# JAVASCRIPT - JAVASCRIPT - JAVASCRIPT -

Listens for events: move mouse, hover, click, scroll, press key

Script tag just before the closing </body> <script> </script>

Close the tag or brower will think ALL the html is a javascript

$(element).on(event-type, thing-to-be-done);

$("button").on("click", function(){ alert("clicked!") } )

When I click on the button, the function alert will display “clicked!”

/\* Comment \*/ vs // single line comment

## Logics and Basics

### Operators, Variables

Modulus: reste de la division euclidienne: 15 % 4 // 3

NOT: ! OR : || AND: &&

&& true if everything is true

false && false -> false

or returns the first falsy value (either false, or null, NaN, etc)

null && ‘user’ -> null (is first falsy value)

|| true if one of them is true

or returns the first truthy value

null || ‘user’ -> ‘user’

! unitary operator, returns opposite of truthy/falseness

!!(stuff) -> original truthiness of stuff

=== fully equal in type and value; == looks at value but not type

5 === ‘5’ is false; 5==’5’ is true

!== is strict inequality, != looks at value not type

<http://dorey.github.io/JavaScript-Equality-Table/>

Recursive:

**Long-Hand Syntax Short-Hand Syntax**

**x = x + 1 x += 1**

**x = x - 5 x -= 5**

**x = x \* 2 x \*= 2**

**x = x / 10 x /= 10**

**x = x % 10 x %= 10**

**x = x + 1 x = x++**

null: created but not assigned a value

### If, then, ternary operator

Ternary operator: for IF, THEN, ELSE shortcut

expression ? value\_if\_truthy : value\_if\_falsy

(10>7) ? “bien sur” : “impossible” gives …. “Bien sur”

Syntax

if (x > 10) {

x += 10;

y += 10;

} else if (x > 5) {

x += 5;

} else {

whatever;

}

Or switch mode

switch (2 \* x) {

case 2:

y = 49;

break;

case 4:

y = 37;

break;

default:

y = 1;

}

Estimate switch… case = if/then … default = else (therefore not necessary)

### Loops: while & for

var x = 10;

while (x > 5) {

x -= 2;

}

Will return 4

for (initialization; condition; finalExpression) {

// A block of code.

}

var x = 10;

for (var i = 0; i < x; i += 1) {

console.log('HELLO');

}

Will write HELLO 10 times:

Starting with i = 0,

While i<x, execute instruction (write HELLO),

Then do i+=1

Break -> stop the loop. Same way as return works, see below

### Functions

Declaration:

function name(x;y;z) {

// Body;

}

Return -> stop the function now -> useful when having an if, so you stop the function if the condition is true

### On Numbers

Integers have the upper limit. Use Long or BigInt

NaN is still of type ‘number’ -> use isNan(x)

Math.pow, Math.sqrt, Math.random

Sort in ascending order:

var points = [40, 100, 1, 5, 25, 10];  
points.sort(function(a, b){return a-b});

The result of points will be:1,5,10,25,40,100

Use b-a to get descending order

Round 8.111111 to 3 decimals

NEW 8.111.toFixed(3)

var result=Math.round(8.111111\*1000)/1000 //returns 8.111

* multiply by 10^3 gets 8111.111, and Math.round of this gives 8111, which divided by 10^3 is what we want

### On String

String.toLowerCase() // str.toUpperCase()

Str.charAt returns character in a specific place

Substring character at a place, or between 2 places

Number(‘1’) -> convert string to a number

Split strings returns an array

‘this text to split’.split(‘ ‘) -> [this, text, to, split]

‘hey’.split(‘e’)[0] -> ‘h’ // ‘hey’.split(‘e’)[1] -> ‘y’

Count occurrences of a letter in a string

I like the idea of using a match better, but it is slower:

console.log(("str1,str2,str3,str4".match(/,/g) || []).length); //logs 3

console.log(("str1,str2,str3,str4".match(new RegExp("str", "g")) || []).length); //logs 4

Use a regular expression literal if you know what you are searching for beforehand, if not you can use the RegExp constructor, and pass in the g flag as an argument.

match returns null with no results thus the || []

The original answer I made in 2009 is below. It creates an array unnecessarily, but using a split is faster (as of September 2014). I'm ambivalent, if I really needed the speed there would be no question that I would use a split, but I would prefer to use match.

