Javascript

●●●

Lecture 3

Eng. Sahar E. Hassan

Javascript: Objects

- ـدObject is an entity that has some properties and some behaviors.
- Why do we need objects?
  - To group properties and functions of the same entity together (ex: grouping shape properties (such as width,height..etc) and behaviors such as (draw,calcArea..etc))
  - To allow us to return more than 1 value from a function call. By returning an object that contains multiple properties instead of just returning a single value.
- Javascript have already built in objects such as : (window, Math,..etc)

Javascript: Objects (continue)

- You can also create custom objects using literal notation.

Ex:

```
var myObj = {
        property1: "property1_value_goes_here",
        property2: "property2_value _goes_here",
        func1: function(){
                    //code goes here
        }
}
```

Javascript: Objects (continue)

- Objects can contain both properties and methods..
- Properties and method names are followed by a colon ":" then the value.
- You should separate properties and methods using a comma ","

Ex:

```
    var coordinates = { x:5 , y10 };
    var person= {
        firstName:"Ahmed",
        lastName:"Mahmoud",
         sayHello:function(){
            alert("I am saying Hello");
        }
    }
```

Javascript: Objects (continue)

- To access a property or a function inside an object use dot "."
- Ex:

```
console.log(person.firstName);          //will log the first name of the person
person.sayHello();                 //will execute the function inside the person object
person.firstName="John";      //will set the firstName property to "John"
```

- You can also use brackets [ ] to access properties and methods inside object. In that case, you must pass the property or method name as a string.
- Ex:

```
alert(person["firstName"])
person["sayHello"]();
```

- If you set a value in a non-existent property in an object, it will create it.
- Ex:

```
person.salary=5000;            //this will set a property named salary to 5000;
```

Javascript: Objects (continue)

- You can also create a javascript object using the new Object() function;
- You will then need to assign properties and methods of that object after creating the object

Ex:

```
var myObj = new Object();
myObj.name="This is my object";
myObj.age=22;
myObj.showHello = function(){
        alert("Hello");
}
```

Javascript: Objects (continue)

- In order to access the properties and functions from inside the object itself, you should use the magical keyword … "***this***"

Ex:

```
    var person = {
        firstName: "Ahmed",
        lastName: "Mohamed",
        sayHello: function(){
            alert(this.firstName + " " + this.lastName + " says hello");
            //alert(firstName) //this will throw an error because firstName is not defined
        }
    }
```

Javascript: Objects (continue)

- You can use return multiple values from a function by returning an object containing those values.

Ex:

```
function getData (){

    var fName = prompt("Enter your firstname");
  var lName = prompt("Enter your lastname");

    var obj = {

        firstName:fName,

        lastName:lName

    }

    return obj;

}
```

Javascript: For.. in

- To loop through the properties of the object, you should use the "for in" statement.

Ex:

```
var obj = { name: "Ahmed",  age: 22,  address : "Alexandria" };

for(var prop in obj){

    //inside the loop, prop is a string that holds the property name (as a string)

    alert(prop); //this will alert each of the object properties names (name then age
  then address)

    alert(obj[prop]); //this will alert the value of each property ("Ahmed" then 22 then
  "Alexandria")

}
```

Javascript: Object.keys

- It’s a method that returns an array of a given object’s own property names.

EX:

```
Var student = { studentName: ‘Mohamed’,
                Age: 20,
                track: ‘Frontend’
              };
console.log(Object.keys(student));        // [‘studentName’, ‘age’, ‘track’]
```

Javascript: type of

- The typeof operator is used to get the data type of its operand.
- The operand can be either a literal or a data structure such as a variable, a function, or an object. The operator returns the data type.
- An array is internally represented as an object.
- It returns a string.

Javascript: type of (continue)

Type	**Result**
Undefined	“undefined”
Null	“object”
Boolean	“boolean”
Number	“number”
String	“string”
Function	“function”
Object	“object”
NaN	“number"

Javascript: Math functions

Function	Description
Math.round(number)	Round the number to the nearest integer.
Math.ceil(number)	Round the number to the higher integer.
Math.floor(number)	Round the number to the lower integer.
Math.pow(x,y)	Calculate x to the power of y.
Math.sqrt(number)	Calculate the square root for the number.
Math.random()	Generate random number between (0 - 1).

