JavaScript

SWE 432, Fall 2016

Design and Implementation of Software for the Web

GEORGE MASON UNIVERSITY

~~JavaScript~~

ECMAScript 6 (ES6)

SWE 432, Fall 2016

Design and Implementation of Software for the Web

GEORGE MASON UNIVERSITY

Today's Objectives

- Learn some history about JavaScript/ECMAScript
- Understand how to write simple ES6 programs
- Understand how to use simple ES6 libraries
- Embed ES6 in your websites

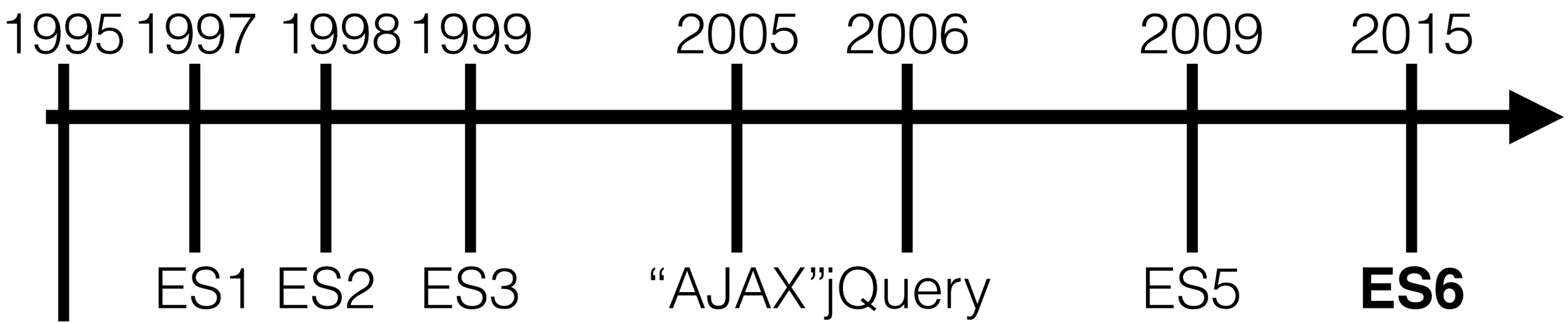
Next **3** lectures will have additional info on ES6, especially sending stuff over the network and manipulating the page

Some great resources on Safari Books Online (http://mutex.gmu.edu:2048/login?URL=http://proquest.safaribooksonline.com/?uicode=viva):
"JavaScript: The Good Parts"
"You Don't know JS: ES6 and Beyond"

http://seecode.run/ for a super easy sandbox!

ES6: Some History

- JavaScript: 1995 at Netscape (supposedly in only 10 days)
  - No relation to Java (maybe a little syntax, that's all)
  - Naming was marketing ploy
  - ECMAScript -> International standard for the language



Then and Now


Netscape: bottom.html (Untitled)
File Edit View Go Bookmarks Options Directory Windows Help
Back Forward Home Reload Images Open Print Find Stop
Netsite: http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/framec
WELCOME TO NETSCAPE
EXPLORING THE NET
COMPANY & PRODUCTS
NETSCAPE STORE
NEWS & REFERENCE
ASSISTANCE
COMMUNITY
EXPLORING THE NET
What's New
What's Cool
Net Directory
Net Search
NETSCAPE STORE
Software
Support
Publications
COMMUNITY
Netscape User Groups
Internet White Pages
COMPANY & PRODUCTS
Netscape Products
Netscape Sales
About Netscape
Netscape Navigator 2.0
Copyright ©1995,1996 Netscape Communications Corporation. All rights reserved.
FREE 60 DAY SERVER TRIAL
internal web sites.
PLATFORM TOOLS AND MORE
Netscape announces the Netscape Development
SEARCH
SPONSOR INDEX
CONTACT INFO
NETSCAPE STORE
TABLE OF CONTENTS
DOWNLOAD SOFTWARE
NETSCAPE GALLERIA
FEEDBACK
Find out more about Frames at How do I do this?. Copyright © 1995 Netscape Communications Corporation


AngularJS — Superheroic Java
https://angularjs.org
ANGULARJS
Home
Learn
Develop
Discuss
Search
ANGULARJS
by Google
HTML enhanced for web apps!
Download AngularJS 1
(1.5.8 / 1.2.30)
Try the new Angular 2
View on GitHub
Design Docs & Notes