Old answer (from 2009):

If you're looking for the commas:

(mainStr.split(",").length - 1) //3

If you're looking for the str

(mainStr.split("str").length - 1) //4

## Array, Object, Prototype

### Array

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array>

var myFriends = ['ellen', 'mary', 'doug', 'pat'];

myFriends[0] -> ‘ellen’

myFriends.length -> 4

To get the (nth) last item: myFriends[myFriends.length-1] -> ‘pat’

var arrayOfArrays = [['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h', 'i']]; arrayOfArrays[0][0] = ‘a’

; arrayOfArrays[1][2]= ‘f

.indexOf(): searching in arrays

var animals = ['bear', 'beetle', 'boa'];

animals.indexOf('boa'); // Evaluates to 2 (index 2 = 3rd position)

Push and Pop: add and remove the last element

Animals.push(‘rat’) -> add ‘rat’ as 4th item (index 3)

Animals.pop() -> remove last item

Shift: take form first position

Unshift: put in the first position

Splice: remove from an array to put to a new one

friends.splice(1, 2): from position 1 (=2nd), remove 2 elements and put them in a new array

array,reverse reverse the array

Split: transform string into an array, splitting according to a rule

Var string = ‘I am a string’

String.split() -> [‘I am a string’]

String.split(‘ ‘) -> ["i", "am", "a", "string"]

But careful, string.split('a') -> ["i ", "m ", " string"]

**To reverse from Split,**

**Join**() method joins all elements of an **array** into a string.

Array.sort: sort the array

Map: change every element of array according to a formula;

function square(x) { return x \* x; }

var resultingArray = [1, 2, 3].map(square);

/\*\*\* Must use RETURN otherwise function will return undefined \*\*\*/

ForEach will execute a function, for each element in the array

It will NOT return a new array but just an undefined. Use forEach if you want to log stuff in console

For each is same as doing a for-loop on each element of the array

for (var i=0; i<friends.length; i++) {

console.log('Hello ' + friends[i]) }

=== friends.forEach(function(buddy)

{console.log('Hello' + buddy) })

var resultingArray = [1, 2, 3].forEach(square)

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach>

Every wil do foreach and stop the first a value is Falsy

Some the same and stops whem value is TRuthy

- var THRESHOLD = 12; var v = [5, 2, 16, 4, 3, 18, 20]; var res; res = v.every(function(element, index, array) { console.log('element:', element); if (element >= THRESHOLD) { return false; } return true; }); console.log('res:', res); // logs: // element: 5 // element: 2 // element: 16 // res: false

Filter

Loops through an array and create a new array keeping only elements matching the condition

Ex: let’s keep only items > 50

var numbers = [12, 5, 8, 130, 44];

var biggers = numbers.filter(function(item) {

return item > 50

});

-> result is array [130]

Reduce

Will perform an operation using , successively, the value of each element in the array

<https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce>

WARNING!! It will **always** use the first 4 parameters as below, they are native to reduce, this is how it works

[0, 1, 2, 3, 4].reduce(function(previousValue, currentValue, index, array) {

console.log(previousValue, currentValue, index, array);

return previousValue + currentValue;

});

0 1 1 [0, 1, 2, 3, 4] *// is the console.log being looped*

3 1 2 2 [0, 1, 2, 3, 4] *// is the console.log being looped*

3 3 3 [0, 1, 2, 3, 4] *// is the console.log being looped*

3 6 4 4 [0, 1, 2, 3, 4] *// is the console.log being looped*

RETURNS 10 (summed 1+2+3+4)

### Associative Arrays / Objects

Associate a key to each element of the array.