Javascript: Shallow & deep copy

- Shallow copy:
  - It means that **one** level is copied. It works fine for an array or object containing only primitive values (string, number, boolean).
  - For objects and arrays that contain other objects or arrays, shallow copy will not work well, as changes made to the nested references will change the data nested in the original object or array.

- Deep copy:
  - It means that **all** levels are copied. And any change made to the nested references will not change the data in the original object or array.

Javascript: Shallow & deep copy (continue)

- **Object shallow copy:**

```
Var originalEmployee= { empName: ‘Ahmed’, age: 23, job: ‘Frontend Engineer’};

Var copiedEmployee = originalEmployee;        //shallow copy

copiedEmployee.empName = ‘Mohamed’;

console.log(originalEmployee.empName);     //Mohamed
```

Javascript: Shallow & deep copy (continue)

- **Object deep copy:**

```
Var originalEmployee= { empName: ‘Ahmed’, age: 23, job: ‘Frontend Engineer’};

Var stringifiedEmp = JSON.stringyfy(originalEmployee);//convert the object to string

Var copiedEmployee = JSON.parse(stringifiedEmp );    //revert the string back to object

copiedEmployee.empName = ‘Mohamed’;

console.log(originalEmployee.empName);     //Ahmed
```

Javascript: Shallow & deep copy (continue)

- **Array shallow copy:**

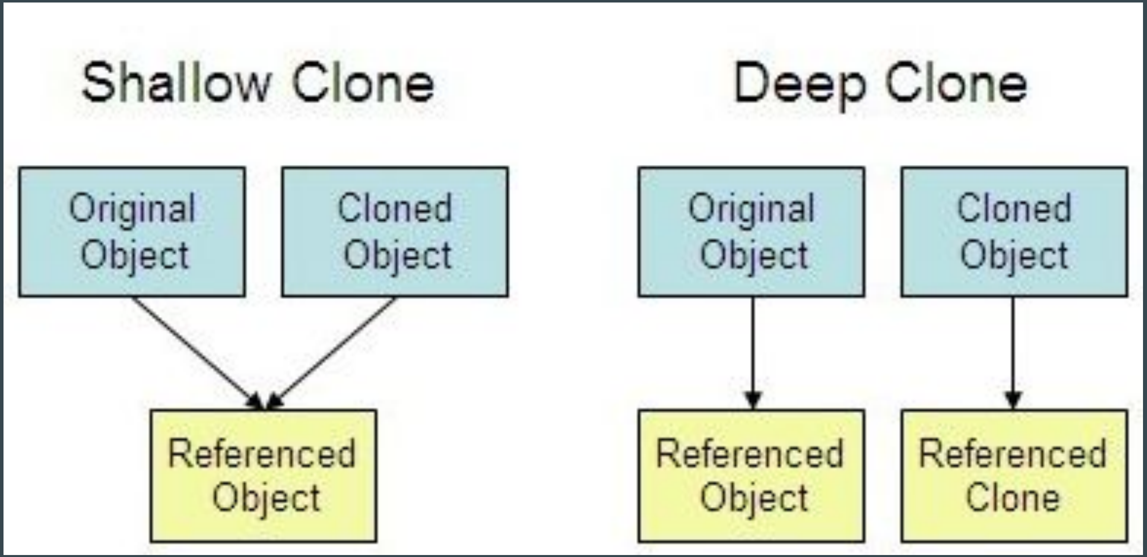
```
Var originalStudents = [‘Ahmed’, ‘Ali’, ‘Mohamed’, ‘Eslam’];

Var copiedStudents = originalStudents;          //shallow copy

copiedStudents[1] = ‘Doha’;

console.log(originalStudents);          //[‘Ahmed’, ‘Doha’, ‘Mohamed’, ‘Eslam’]
```