Step 0: Embedding ES6 in HTML

- Use the <script> tag to embed ES6 code

```
<script type="text/javascript">
    // Code goes here.
</script>
```

  - or

```
<script type="text/javascript" src="path/to/
file.js"></script>
```

- Script is evaluated once encountered by browser

Basics: Variables

- Variables are *loosely* typed
  - String:
    ```
    var strVar = 'Hello';
    ```
  - Number:
    ```
    var num = 10;
    ```
  - Boolean:
    ```
    var bool = true;
    ```
  - Undefined:
    ```
    var undefined;
    ```
  - Null:
    ```
    var nulled = null;
    ```
  - Objects (includes arrays):
    ```
    var intArray = [1,2,3];
    ```
  - Symbols (named magic strings):
    ```
    var sym = Symbol('Description of the symbol');
    ```
  - Functions (We'll get back to this)
- Names start with letters, $ or _
- Case sensitive
- Can make any variable a constant at declaration:
  ```
  const numConst = 10; //numConst can't be changed
  ```

More Variables

- Loose typing means that JS figures out the type based on the value

```
var x; //Type: Undefined
x = 2; //Type: Number
x = 'Hi'; //Type: String
```

- Variables have block scope - if defined in a function, can only be seen in that function, if defined outside of a function, then global. Can also make arbitrary blocks:

```
{
    let a = 3;
}
//a is undefined
```

Loops and Control Structures

- `if` - pretty standard

```
if (myVar >= 35) {
    //...
} else if(myVar >= 25){
    //...
} else {
    //...
}
```

- Also get `while`, `for`, and `break` as you might expect

```
while(myVar > 30){
    //...
}

for(var i = 0; i < myVar; i++){
       //...
       if(someOtherVar == 0)
          break;
}
```

Operators

```
var age = 20;
```

Operator	Meaning	Examples
==	Equality	`age == 20` `age == '20'`
!=	Inequality	`age != 21`
>	Greater than	`age > 19`
>=	Greater or Equal	`age >= 20`
<	Less than	`age < 21`
<=	Less or equal	`age <= 20`
===	Strict equal	`age === 20`
!==	Strict Inequality	`age !== '20'`

Annoying

Functions

- At a high level, syntax should be familiar:

```
function add(num1, num2) {
    return num1 + num2;
}
```

- Calling syntax should be familiar too:

```
var num = add(4,6);
```

- Can also assign functions to variables!

```
var magic = function(num1, num2){
    return num1+num2;
}
var myNum = magic(4,6);
```

- Why is this cool?

Default Values (ES6)

```
function add(num1=10, num2=45) {
    return num1 + num2;
}

   var r = add(); // 55
var r = add(40); //85
var r = add(2,4); //6
```

Rest Parameters

```
function add(num1, ... morenums) {
    var ret = num1;
    for(var i = 0; i < morenums.length; i++)
        ret += morenums[i];
    return ret;
}
```

```
add(40,10,20); //70
```

=> Arrow Functions

- Simple syntax to define short functions *inline*
- Several ways to use

Parameters

```
var add = (a,b) =>{
            return a+b;
          }
```

```
var add = (a,b) => a+b;
```

If your arrow function only has one expression, ES6 will automatically add the word "return"

Objects

- What are objects like in other languages? How are they written and organized?
- Traditionally in JS, no *classes*
- Remember - JS is not really typed… if it doesn't care between a number and a string, why care between two kinds of objects?

```
var profJon = {
    firstName: "Jonathan",
    lastName: "Bell",
    teaches: "SWE 432",
    office: "ENGR 4322",
    fullName: function(){
        return this.firstName + " " + this.lastName;
    }
};
```

Working with Objects

```
var profJon = {
    firstName: "Jonathan",
    lastName: "Bell",
    teaches: "SWE 432",
    office: "ENGR 4322",
    fullName: function(){
        return this.firstName + " " + this.lastName;
    }
};
```

Our Object

```
console.log(profJon.firstName); //Jonathan
console.log(profJon[“firstName”]); //Jonathan
```

Accessing Fields

```
console.log(profJon.fullName()); //Jonathan Bell
```

Calling Methods

```
console.log(profJon.fullName); //function...
```

Prototypes

Effectively allow you to define the constructor for a class

```
function Faculty(first, last, teaches, office)
{
    this.firstName = first;
    this.lastName = last;
    this.teaches = teaches;
    this.office = office;
    this.fullName = function(){
        return this.firstName + " " + this.lastName;
    }
}
```

```
var profJon = new Faculty("Jonathan", "Bell", "SWE432",
"ENGR 4322");
```

Remember… There's no Class!

```
var profJon = {
    firstName: "Jonathan",
    lastName: "Bell",
    teaches: "SWE 432",
    office: "ENGR 4322",
    fullName: function(){
        return this.firstName + " " + this.lastName;
    }
};
```

Our Object

```
profJon.officeHours = "Tuesdays 10:30-12:00";
```

Lazily creates a new property and sets it

```
delete profJon.office;
```

Deletes a property

JSON: JavaScript Object Notation

Open standard format for transmitting *data* objects.