Create using {}, combination of key-value pairs

var lunches = { 'Josh' : 'pasta','Floyd' : 'salad','Matt' : 'tuna sandwich','Shannon' : 'soup' };

var candidateData = 'name' : "John Doe",'age' : 32,'isFullTime' : true, 'pastEmployers' : ['Microsoft','Google','Amazon'],yrsExperience' : {'ruby' : 3,'java' : 6,'javascript' : 5}}

Then access using normal array[] , or array.key

There is no indexing anymore, we access values based on the key:

candidateData[‘name’] === ‘John Doe’

Same as candidateData.name

candidateData['yearsExperience']['javascript'] === 5

And to add a new key-value pair there is no concept of number or order, just type in the new pair:

candidateData[‘Hobby’]=’Basket’

### Objects, Prototypes

-> Object is unique, repeating for each new house is long !!

var house1 = { sqfoot: 3000, bathrooms: 8, bedrooms: 10 }

var house2 = etc…

-> Create various instances of an object

function House(**sqfoot**, **bathrooms**, **bedrooms**) {

this.sqfoot = **sqfoot**;

this.bathrooms = **bathrooms**;

this.bedrooms = **bedrooms**;

}

To create 2 houses I simply do

var house1 = new House(3000, 8, 10); //*CONVENTION IS TO CAPITALISE*

var house2 = new House(1500, 1, 2);

var house3 = new House(2500, 4);

var house4 = new House(undefined, 1, 1);

house1 -> House {sqfoot: 3000, bathrooms: 8, bedrooms: 10}

house2 -> House {sqfoot: 1500, bathrooms: 1, bedrooms: 2}

house3 -> House {sqfoot: 2500, bathrooms: 4, bedrooms: undefined}

house4 -> House {sqfoot: undefined, bathrooms: 1, bedrooms:1 }

to populate the bedrooms in house3….

house3.bedrooms = 6

house1 is an INSTANCE of House

house1 instanceof String -> true

Bit different when one parameter of the object is a function

// CF below for faster way using Prototype //

var Person = function(firstName, lastName) {

OR

function Person(firstName, lastName) {

this.firstName = firstName;

this.lastName = lastName;

this.fullName = function() {

return ('Hello '+ firstName + ' ' + lastName);

}

}

var jeremy = new Person(‘Jeremy’, ‘Marer’)

jeremy.fullName**()** -> Jeremy Marer ***// fullName is a function so use ()***

jeremy is an instance of Person

jeremy instanceof Person -> true

**More efficient using Prototype**

**Also enables to add new key/value pairs to an object**

function Person(firstName, lastName) {

**this.firstName** = firstName;

**this**.**lastName** = lastName;

}

Person.prototype.fullName = function() {

return ('Hello '+ **this**.**firstName** + ' ' + **this.lastName**);

}

*// proto is a function! Don’t forget to RETURN something*

### Iterating through object keys and values

Var test = {“name”: \j};

Object.keys(test) list the keys of an object

For…in iterates through properties of an object

for (var prop in obj) {

do xxx

*// prop always refer to the property currently being examined*

*// obj always refer to the object*

*// obj[property]*

}

cf pre-made functions

## Debugger

*// Only works if the console is open !!*

Write debugger; under the line you want to stop at,

Then load the console in Chrome

Put break points on lines where you want the code to stop

Play will load code until next break

* Step over will run the next function and not go line by line
* Step into will send you *inside* the next function so you can test it
* step over will run the function line by line

Step out will send you out of the function, when you know the rest is good

## DOM Document Object Model

On html page one div has id='js-repo-pjax-container'.

Rigt-click ‘inspect element’ -> go to console elements list in the page tree

Use Javascript to assign this id to a var called element:

var element = document.getElementById('js-repo-pjax-container')

To change the text and background color inside the div

element.innerHTML = 'this has been changed by DOM manipulation'

element.style.backgroundColor = 'red';

**Create elements on a page, = in the DOCUMENT**

var element = document.createElement('div') *//declare a div*

var parent = document.getElementById('body'); *//create parent element, here we gave <body =id”body”>*

parent.appendChild(element); // put our div inside the body

OR declare a var for this child: var child = parent.appendChild(element);

**Find all elements with class=”students”**

**var students = document.getElementsByClassName('students');**

body.children list all children within body

body.childNodes similar with more info

**Use children to “navigate”through the tags.**

EG A form has an id, and then 2 inputs inside = 2 children

Text is the value of the first input form

var former = document.getElementById('former');

former.addEventListener('submit', function(e) {

alert(this.children[0].value);

