JavaScript or:

HOW I LEARNED TO STOP WORRYING AND LOVE A CLASSLESS* LOOSELY TYPED LANGUAGE

Two parts

Part 1 – JavaScript the language

Part 2 – What is it good for?

Part 1 – JavaScript the language

What is JavaScript?

Why do people hate JavaScript? / Should *you* hate JavaScript?

Do you even need to know JavaScript?

How to adapt and overcome if you come from a Java/C# background

About the JavaScript language

Some history – If its broken, why not fix it?

“Lives” in the browser

Very powerful and fast

Easy to learn

Just got a major upgrade that makes the transition from C# or Java that much easier

JS6 === ECMA Script 2015

Lots of new features (rest-parameters, lambda (arrow) functions, generators)

Making syntax more intuitive and hiding more complex patterns with more easily understood interfaces – classes and inheritance

The “let” keyword

Ability to check if you are out of bounds with your numbers

What are the difficult parts?

Loosely typed language / dynamic typing

The “var/let” type

Undefined/NaN and other strange results

Hoisting

Scope

“this”

Do not trust the compiler!

Everything (almost) is done by functions!

The “var/let“ type

An untyped variable, it can be anything!

No such thing in Java. C# has it but it is not the same at all since it implicitly sets the variable type when compiling.

‘var’ or ‘let’?

```
let a = 1;          // number
a = {};             // object
a = “ hello“;       // string
```

Hoisting and coalescing

Hoisting

All variable **declaration** takes place first in a function no matter where they are actually written.

Coalescing

All variables tries to adapt to their surroundings – what am I supposed to be now? Can I be a number, a string? Truthy? Falsy?

Scope

Scope is not the same as in Java or C# (ie not bracket defined, unless you use ‘let’ or ‘try – catch’)

Everything is in global scope by default!

var a = “ Im a global “; // defined in global scope

Scope in JavaScript is defined by... Functions unless you use the let keyword

Truth and truthiness in JavaScript

“If you tell the truth, you don't have to remember anything.”

This is especially true when we come to JavaScript.

```
==      //truthy
!=      // falsy
===     // true
!==     // false
```

Truth and truthy

```
1 == '1'                    // true
false == 0                  // true
undefined == null           // true
undefined == ' '            // false
null == 0                   // false
' ' == false                // true
' ' == 0                    // true
```

falsy : null, undefined, "", 0, NaN

truthy: all the rest

Always use === or !== when writing JavaScript!

Functions functions functions...

As soon as you do not know how to do something in JS, write a function.

All functions return something, a function without a return statement returns undefined.

Functions are objects, this means that a function can be stored in a variable, array, or object. Also, a function can be passed to, and returned from, a function.

Functions (are like methods… mostly)

Mostly the same as a method in Java or C#

```
// Java
public static int addFunction (int val1, int val2) {
        return (val1+val2);
}
//JavaScript
funciton addFunction(val1,val2) {
        return (val1+val2);
} // or
let myFunc = function(val1,val2) {
        return(val1 +val2);
}
```

Mostly…

No overloading – last one is always the ‘correct’ one.

No return type but always returns something.

No check for how many arguments we pass in.

Special property arguments to deal with unspecified number of arguments

Callbacks

Used for ajax calls before promises – async programming

Quite tricky to wrap your head around, sort of like recursion

You need to know about them, especially in NodeJS

Promises and generators can be a big help if they feel overwhelming

Objects (which may be or contain.. functions)

Similar to serialized data , such as JSON

```
var myDotaHero  {
    name: 'Luna',
    'typical role': 'Carry',
    skillvalues : {
        str:15,
        dex:18,
        int:16
    }
}
```

Objects part 2

To retrieve a value we use dot or bracket notation

```
var name = myCarry.name;
var nameAlt = myCarry["name"];
var str = myCarry.skillValues.str;
```

We can also add new properties and even functions to our object dynamically

```
myCarry.rival = "Mirana";
myCarry.attack = function () {
        var target = this.rival;
}
```

“this”

“this” - the current state of which the method was invoked off of.

```
function changeColor() {
        return this;                                    // this points to the browser window
}


var myObject = {
    firstName:"John",
    lastName: "Doe",
    fullName: function () {
        return this.firstName + " " + this.lastName;    // this points to the object
    }
}
myObject.fullName();
```

Classes

New keyword – class

```
Class DotaHero {
        constructor () {
        }
        attack() {
                return('attacked');
        }
}
let magnus = new DotaHero();
console.log(Magnus.attack());
```

The prototype design pattern

```
function Car( model, year, miles ) {
        this.model = model;
        this.year = year;
        this.miles = miles;
}

Car.prototype.toString = function () {
        return this.model + " has done " + this.miles + " miles";
};

var myCar = new Car("Volvo");
alert(myCar.toString());
```

Generators

Yield – keyword

JavaScript is single threaded

Generators gives us cooperative code that may be released and then started up again.

```
function* foo() {
var a = yield 1;
var b = yield 2;
var c =yield 3;
}
```

Execellent for async operatoins – looks much nicer than promises or callbacks

So… can I use JS6?

