Client Side Technologies

JavaScript Built-in objects

ITI – Assiut Branch
Eng. Hany Saad

Quiz…?

1. JavaScript code is interpreted by:

a- Web server b- JavaScript Console in the browser c- Java Compiler

2. JavaScript code is executed on:

a- Server side b- Client side

3. JavaScript Can't access any files on client PC, except cookies.

a- True b- False

4. What's the result of executing This JS code?

```
<script>
        <Center> My First JS code </Center>
        document.write ("<b> Hello World</b>");
</script>
```

a- Will print Hello World in bold letters b- Error in Line 2 c- Error in line 3

JavaScript Objects

❑ JavaScript Objects fall into 4 categories:

1. **Custom Objects**
   - Objects that you, as a JavaScript developer, create and use.
2. **Built – in Objects**
   - Objects that are provided with JavaScript to make your life as a JavaScript developer easier.
3. **BOM Objects "Browser Object Model"**
   - 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

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

String Object

- **Enables us to work with and manipulate strings of text.**
- **String Objects have:**
  - **One Property:**
    - length : gives the length of the String.
  - **Methods that fall into three categories:**
    1. Manipulate the contents of the String
    2. Manipulate the appearance of the String
    3. Convert the String into an HTML element

String Object (Cont.)

1- Methods manipulating the contents of the String

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

Method	Description	Example
charAt(index)	Returns the character at the specified index	document.write(myStr.charAt(0)); //L
indexOf(string)	Returns the position of the first found occurrence of a specified value in a string	document.write(myStr.indexOf("at")); //12 document.write(myStr.indexOf("@")); //-1
lastIndexOf(string)	Returns the position of the last found occurrence of a specified value in a string	document.write(myStr.lastIndexOf("a")); //16

substring(index,index)	Extracts the characters from a string, between two specified indices	document.write(myStr.substring(1, 7)); //et's s
		document.write(myStr.substring(2)); //t's see what happens!

String Object (Cont.)

1- Methods manipulating the contents of the String (Cont.)

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

Method	Example	Returned value
replace(string)	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring	document.write(myStr.replace(/e/,”?”)); //L?t's see what happens!
		document.write(myStr.replace(/e/g,”?”)); //L?t's s?? what happ?ns!

String Object (Cont.)

2- Methods manipulating the appearance of the String

Method name	Example	Returned value
bold()	"hi".bold()	hi
fontcolor()	"hi".fontcolor("green")	<FONT COLOR="green“>hi</FONT>
fontsize()	"hi".fontsize(1)	<FONT SIZE="1">hi</FONT>
italics()	"hi".italics()	<I>hi</I>
strike()	"hi".strike()	<STRIKE>hi</STRIKE>
sup()	"hi".sup()	^{hi}
toLowerCase()	"UPPERcase".toLowerCase()	uppercase
toUpperCase()	"UPPERcase".toUpperCase()	UPPERCASE

String Object (Cont.)

Method name	Example	Returned value
link(string)	“Click me".link("linktext") Or myStr. link("linktext")	Click me

❑ String Object Complete Reference:

- http://www.w3schools.com/jsref/jsref_obj_string.asp

Math Object

- **Allows you to perform common mathematical tasks.**

- **The Math object is a static object.**

- **Math object has:**
  - Properties (constant values)
  - Methods

- **Example:**

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

Math Object(Cont.)

1. Math Object Constants

Name	Returned value
Math.E	Returns Euler's constant
Math.PI	Return the value of π (PI)

Math Object(Cont.)

2. Math Object Methods

Name	Example	Returned value
cos(n)	Math.cos(.4)	0.9210609940028851028
sin(n)	Math.sin(Math.PI)	0
tan(n)	Math.tan(1.5 * Math.PI)	infinity
acos(n)	Math.acos(.5)	1.0471975511965976 31
asin(n)	Math.asin(1)	1.5707963267948965 58
atan(n)	Math.atan(.5)	0.4636476090008060935
exp(n)	Math.exp(8)	2980.957987041728302
log(n)	Math.log(5)	1.6094379124341002 82

Math Object(Cont.)