Javascript: Shallow & deep copy (continue)

- **Array deep copy:**

```
Var originalStudents = [‘Ahmed’, ‘Ali’, ‘Mohamed’, ‘Eslam’];

Var stringifiedStudent = JSON.stringyfy(originalStudents ); //convert the array to string

Var copiedStudent = JSON.parse(stringifiedStudent);   //revert the string back to array

copiedStudents[1] = ‘Doha’;

console.log(originalStudents);          //[‘Ahmed’, Ali, ‘Mohamed’, ‘Eslam’]
```

Javascript: Shallow & deep copy (continue)



Javascript: Window object

- All predefined methods and variables are attached to the window object.

  window.alert( ) is the same as alert( ).

- All new global custom methods and variables are also attached to window

  ```
  var x = 4;
  alert(window.x);
  ```

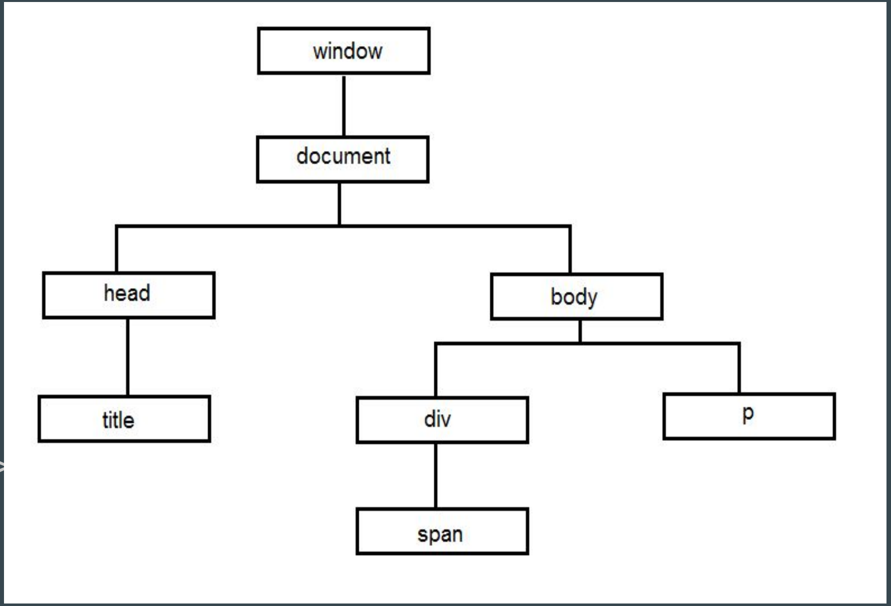
Javascript: BOM vs DOM

- BOM stands for **Browser Object Model**, which contains the data about the browser rather than the document to be rendered.
- BOM data can be accessed using the *window* object
- DOM stands for **Document Object Model**, which contains data about the rendered data (HTML elements)
- DOM data can be accessed using the *document* object

Javascript: BOM tree

- DOM elements are represented as nodes connected together in the form of a tree.

```
<html>
  <head>
    <title>Welcome to Javascript</title>
  </head>
  <body>
    <p>Hello World!</p>
          <div>
            <span> I love Javascript </span>
          </div>
  </body>
</html>
```



Javascript: Lab 3

Exercise 1:

- Create a function that accepts an object as a parameter.
- The function should alert the properties and their values.

  Ex: name: Ahmed

  age: 22

  track: ui
- The function should ignore properties of type “function”.

Javascript: Lab 3

Exercise 2:

- Create a function that generates a random number between (1-6) and returns it
- Ask the user if he want to generate a random number or not, if he clicks ok, call the function and display the number to the user.
- Keep asking the user if he wants to generate a random number or not until he clicks cancel
- Assume that the winning number is “3”, modify the function to reduce the probability of getting the number “3”

Javascript: Lab 3

Exercise 3:

- Create a function that accepts an array of strings as a parameter
- The function should return a new array (it should NOT modify the original array)
- The new array returned should have the same elements as the original array but each element is lowerCased