Yes, otherwise this lecture would have been kinda pointless and we would have talked about prototype patterns and the ‘var’ keyword etc.

No, current browsers does not support it yet (Edge(!)being your best bet). But you can use a transpiler to compile your JS6 into older JS that is runnable in all browsers.

Hard to find help since its such a major update, stack overflow has a lot of JS5 help – might be a reason to look to a framework instead.

For a complete list of browser compability to ES5,6 and 7,go to:
https://kangax.github.io/compat-table/es6/

Testing JavaScript

Traditionally it has been hard to accurately test front end development code. Not any more!

Karma

Jasmine

QUnit

Alternatives to JavaScript

Dart (Google - compiles to JavaScript)

TypeScript (MS – compiles to JavaScript)

GWT (Write in Java – compiles to JavaScript)

CoffeScript (Compiles to JavaScript)

The first three examples are made to make JavaScript more strongly typed. This may or may not be what you want since it also removes a lot of the inherited power of the language.

Part 2 – What is it good for?

JavaScript is the language we use to speak to the web browser.

The Document Object Model (DOM)

One way to represent a webpage and manipulate is through the Document Object Model

View the DOM : F12 in the browser of your choice.

JavaScript was made for interacting with the DOM!

Wait for the DOM before running JS-code

```
window.onLoad = function () {
        // excecutes as soon as the whole page is loaded, including images etc
};

document.addEventListener("DOMContentLoaded", function(event){
        // executes as soon as the DOM is loaded
}
```

DOM manipulation- getting elements

By tag name

```
document.getElementsByTagName("input");
```

By id:

```
document.getElementById("id");
```

By class:

```
document.getElementsByClassName("className");
```

Node lists

A nodelist is like an array but is always a resultset from our document.getElements...

```
var nodes = document.getElementsByClass('myClass');
nodes[0].innerHTML = ”my own text”;
for (var i = 0; i < = nodes.length; i++ ) {
        nodes[i].innerHTML = ”my own text”;
};
// or
[].forEach.call(nodes, function(element, index) {
        element.className = ”backlogHeading";
});
```

JQuery and other frameworks

“The problem with JavaScript is not the language itself - it's a perfectly good prototyped and dynamic language. If you come from an OO background there's a bit of a learning curve, but it's not the language's fault.

The real problem is that it's compiled by the browser, and that means it works in a very different way depending on the client.”

So man created frameworks to deal with this… JQuery being the most common as well as a prerequisite for several others.

Why use JQuery?

It seems very simple and easy to understand

The almighty $

Easy to pick up and most of all it helps the programmer to tame the DOM manipulation process in different browsers and platforms.

```
$(“#myId”).addClass(“selectedBackground”);


$(“.myClass”).click(function(){
        alert(“clicked!”);
});
```

Why not to use JQuery

Don’t use it unless you have to support IE8. IE9 can still be managed with polyfills.

It is slow and huge in comparison to vanilla JavaScript.

Anything you can do in JQuery you can do in JavaScript*. Build your own helper functions when needed instead!

* Some things may be a bit difficult though, like $(“#myId”).closest(“li”);

Alternatives to JQuery

Some people really like their $ so if you feel you really need it, there are alternatives such as:

Zepto

MinJS

Build a custom JQuery build with the functionality you need

Other frameworks

There are a LOT of JavaScript frameworks to help or hinder your coding. Some of them need JQuery while others are standalone.

Durandal, UnderscoreJS, Knockout (excellent tutorials), RequireJS, BackboneJS to name a few.

Angular JS(Google), Ember JS are the two of the most popular at the moment. They both feature MVVM-like functionality (The view (HTML) speaking to the Model (our code) and vice versa).

HTML example

<input ng-model="age"/>

Can in angular be used directly in the code as it has the same value as:

$scope.age

The end

I hope you've picked up a few pointers that will help you in your JavaScript transition.

Thank you for your time.

Andreas Ahlgren
aa@computas.com