2. Math Class Methods(cont.)

Name	Example	Returned value
max(x,y,…)	Math.max(1 , 700)	700
min(x,y,…)	Math.min(1 , 700,2)	1
sqrt(n)	Math.sqrt(9801)	99
pow(x,n)	Math.pow(2, 3)	8
abs(n)	Math.abs(-6.5)	6.5
random()	Math.random()	.7877896
floor(n)	Math.floor(8.9)	8
ceil(n)	Math.ceil(8.1)	9
round(n)	Math.round(.567)	1

❑ Math Object Complete Reference:

- http://www.w3schools.com/jsref/jsref_obj_math.asp

Array Object

To declare an array:

```
<script>
        var colorArray = new Array();
        colorArray [0]=“red”;
        colorArray [1]=“blue”;
        colorArray [2]=“green;
//OR
        var colorArray = new  Array(“red”,”blue”,”green”);
//OR
        var colorArray=[];
        var colorArray=[“red”,”blue”,”green”];
</script>
```

Array Object has One Property:

- **length :** gives the length of the array

Array Object (Cont.)

1- Object Methods

var arr1=new Array("A","B","C"); var arr2 = new Array("1","2","0")

Methods	Description	Example

join()	Joins all elements of an array into a string	document.write(arr1.join());
		//A,B,C
		document.write(arr1.join("*"));
		//A*B*C

reverse()	Reverses the order of the elements in an array	document.write(arr1.reverse()); //C,B,A
pop()	Removes the last element of an array, and returns that element	document.write(arr1.pop()); //C, and the length becomes 2

push(element)	Adds new elements to the end of an array, and returns the new length	document.write(arr1.push("D"););
		//4 (Length of the array)
		//and arr1[3]="D"

Array Object (Cont.)

1- Object Methods

var arr1=["A","B","C"]; var arr2 = [1,2,10];

Methods	Description	Example

shift()	Removes the first element of an array, and returns that element	document.write(arr1.shift()); // A
		//arr1[0] ="B" & arr[1]="C"

unshift(element)	Adds new elements to the beginning of an array, and returns the new length	document.write(arr1.unshift("D")); //4 //arr1[0]="D"
sort()	Sorts the elements of an array alphapitacally (By default, and can take a function for custom sort)	document.write(arr2.sort()) ;//1,10,2
splice()	See: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array	
find()		
filter()		

- **Array Object Reference:** http://www.w3schools.com/jsref/jsref_obj_array.asp

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

Array Object (Cont.)

- **Associative Arrays:** The Arrays That Aren't
  - Associative arrays provide let you specify key-value pairs.
  - 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:
  - 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:**

```
<script>
        var assocArray = new Array( );
        assocArray[“Ahmed"] = “Excellent";
        assocArray[“Tareq"] = “pass";
</script>
```

Literal object

What is an Object?

- An object is an unordered list of primitive data types (and sometimes reference data types) that is stored as a series of name-value pairs.
- Each item in the list is called a property (functions are called methods).

```
//creating object with some properties
var emp1= { Name: "Richard", Age: 30};
var emp2={Name:”Ali”, Age:20};

// using object
//you don’t need to create instance
emp1. Name=“Mahmoud”;
. alert(emp1.Name);
```

Literal object (Cont.)

❑ Adding Methods to the object

```
var emp1={ name:“Aly", age: 23,
                show:function ( )
                {
                        alert(this.name + " is " + this.age + " years old.“);
                }
        };

// calling method, and again you don’t need to create instance
emp1.show();
```

Date Object

- To obtain and manipulate the date and time in a script.

  - **Syntax:**

```
<script>
    var d = new Date(); // holds current date
    var d = new Date(dateString); // Ex. "October 13, 2014 11:13:00"
    var d = new Date(year, month, day, hours, minutes, seconds, milliseconds);
</script>
```

Date Object (Cont.)

❑ 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	1900 90 2000

Date Object (Cont.)

- **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 (Cont.)