Where am I (url): window.location.href

Find image with id=’diagram’ then change the source of the image

var element = document.getElementById('diagram')

element.setAttribute('src', `img/dom\_basic.png`)

### Local Storage and JSON objects

Set, Create, Remove key-value pairs

localStorage.setItem('myCat', 'Tom') 🡪 Storage {myCat: "Tom", length: 1}

localStorage.getItem('myCat') 🡪 "Tom"

localStorage.removeItem('name')

localStorage.clear() 🡪 RESET

Replace value by doing new setItem

***!! Only pass strings as values will be stored as Strings***  (except Booleans)

localStorage.setItem('myArray', [1,2,3]) 🡪 {myArray: "1,2,3", length: 1}

**Object will be messed up, hence we use JSON methods – stringify**

**JSON to Add objects to local values**

var bob = {name: 'Bob', age: '81'}

var stringBob = JSON.stringify(bob) 🡪 "{"name":"Bob","age":"81"}"

localStorage.setItem('coolPerson', stringBob)

🡪Storage {coolPerson: "{"name":"Bob","age":"81"}", myArray: "1,2,3", length: 2}

**JSON parse to get the Object back, and to get values from the object**

var stringBob = localStorage.getItem('coolPerson') 🡪 "{"name":"Bob","age":"81"}"

(normal, everything in localStorage is a string!)

JSON.parse(stringBob).age 🡪 "81"

On Arrays

Students 🡪 ["bob", "jo", "sally"]

stringStudents = JSON.stringify(students) 🡪 "["bob","jo","sally"]"

JSON.parse(stringStudents) 🡪 ["bob", "jo", "sally"]

JSON.parse(stringStudents)[0] 🡪 "bob"

## Javascript Events

Syntax: Element.addEventListener(event, function, useCapture)

Create event when we click on an element with id=’event-click’

var eventClickButton = document.getElementById('event-click');

eventClickButton.addEventListener('click', function() {

alert('Ive been clicked!');

});

Same if working on an object. Example change picture on mouse hover

var billMurray = document.getElementById(‘bill’);

**billMurray**.addEventListener(‘mouseover’, function(e) { *//when hovering over pic*

**this**.setAttribute(‘src’, ‘new url/300/200’);

**//***better* ***this****.src = new url*

})

var billMurray = document.getElementById(‘bill’);

**billMurray**.addEventListener(‘mouseout’, function(e) { *//when stop hovering*

**this**.setAttribute(‘src’, ‘new url/300/200’);

})

***//Note the (e) inside function; shortcut for “event”, it links back to this locally specific event for this specific object – eg above, refers to mouseout event on billMurray***

***// If calling an existing function inside an event, DON’T use ()***

thingy.addEventListener('click', oneFunction, false);

function oneFunction (fgfg) {fgh} ;

***// Can ONLY create events on one element at a time –***

***// getElementsByClassName or Tag returns an ARRAY, se select getELxxx[i]***

**To program multiple events, we need to use a for loop**. Say we have multiple radio buttons with class=’radio\_Event’ and want to uncheck them

Var radios = document.getElementsByClassName(‘radio-event’) *// all elements with this class*

For var (i=o; i<radios.length; i++) {

***// individually adding the events to each element: radios is an array***

Radios[i].addEventListener(‘change’, function(e) {

(for var k=0; k<radios.length; k++) {

radios[k].checked=false;

}

})

}

Event target to check what item is impacted by the event.

