JavaScript 101

31 March 2017

Uğur ORUÇ
MSP - Meteor.js Developer

Ergenekon Yiğit
MSP - Open Source Developer

Welcome

Content of THIS Lesson

- History
- Where to use & Installation
- Variables & Types
- Scopes & Events
- Closures

History

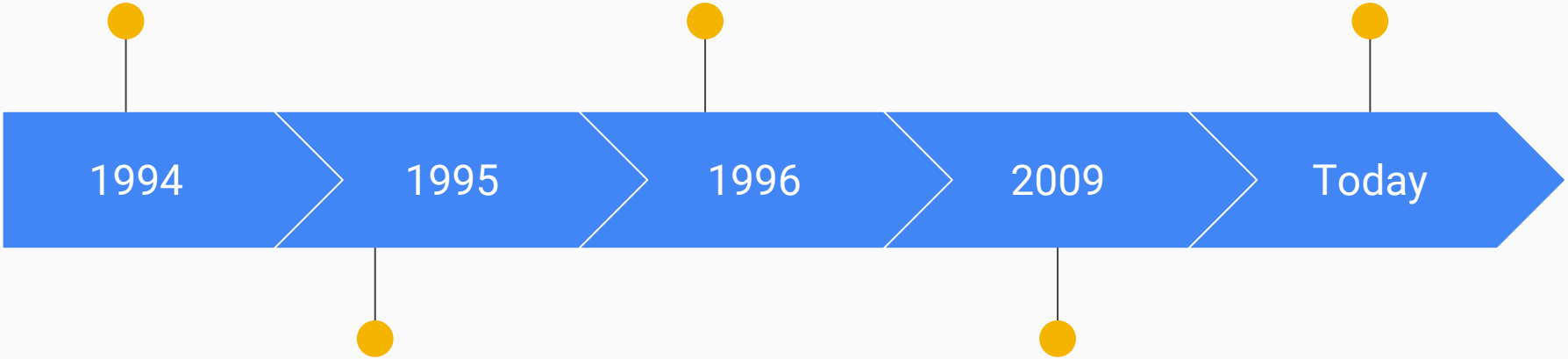
Mosaic Netscape

- Brandon Eich

ECMA

International Standards

ES2016 released,
ES2017 coming



Mocha > LiveScript > JavaScript

ES5 - NodeJS

Where to use & Installation

- You can use JavaScript anywhere you want. 😊 Such as browser , server and devices.
- You should download ... 😊 Just kidding there’s no installation for JavaScript. It has already installed on your computer.
- Let’s dive in ...

Materials

- Computer
- Internet
- Browser
- Text Editor & IDE

Variables & Types

There is 3 variable types on javascript :

- var
- let
- const

Difference Between Variable Types

Data Types

There is 5 data types can contain values

- String
- Number
- Boolean
- Object
- Function

There is 3 variable types of objects :

- Object
- Date
- Array

There is 2 data types cannot contain values

- null
- undefined

```
typeof "John"                    // Returns "string"
typeof 3.14                      // Returns "number"
typeof NaN                       // Returns "number"
typeof false                     // Returns "boolean"
typeof [1,2,3,4]                 // Returns "object"
typeof {name:'John', age:34}     // Returns "object"
typeof new Date()                // Returns "object"
typeof function () {}            // Returns "function"
typeof myCar                     // Returns "undefined" *
typeof null                      // Returns "object"
```

Typeof Operator

Comparison Operators

“ == ”	Equal Value
“ === ”	Equal Value and Type
“ != ”	Not Equal Value
“ !== ”	Not Equal Value or Type
“ < , > , <= , >= ”	

Objects

```
var Car = {
        type:"Fiat",
        model:"500",
        color:"white"
};
```

Car.type => Fiat

Car.model => 500

Car[“type”] => Fiat

Car[“model”] => 500

Arrays & Methods

```
var cars = ["Saab", "Volvo", "BMW"];

var cars = new Array("Saab", "Volvo", "BMW");


cars.push(“Mercedes”);

cars[3] = “Mercedes”;
```

```
cars.sort() => [“BMW”, ”Saab”, ”Volvo”];

cars.reverse() => [“Volvo”, ”Saab”, ”BMW”];

var points = [40, 100, 1, 5, 25, 10];

points.sort(function(a, b){

        return a - b;  // 0.5 - Math.random()

})
```

Date Formats

Type	Example
ISO Date	"2015-03-25" (The International Standard)
Short Date	"03/25/2015"
Long Date	"Mar 25 2015" or "25 Mar 2015"
Full Date	"Wednesday March 25 2015"