1. get Methods

var myDate= new Date ("November 25,2006 11:13:55");

Name	Example	Returned Value
getDate()	myDate.getDate()	25
getMonth()	myDate.getMonth()	10
getFullYear()	myDate. getFullYear()	2006
getDay()	myDate.getDay()	0
getHours()	myDate.getHours()	11
getMinutes()	myDate.getMinutes()	13
getSeconds()	myDate.getSeconds()	55
getTime()	myDate.getTime()	11:13:55

Date Object (Cont.)

2. set Methods

```
var  someDate = new Date ();
var   myDate = new Date ( "November 25,2006 11:13:55");
```

Name	Example
setDate(number)	someDate.setDate(25)
setHours(number)	someDate.setHours(14)
setMinutes(number)	someDate.setMinutes(50)
setMonth(number)	someDate.setMonth(7)
setSeconds(number)	someDate.setSeconds(7)
setTime(TimeString)	someDate.setTime(myDate.getTime())
setFullYear(number)	someDate.setFullYear(88)

Date Object (Cont.)

3. to Methods

var myDate = new Date ("November 25,2006 11:13:00");

Name	Example	Returned Value
toUTCString()	myDate.toUTCString()	Sat, 25 Nov 2006 09:13:00 UTC
toLocaleString()	myDate.toLocaleString()	25 نوفمبر, 2006 11:13:00 ص (Based on date format in your OS)
toLocaleTimeString()	myDate.toLocaleTimeString()	11:13:00 ص
toLocaleDateString()	myDate.toLocaleDateString()	01 نوفمبر, 2006
toString()	myDate.toString()	Sat Nov 25 11:13:00 UTC+0200 2006
toDateString()	myDate.toDateString()	Sun Nov 1 2006

- **Date Object Complete Reference:**
  - http://www.w3schools.com/jsref/jsref_obj_date.asp

Regular expression 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.
- Regular expressions is often used for the purposes of validation.
- In the validation process; you don't kmyDate what exact values the user will enter, but you do kmyDate the format they need to use.

Regular expressions(Cont.)

- **Pattern:**
  - Mandatory parameter, the regular expression you use to match text.

- **Mode/Flag:**
  - Optional parameter, indicates the mode in which the Regular Expression is to be used:
    - → (i) ignore case.
    - →(g) global search.
    - →(m) Multiline

<

  - Flags can be passed in any combination and/or in any order

Regular expressions(Cont.)

❑ Regular Expression syntax:

- Regular expressions can be created:
  - → Explicitly using the RegExp object:

```
var searchPattern = new RegExp(“pattern” [ , “flag”]);
var searchPattern = new RegExp("j.*t", "i");
```

  - → Using literal RegExp:

```
var myRegExp = / pattern / [flag] ;
var myRegExp = /j.*t/i;
```

Mode

Regular Expression

Regular expressions(Cont.)

- **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“.

Regular expressions(Cont.)

❑ 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

Regular expressions(Cont.)

- 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	

- /d (any digit)
- () (group)

Regular expressions(Cont.)

❑ Regular Expression 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.

Regular expressions(Cont.)

❑ Regular Expression Object Methods:

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

```
var reg=/j.*t/ ;
var t= reg.test("Javascript")
```

→false **case sensitive**

- **exec()**
  - returns first matched strings.
  - Example:

```
Var reg=/j.*t/ i;
Var str="Jscript is the same of javascript";
var res= reg.exec(str);
```

Regular expressions(Cont.)

- **String Methods that Accept Regular Expressions as Parameters :**
  - **match()**

    - returns an array of matches.

  - **search()**

    - returns the position of the first match.

  - **replace()**

    - allows you to substitute matched text with another string.

```
document.write(myStr.replace(/e/,"?"));
//L?t's see what happens!

document.write(myStr.replace(/e/g,"?"));
//L?t's s?? what happ?ns!
```

  - **split()**

    - also accepts a RegExp when splitting a string into array elements.

Regular expressions(Cont.)

- **RegExp Object patterns Reference:**
  - http://www.w3schools.com/jsref/jsref_obj_regexp.asp
  - http://regexlib.com/CheatSheet.aspx

- **RegExp Library:**
  - http://www.regxlib.com/

- **Test your Regular Expression:**
  - https://regex101.com/#javascript
  - http://www.regexr.com/
  - http://regexpal.com/

Error Object

- **Whenever an error occurs, an instance of the Error object is created to describe the error.**

- **Error objects can be created in 2 ways:**
  - Explicitly:

```
var newErrorObj = new Error();
```

  - Implicitly:
    → thrown using the throw statement.

