CITY UNIVERSITY
LONDON

Introduction to JavaScript

Aris Markogiannakis

City University - Short Courses

CS3606 : JavaScript 2: Advanced Javascript for Websites and Web

Tutor

- My name is Aris Markogiannakis
- I have been a web developer for over 15 years now, working in a number of institutions and corporates
- I am currently teaching Advanced JavaScript and .Net Core at City Short Courses
- I have been teaching at Short courses for over 5 years now.

About you?

- Tell me your name
- Job Title
- If you want where you work! And what you do at the moment?
- And what you would like to learn from this course.
- What is your knowledge in JavaScript

A very short history of JavaScript

- 1996 – Changed from LiveScript to JavaScript to attract Java Developers
- 1997 – ECMAScript 1 became the first version
- 2009 – EcmaScript 5 (ES5) was released with lots of new features
- 2015 – EcmaScript 2015 (ES2015) was released: the biggest update ever
- 2016 – EcmaScript 2016 (ES2016) was released with minor changes only.

What about today

- ES5 is fully supported in all modern browsers
- ES6/ES2015 only partial support in modern browsers, no support in older browsers. Can't use in production.
- ES2016 Almost no support in modern browsers

Course Information

- We will be using ES5 for the beginning of the course
- We will switch to ES6 after the class 6
- We will finish with two React classes
- But firstly will go through the basics today and we will do a Revision in JavaScript before we get into the very advanced areas of the programming language

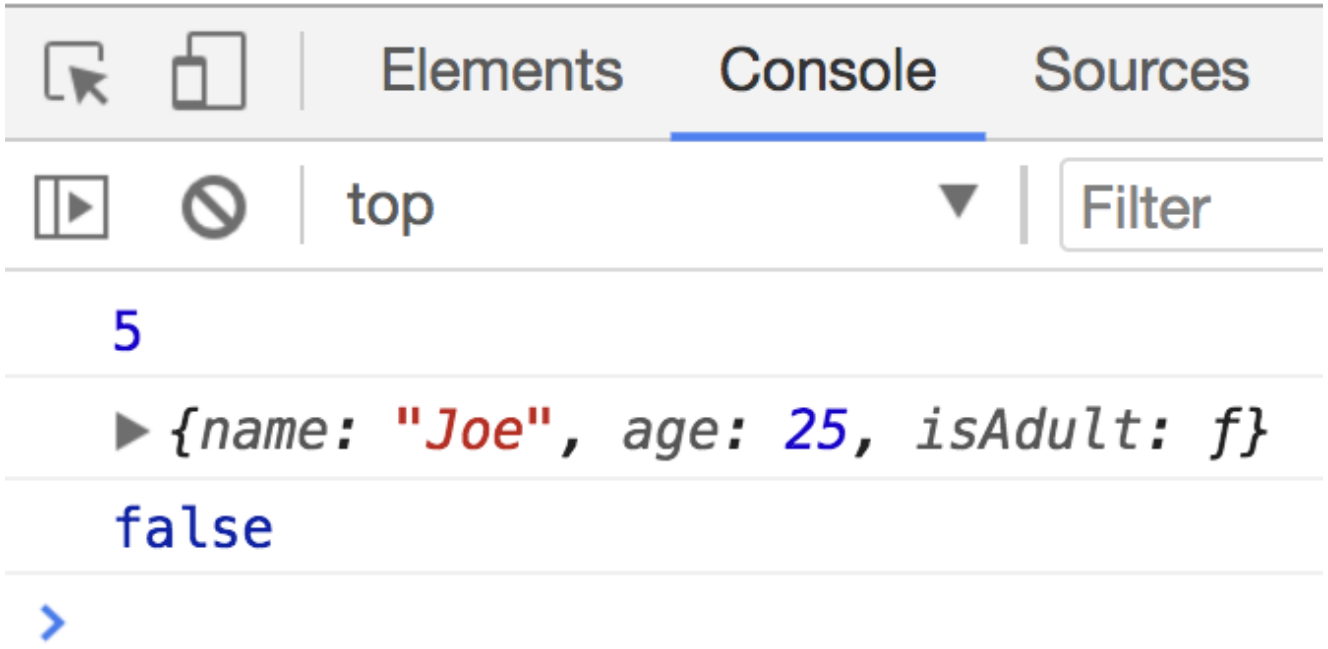
Debugging

- What are the best ways of debugging our code
  - Using console host object, it is implemented by the browsers and it is not a part of JavaScript specification
  - We use the log method to inspect expressions
  - We then press F12 and then hit the console tab.

```
// Some variables to test with
var a = 5,
    b = 10,
    c = {
        name: "Joe",
        age: 25,
        isAdult: function () {
            var isOverEighteen = this.age >= 18;
            return isOverEighteen;
        }
    };

// Inspect some variables...
console.log(a);
console.log(c);
// Inspect an expression
console.log(a > b);
```