EG an <ul> with a <li> of vegetables. Wanna now what item in the list I clicked on

var vegetables = document.getElementById('vegetables');

vegetables.addEventListener('click', function(e) {

alert(event.target.textContent)

})

Functions to make Jscript asynchronous

setTimeout(function() {

alert(‘I will run after 3,500 milliseconds’);

}, 3500);

setInterval(function() {

alert(‘I will run after 1,000 milliseconds’);

}, 1000)

### Clear the timers: clearTimeout() and clearInterval()

### List of possible events

<https://developer.mozilla.org/en-US/docs/Web/Events>

‘click’, ‘dblclick’, ‘mouseup’, ‘mousedown’, ‘mouseover’, ‘mouseout’

‘onscroll’

‘focus’ when you click into an input field

‘blur’ when outside of a submit text box

**On forms and buttons**

**‘change’ when a form is changed**

Submitting forms: submit

var formSubmitEvent = document.getElementbyId(‘form\_event’);

forSubmitEvent.addEventListener(‘submit’, function(e) {

e.preventDefault();

})

<https://developer.mozilla.org/en/docs/Web/HTML/Element/form>

## jQuery

jQuery is a load of functions to make shortcuts on javascript

***Always returns an ARRAY – works like CSS seclectors***

Always put the script to jQuery before all other js files (still bottom of body)

Navigating through elements

$(‘thing’) = document.getElementsByTagName / ClassName / Id

$(‘tag’) - $(‘.class’) - $(‘div .class’) - $(‘#id’)

***Can only do a jQ function on a jQelement!!! (ie an array)***

var cities = $(‘li’) -> [‘barcelona’, ‘madrid’]

Console.log( cities[0].text() )

🡪 ERROR because citie[0] is NOT jQuery object, while .text() is

Console.log( $(cities[0].text() ) ) -> will work!

Find an element, even within another element

$('section').find('span')

Next element nested **at same level //** Prev returns the one before, **nested at same level**

$('ul').next() // $('ul').prev()

Navigating through “family”

sibling() .parent() .children()

***//mind the syntax and space on children***

$('p').first() -> fisrt paragraph; // $(‘p’).last()

$(‘section:first) -> first of type section

${‘section :first-child’) -> first-child of the parent, in this case section

$('ol :nth-child(2)') -> second child in the <ol>

$('li:nth-child(2)') -> ALL <li> which are a second child

If we omit the argument, .index() will return the position of the first element within the set of matched elements in relation to its siblings

**EACH to loop through elements**

$('li').each(function(index, element) {

/*/ index ->index of the element in the array*

*// element -> the jQuery element*

console.log(index, $(element).text() ) ;

***// .text() to convert to text – remember all $ elements are arrays***

})

Find all the classes, or attributes, of an object

$('.secret\_message').attr('class')

Actions

APPEND / PREPEND

inserts the specified content as the l**ast child**

<ul class="cities">

<li>Madrid</li>

<li>Barcelona</li>

</ul>

var newCity = 'Granada';

$('.cities').append('<li>' + newCity + '<li>')

Can also select element on page and append to another

$( ".container" ).append( $( "h2" ) );

List all text inside the <li> above

$(‘.cities li).each( function(index, element) {

console.log( $(element).text() )

***// only call a jQuery function on a jQuery element !!***

Remove element .remove()

Empty “inside”an element .empty()

$foo.empty() === $foo.children().remove()

Remove/add class

$('section').removeClass('no\_js') / $('.message').addClass('hidden')

t class on/off

$('.secret\_message').toggleClass('hidden') -> if not there, add it; if there, remove it

Change text: $('header > h1').text('My Page');

Change the html content : $('blockquote').html('<span>no quote</span>');

Change parameters… use CSS!

$('h2').css('text-decoration', 'underline');

Combine jQuery and Jscript!

$(this).attr('src', "http://url") === this.src="http:/url”);

Change text to uppercase in an input box

FORMS - FORMS

If you want to quickly gather all the input field name and value pairs from a form you should use .serialize()

Clear the content of a form: $('#form\_name').trigger('reset');

### Event Delegation

If you are using jQuery to set events then you should consider using `jQuery#on` syntax and event delegation when possible.

