has the following objects:**
  - **window *:* the top-level object; has properties that apply to the entire window.**
  - **navigator *:* has properties related to the name and version of the Navigator being used.**
  - **document *:* contains properties based on the content of the document, 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
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(,)

Window Methods (WindowTimers)

Name	Description	Syntax
setInterval()	Evaluates an expression at specified intervals	window.setInterval(*exp*, *time_interval*)
clearInterval()	Used to clear a time interval set using the above method	clearInterval(id_of_setInterval)
setTimeout()	Evaluates an expression after a specified number of milliseconds	window.setTimeOut(*exp*, *time_interval*)
clearTimeout()	Used to clear a timeout set using the above method	clearTimeout(id_of_setTimeout)

Example!

Assignments