- No functions, only key / value pairs
- Values may be other objects or arrays

```
var profJon = {
    firstName: "Jonathan",
    lastName: "Bell",
    teaches: "SWE 432",
    office: "ENGR 4322",
    fullName: function(){
        return this.firstName + " " + this.lastName;
    }
};
```

Our Object

```
var profJon = {
    firstName: "Jonathan",
    lastName: "Bell",
    teaches: "SWE 432",
    office: "ENGR 4322",
    fullName: {
        firstName: “Jonathan”,
        lastName: “Bell”}
};
```

JSON Object

Interacting w/ JSON

- Important functions
- JSON.parse(jsonString)
  - Takes a *String* in JSON format, creates an *Object*
- JSON.stringify(obj)
  - Takes a Javascript *object*, creates a JSON *String*
- Useful for persistence, interacting with files, debugging, etc.
  - e.g., console.log(JSON.stringify(obj));

Arrays

- Syntax similar to C/Java/Ruby/Python etc.
- Because JS is loosely typed, can mix types of elements in an array
- Arrays automatically grow/shrink in size to fit the contents

```
var students = ["Alice", "Bob", "Carol"];
var faculty = [profJon];
var classMembers = students.concat(faculty);
```

Arrays are actually objects… and come with a bunch of "free" functions

Special Array Functions

- Length
```
var numberOfStudents = students.length;
```
- Join
```
var classMembers = students.concat(faculty);
```
- Sort
```
var sortedStudents = students.sort();
```
- Reverse
```
var backwardsStudents = sortedStudents.reverse();
```
- Map
```
var capitalizedStudents = students.map(x=>x.toUpperCase());
//["ALICE","BOB","CAROL"]
```

For Each

- ES6 provides 2 handy ways to loop over arrays and objects with for each
- For **of** (iterates over values):

```
for(var student of students)
{
    console.log(student);
} //Prints out all student names
```

- For **in** (iterates over keys):

```
for(var prop in profJon){
    console.log(prop + ": " + profJon[prop]);
}
```

Output:
firstName: Jonathan
lastName: Bell
teaches: SWE 432
office: ENGR 4322

Arrays vs Objects

- Arrays are Objects
- Can access elements of both using syntax

```
var val = array[idx];
```

- Indexes of arrays must be integers
- Don’t find out what happens when you make an array and add an element with a non-integer key :)