Date Methods

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Function Declaration

```
function name (parameter1, parameter2, parameter3){

        code to be executed

}

name(parameter1, parameter2 ,parameter3);
```

```
var x = myFunction(4, 3);          // Function is called, return value will end
up in x

function myFunction(a, b) {
    return a * b;                  // Function returns the product of a and b
}
```

```
var x = toCelsius(77);
var text = "The temperature is " + x + " Celsius";
```

Example Of Function Calling

Javascript Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

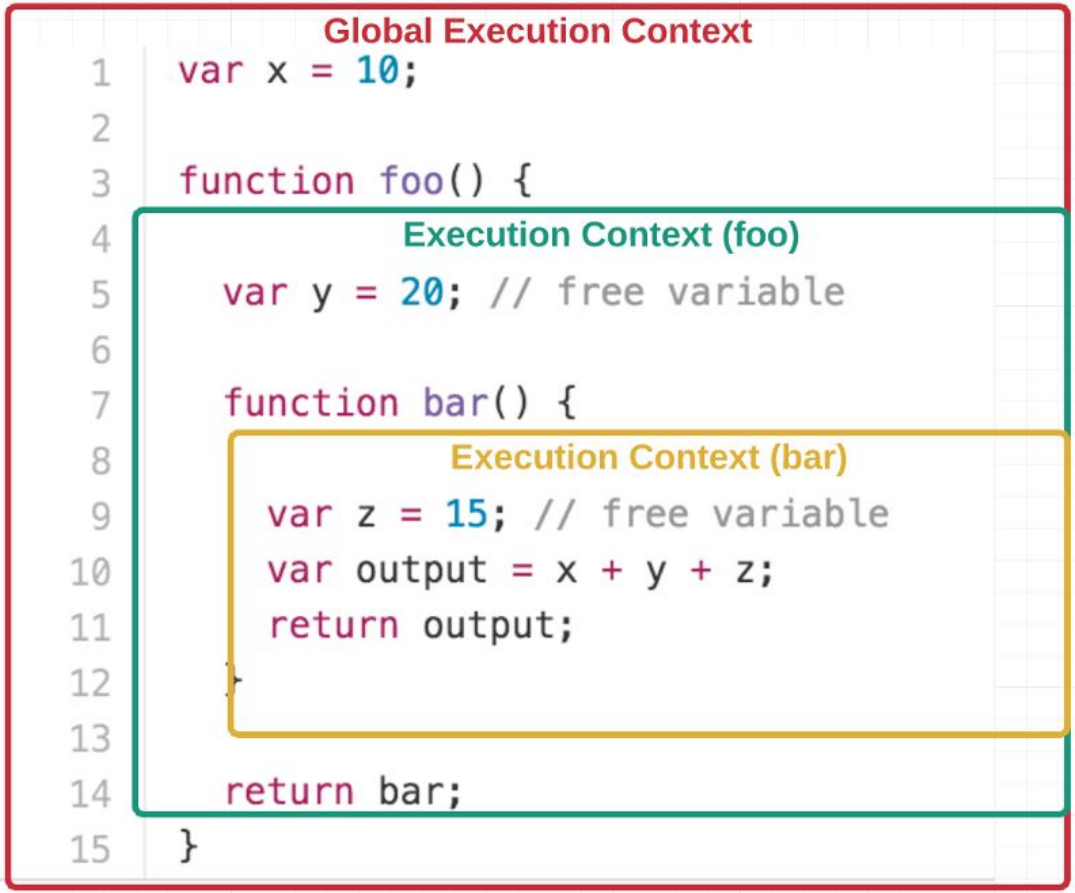
Closures

Closures

Closures are functions that refer to independent (free) variables. In other words, the function defined in the closure ‘remembers’ the environment in which it was created.

```
function numberGenerator() {
  // Local “free” variable that ends up within the closure
  var num = 1;
  function checkNumber() {
    console.log(num);
  }
  num++;
  return checkNumber;
}

var number = numberGenerator();
number(); // 2
```

Example of Closures



Example of Closures

```
function sayHello() {
  var say = function() { console.log(hello); }
  // Local variable that ends up within the closure
  var hello = 'Hello, world!';
  return say;
}
var sayHelloClosure = sayHello();
sayHelloClosure(); // ‘Hello, world!’
```

Example of Closures

```
var result = [];

for (var i = 0; i < 5; i++) {
  result[i] = function () {
    console.log(i);
  };
}

result[0](); // 5, expected 0
result[1](); // 5, expected 1
result[2](); // 5, expected 2
result[3](); // 5, expected 3
result[4](); // 5, expected 4
```

Example of Closures

```
var result = [];

for (var i = 0; i < 5; i++) {
  result[i] = (function inner(x) {
    // additional enclosing context
    return function() {
      console.log(x);
    }
  })(i);
}

result[0](); // 0, expected 0
result[1](); // 1, expected 1
result[2](); // 2, expected 2
result[3](); // 3, expected 3
result[4](); // 4, expected 4
```