Debugging


Elements
Console
Sources
top
Filter
5
▶ {name: "Joe", age: 25, isAdult: f}
false
>

Types

Which types do you know in JavaScript

- string
- number
- boolean
- null and undefined
- object
- symbol (new to ES6)

Arrays

- Remember… Arrays are collections/lists of things
- The values we store in arrays can be anything:
  - Strings, Integers, Booleans, Objects, other arrays
- The 1st element's index is 0
- The last element's index is the length of the array - 1

```
var myArray = ["a", "b", "c", "d"];

var firstElement = myArray[0];
var thirdElement = myArray[2];

console.log(firstElement);
console.log(thirdElement);
```

Iterating through an Array

- We can iterate through the Array using the for loop
- We will extend more on Arrays next week.

```
var myArray = ["a", "b", "c", "d"];
var totalEls = myArray.length;

// Value of "i" increases on each iteration
for (var i = 0; i < totalEls; i++) {
    // Use "i" to get elements by their index
    console.log (myArray[i]);
}
```

Exercise 1

- Now it is time to do your exercise 1, please refer to moodle and download the Lesson 1- Exercises

Object

Refers to a compound value when you can set properties that hold their own values of any type.

We use object literals and instead of square brackets we use curly brackets.

```
var objExample= {
   a: "Hello world",
   b: 42,
   c: true
}
```

Or

```
var objExample = new
Object(
```

Objects

- How do we read/write (data mutation) an object?
  - Using the dot notation – objExample.a
  - objExample.a = 25; or objExample[a] = 25; works the same as arrays!
- Is there another way to declare an object?
  - var objExample = new Object();
  - objExample.name = “Smith”;
- Can an object have functions?
  - Yes it can!

typeof

- We can check the type of a variable by using the typeof method.

```
var a;
typeof a;  // undefined

a = "hello world"
typeof a;  // string

a= 42;
typeof a; // number
```

Exercises 2

- Lets do the same with using objects if time allows..

DOM

- JavaScript rarely works alone
- When a web page is loaded by the browser, the browser builds a DOM for the page
  - DOM: Document Object Model
  - A programmatic representation of the web page, as a tree structure
  - Each element of the HTML page becomes a node in the tree
    - Element: tags, comments, text
- We access the DOM programmatically via the document object variable:
  - var pTitle = document.title
  - var pURL = document.URL
  - var pBody = document.body;
  - Var pLinks = document.links;

Elements

- Each tag is represented by an Element object and properties that allows us to manipulate the element via the DOM!
- How do we access an element?
  - If the element has an id (has to be unique) we can use
    - **document.getElementById('container')**
  - Using the **querySelectorAll** method to get elements either by their CSS class element or their tag name.
    - Document.querySelectorAll('p');
    - Document.querySelectorAll('.highlight');

```
// Look up element with id: content
var content = document.getElementById('content');
if (content !== null) {
    console.log(content.innerHTML);
}
```

```
// Look up element with id: content
var content = document.getElementById('content');
if (content !== null) {
  var pElements =  content.querySelectorAll('p');
  console.log(pElements);
}
```

Events

- JavaScript programming is al
  about responding to events
  occur in the browser.
- Events: click, mouseover, key
  load. etc. To run code in resp
  to browser events, use the **addEventListener** method of the Element object **el.addEventListener(eventName, functionToCall);**

```
// The function to run
function sayHello(event) {
    console.log("Hello world!");
}

// Element to listen for clicks on
var trigger = document.getElementById("myButton");

// Adding the listener:
// run the sayHello function when trigger is clicked
trigger.addEventListener("click", sayHello);
```

Insert content to the page

- What if we wanted to add our own text or an element to the page, can we?
- **insertAdjacentHTML()** parses the specified text as HTML or XML and inserts the resulting nodes into the DOM tree at a specified position.

- It takes two arguments
  - Position

Exercise 3

- Now it is time to do the Exercise 3