Exercise: Getting Our Feet Wet With JSON

https://jsfiddle.net/4sgz8dn3/

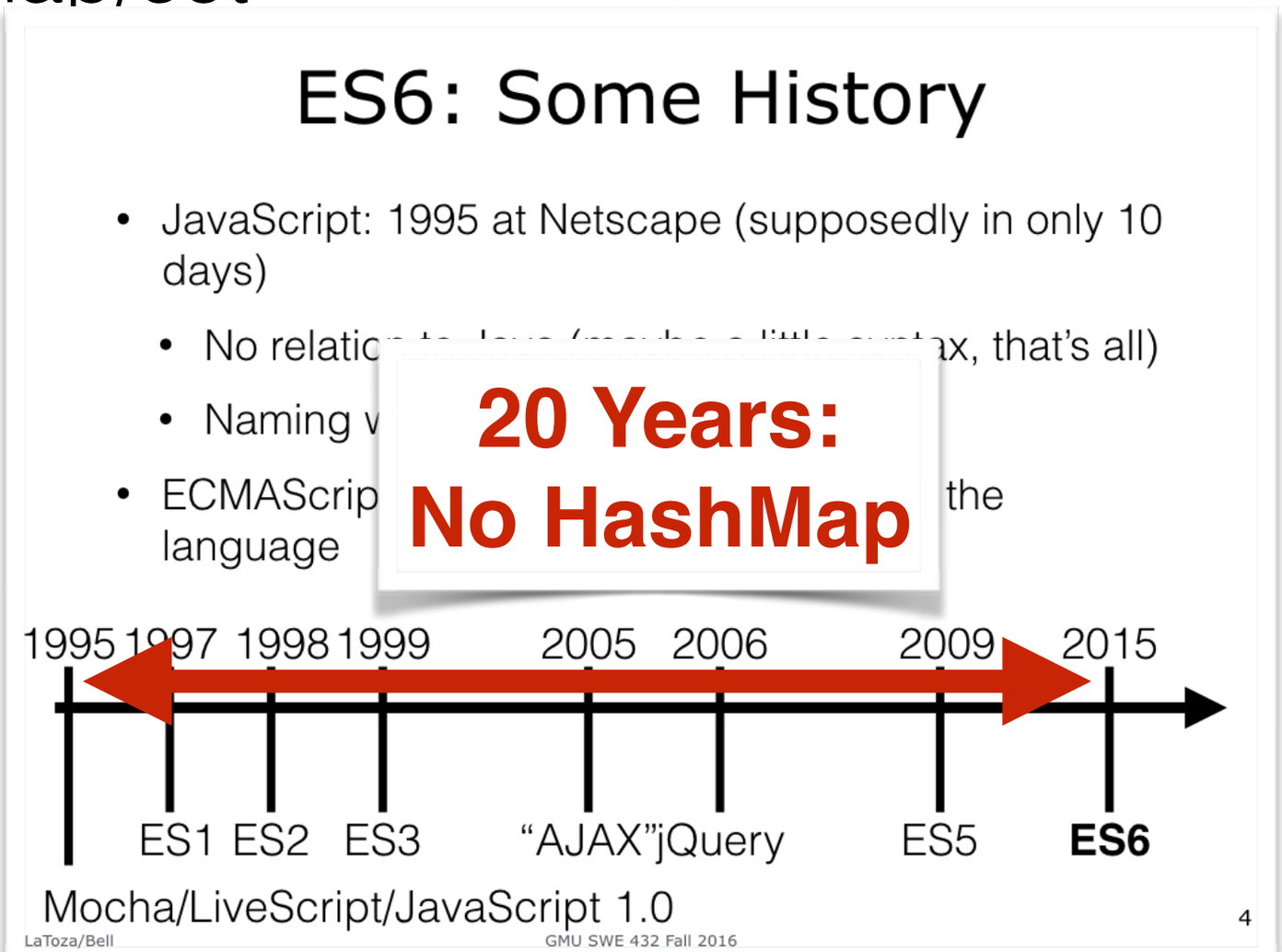
String Functions

- Includes many of the same String processing functions as Java
- Some examples
  - var stringVal = ‘George Mason University’;
  - stringVal.endsWith(‘University’) // returns true
  - stringVal.match(….) // matches a regular expression
  - stringVal.split(‘ ‘) // returns three separate words

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

ES6 Collections

- Map, set (compare to HashMap, HashSet)
- WeakMap, WeakSet
  - When the element is deleted, it disappears from the map/set



ES6 Collections

- Map, set (compare to HashMap, HashSet)
- WeakMap, WeakSet
  - When the element is deleted, it disappears from the map/set

```
let m = new Map()
m.set("hello", 42)
m.set(s, 34)
m.get(s) === 34
m.size === 2
for (let [ key, val ] of m.entries())
  console.log(key + " = " + val)
```

Node.js Getting Started

- Download and install it: https://nodejs.org/en/
  - We recommend v4.5.0 LTS (LTS -> Long Term Support, designed to be super stable)
- Demo: Hello world server
- Demo will show:
  - Using package manager to get a package (express)
  - Running a simple node application

Demo: Hello World Server

1: Make a directory, myapp

2: Enter that directory, type **`npm init`** (accept all defaults)

3: Type **`npm install express --save`**

4: Create text file app.js:

```
var express = require('express');
var app = express();
var port = process.env.port || 3000;
app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(port, function () {
  console.log('Example app listening on port' + port);
});
```

5: Type **`node app.js`**

6: Point your browser to http://localhost:3000

Creates a configuration file for your project

Tells NPM that you want to use express, and to save that in your project config

Let's not worry about JavaScript syntax until next Thursday!

Runs your app

Exercise: Arrays and Node.JS

http://bit.ly/2cEKHZu

What's next?

- Interacting with HTML
- Interacting with remote clients/servers
- And more!