Client-side Technologies

Dr. Niveen Nasr El-Den
iTi

Day 4

JavaScript Fundamentals cont.

Basics of JavaScript cont.

User-defined functions

- Function can be called before its declaration block.
- Functions are not required to return a value
  - Functions will return undefined implicitly if it is to set explicitly
- When calling function it is not obligatory to specify all of its arguments
  - The function has access to all of its passed parameters via arguments collection
  - We can specify default value using || operator or ES6 default function parameters

Example!

User-defined functions with default value

```
function dosomething (x) {
     x = x || “nothing was sent“;
    //x = x ?x: “nothing was sent”;
    //x = ( typeof x == “number”) ? x : 10;

     console.log (“value is :” + x);
}
```

```
dosomething(“hello”)
// value is : hello

dosomething()
// value is : nothing was sent

dosomething(0)
// value is : 0
```

ES6 Default value

```
function doSomething (x = “nothing was sent“) {
        console.log (“value is :” + x);
}
```

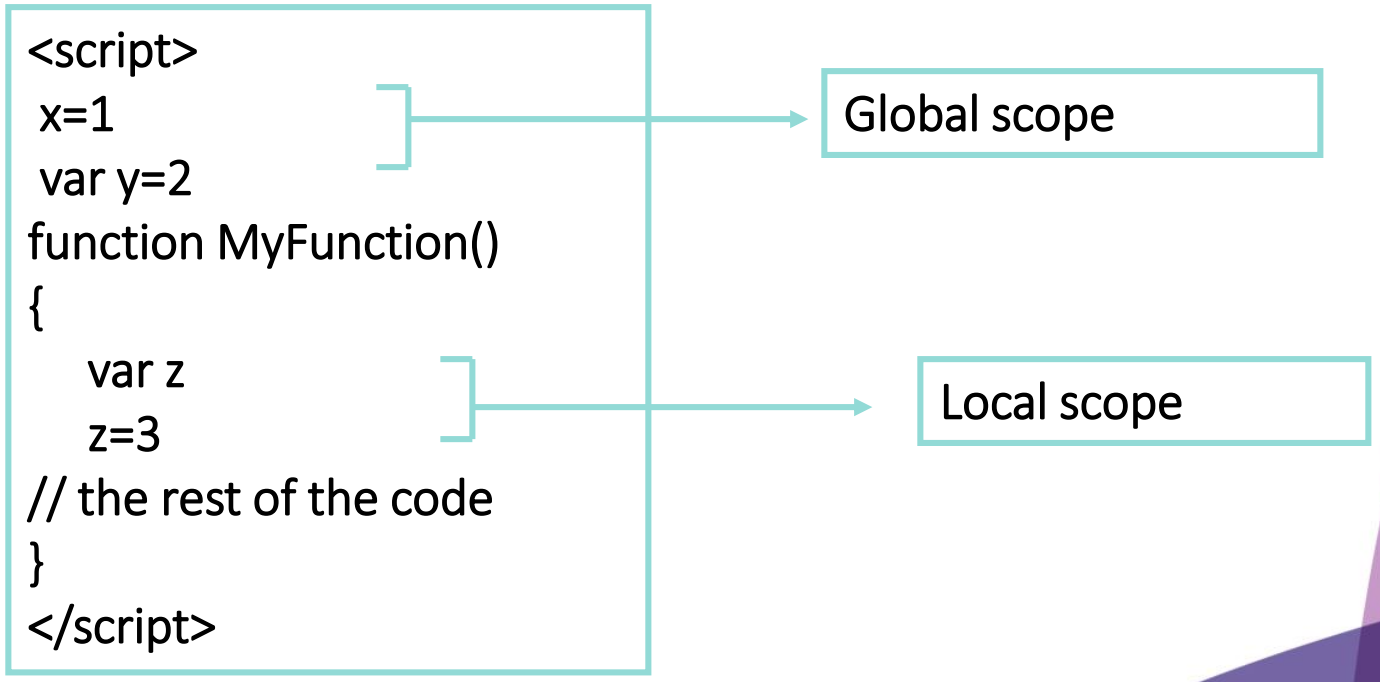
```
doSomething(“hello”)
// value is : hello

doSomething()
// value is : nothing was sent

doSomething(0)
// value is : 0
```

JavaScript Variables Lifetime

- Global Scope
  - A variable declared outside a function and is accessible in any part of your program
- Local Scope
  - A variable inside a function and stops existing when the function ends.