Example of Closures

```
var result = [];

for (let i = 0; i < 5; i++) {
  result[i] = function () {
    console.log(i);
  };
}

result[0](); // 0, expected 0
result[1](); // 1, expected 1
result[2](); // 2, expected 2
result[3](); // 3, expected 3
result[4](); // 4, expected 4
```

Example of Closures

JSON

JavaScript Object Notation

JSON is language independent

JSON is lightweight data interchange format

```
{
        "employees":[
        {
                "firstName":"John",
                "lastName":"Doe"
        }, {
                "firstName":"Anna",
                "lastName":"Smith"}
        ]
}
```

Apply & Call & Bind

Apply & Call

```
function sayHello(firstName, secondName) {
    console.log(`${this.sayHello()} ${firstName} ${secondName}`);
}

var context = {
    sayHello() {
        return 'Hello';
    }
}

const firstName = 'Alex';
const secondName = 'Perry';

sayHello.call(context, firstName, secondName); //Hello Alex Perry
```

Arguments are separated

```
function sayHello(firstName, secondName) {
    console.log(`${this.sayHello()} ${firstName} ${secondName}`);
}

var context = {
    sayHello() {
        return 'Hello';
    }
}

const firstName = 'Alex';
const secondName = 'Perry';

sayHello.apply(context, [firstName, secondName]); //Hello Alex Perry
```

Arguments in array

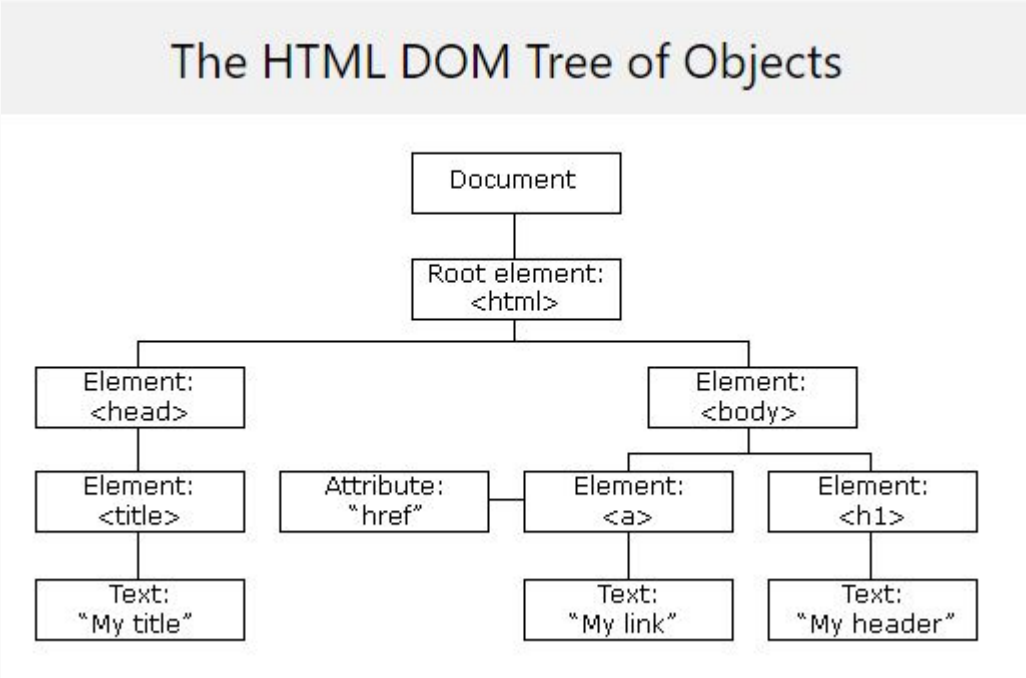
Bind

```
function sayHello(firstName, secondName, middleName) {
    console.log(`${this.sayHello()} ${firstName} ${middleName} ${secondName}`);
}

var context = {
    sayHello() {
        return 'Hello';
    }
}

const firstName = 'Alex';
const secondName = 'Perry';
const middleName = 'James';

const boundFunc = sayHello.bind(context, firstName, secondName);

boundFunc(middleName); //Hello Alex James Perry
```

The bind method enables you to pass arguments to a function without invoking it. Instead, it returns a new function with the arguments bound preceding any further arguments.

JavaScript HTML DOM Document

HTML Document Object Model



DOM Methods & Properties

DOM Properties are values can be set or change

DOM Methods are actions can perform on HTML Elements

References

- W3Schools
- Apply & Call & Bind
- StackOverflow