Error Object(Cont.)

❑ Error Object Properties:

Property	Description
description	Plain-language description of error
fileName	URI of the file containing the script throwing the error
lineNumber	Source code line number of error
message	Plain-language description of error (ECMA)
name	Error type (ECMA)
number	Microsoft proprietary error number

Error Object(Cont.)

- **Error constructor:**
  - var e = new Error();
- **Six additional Error constructor ones exist and they all inherit Error:**

EvalError	Raised by eval when used incorrectly
RangeError	Numeric value exceeds its range
ReferenceError	Invalid reference is used
SyntaxError	Used with invalid syntax
TypeError	Raised when variable is not the type expected
URIError	Raised when encodeURI( ) or decodeURI( ) are used incorrectly

Using *instanceOf* when catching the error lets you know if the error is one of these built-in types.

Error Object(Cont.)

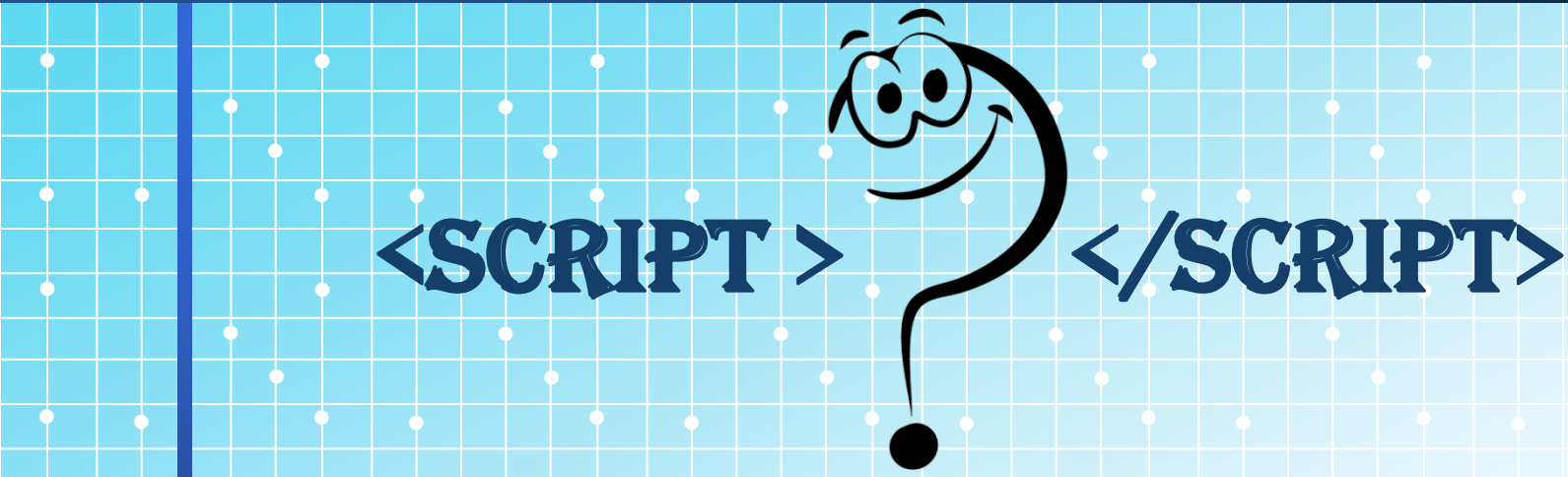
❑ Error Object standard Properties:

- **Name:** The name of the error constructor used to create the object
- **Example:**

```
var e = new Error('Oops');
```

- **Message:** Additional error information
- **Example:**

```
var e = new EvalError('jaavcsritp is _not_ how you spell it');
document.write(e.name); //EvalError
document.write(e.description); // jaavcsritp is _not_ how you spell it
```



<script>document.writeln(“Thank You!”)</script>