Variable Scope

- It is where var is available in code
- All properties and methods are in the public global scope
  - They are properties of the *global* object "window"
- In JavaScript, var scope is kept within functions, but not within blocks (such as while, if, and for statements) scope
  - NOTE: ES6 represents block scope via let, const.
- Variables declared with var
  - inside a function are local variable
  - outside any function are global variable
    - i.e. available to any other code in the current document

Example!

ES6 Variable Declarations

- **Constants**
  - Constants are declared with const keyword and must be assigned at the time of the declaration.

    ```
    const myConst = value;
    ```
  - Constants are read-only, can't be modified later .
  - Constants can't be declared with the same name as a function or variable in the same scope.
  - A constant can be global or local to a function where it is declared.
- **"let"**
  - Similar to variables but used for block declaration
- Naming a const/let in JavaScript follow the same rule of naming a var except that the const keyword is always required, even for global constants and let is required for block declaration.
- A **script scope** is created when let and /or const is declared in **global scope**

Hoisting

- Hoisting takes place before code execution
- Variables
  - Any variable declared with var is hoisted up to the top of its scope
  - Hoisted variables are of undefined value.
  - We can refer to a var declared later without getting any exception or reference error.
  - Hoisting of let and const of is different from var hoisting
- Functions
  - Function statements are hoisted too.
  - Functions are available even before its declaration

Example!

Temporal Dead Zone

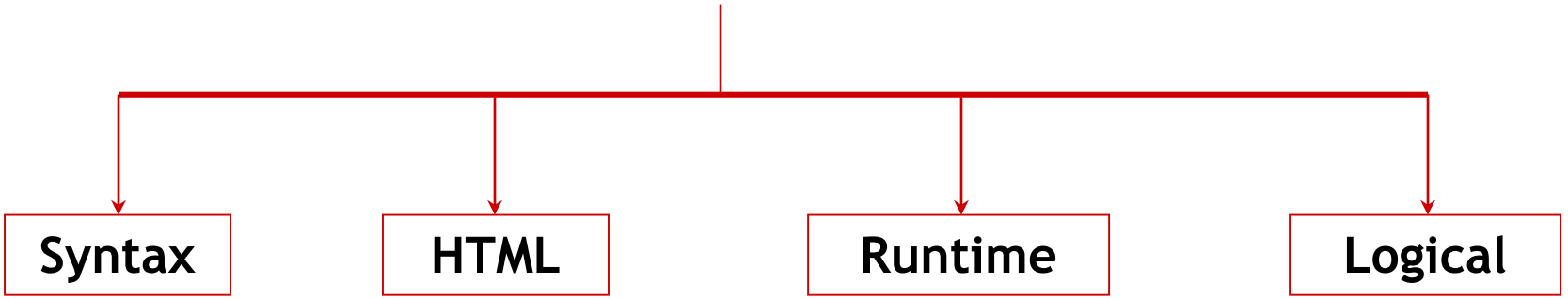
- **Temporal Dead Zone** (TDZ)is the period of time during which the let and const declarations cannot be accessed.

- TDZ starts when the code execution enters the block which contains the let or const declaration and continues until the declaration has executed.

- TDZ forbids any use of the variable before its declaration

JavaScript Debugging Errors

Types of Errors



Inaccurate capitalization, or forgetting to close quotation marks or parentheses.

Forgetting a fundamental HTML step will cause your JavaScript code to fail.

Technically correct but performs an invalid function, such as dividing by zero, generate script that calls a non-existent function

Code that may not return an error but does not produce the result you expect.

JavaScript Console Object

- Modern browsers have JavaScript console within developer tool (F12) where errors in scripts are reported
  - Errors may differ across browsers
- Console Object is a non-standard that provides access to the browser's debugging console
- The console object exists only if there is a debugging tool that supports it.
  - Used to write log messages at runtime
- Do not use it on production

JavaScript Console Object

- Methods of the `console` object:
  - debug(message)
  - log(message)
  - warn(message)
  - error(message)
  - clear()
  - etc..

https://developer.mozilla.org/en/docs/Web/API/console

JavaScript Special Characters

Character	Meaning
\b	Backspace
\v	Vertical tab
\t	Horizontal tab
\n	New line
\r	Carriage return
\\	Backslash
\’	Single quote
\”	Double quote

JavaScript Objects

JavaScript Objects for Client-side RTE

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

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!

Other Useful Methods

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

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.
- etc.

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 RegEx 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", "Goooogle", "Gooooogle	
+	1 or more	/Goo+gle/ →"Google", "Goooogle", "Gooooogle	
?	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/

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
- Array is a data structure that used to represent list of items
- 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

Other Useful Methods

Method name
toReversed()
toSorted()
toSpliced()
with()
at()
fill()
flat()
indexOf()
include()

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

Assignments