Adding `click` handlers to each `tr` should be avoided.

```javascript

// bad

$("table#myTable tr").each(function (index, el) {

$(el).on("click", function (e) {

console.log("clicked tr", index);

});

});

```

Add a `click` handler to the container of the collection.

```javascript

$("table#myTable").on("click", "tr", function (e) {

console.log(this);

console.log(e.target);

console.log(e.currentTarget);

});

```

### Adding Data To Elements

You can add `data` to an element using `jQuery#data` method.

```javascript

var $myDiv = $("div#greeting");

$myDiv.data("count", 1);

$myDiv.data("count") // => 1

$myDiv.data(); // => { count: 1 }

```

Calling Events and Listerners

var setUpEventListeners = function() {

$('#helloButton').on('click', showPortfolio);

}

**!!! DON’T PASS ARGUMENTS INTO FUNCTION WHEN DEFINING EVENTS !!!**

**NO!! $('.greeting').on('click', showGreeting($(this).attr('id')));**

**YES !! $('.greeting').on('click', function(){ showGreeting($(this).attr('id') );**

BEST PRACTICE

1. create a Var/function to set all event listeners
2. define the functions for each events separately, for clarity
3. wrap the function for events into the doc.ready event, to make sure the events are NOT run before the doc is loaded **// Cannot call docs from the DOM… until the DOM is loaded**!

$(document).ready(function(){

setUpEventListeners();

initialize();

})

var setUpEventListeners = function() {

$('#helloButton').on('click', showPortfolio);

$('.aClass').on('click', anotherFunction);

}

var showPortfolio = function() {

console.log(event);

alert('hello')

var anotherFunction () {}

}

var initialize = function() {

$('section').removeClass('no\_js');

$('.message').addClass('hidden');

}

# AJAX – AJAX – AJAX - AJAX

<http://api.jquery.com/jquery.ajax/>

Asynchronous javascript and xml

EG youtube: only load the comments when you scroll down past a point

API: Point is to receive a JSON object from another source and use it in our own code

All in JSON – good to use “double-quotes”

*// Open python server python -m SimpleHTTPServer 9000*

*// Create an AJAX request (AJAX built-in in Javascript)*

### This is the crude way to understand the logic, see at bottom MUCH FASTER WITH JQUERY

Week5 day 4, rest\_countries

function ajaxRequest(method, url){

var request = new XMLHttpRequest();

request.open(method, url);

request.send();

return request;

}

*// we only want request status 4 and ok*

function successfulRequest (request){

return if (request.readyState === 4 && request.statusText == 'OK')

}

function getRegions(event) {

var regions = []; *// will be used for storage*

var request = ajaxRequest('GET', 'https://restcountries.eu/rest/v1/all');

*// Remember JS is synchronous but we need to wait until the request is fully processed in order to do something*

request.onreadystatechange = function() {

*// console.log(request); => Loads of various request objects, all looking similar (a hash) with long JSON objects*

if successfulRequest(request)) {

regionSelect.innerHTML = ''

*// regionSelect.innerHTML ='<option value="default">-- Select your region --</option' => was much quicker than creating the optionDefault below...*

var response = JSON.parse(request.response) *// is a long JSON string => parse it to get an Array: each array[i] lists infos on a country -> we need the region, and we store it in the regions array*

for (var i=0; i < response.length; i++) {

*// need to avoid duplicate regions*

if (regions.indexOf(response[i].region) === -1 && response[i].region.length > 0) {

regions.push(response[i].region);

*// now add it to the dropdowm menu: "select" made of "options"*

var optionRegion = document.createElement('option');

optionRegion.setAttribute('value', response[i].region);

optionRegion.innerHTML = response[i].region;

regionSelect.appendChild(optionRegion);

}

}

*// just add a "select region" to the dropdown*

var optionDefault = document.createElement('option');

optionDefault.setAttribute('value', 'default');

optionDefault.innerHTML = '-- Select your region --';

regionSelect.insertBefore(optionDefault)

} *// end if readyState == 4*

} // *end request.onreadystatechange*

} *// end getRegion*

var getRegionsButton;

var regionSelect;

document.addEventListener('DOMContentLoaded', function() {

console.log('heyyyy');

*// Define our variables*

getRegionsButton = document.getElementById('get-regions');

regionSelect = document.getElementById('region-select');

*// Add Event Listeners*

getRegionsButton.addEventListener('click', getRegions);

});

### AJAX IS MUCH FASTER WITH JQUERY

Week5 day 4, rest\_countries

function getRegions(event) {

var regions = [];

// Create an AJAX request (AJAX shortcut in jQuery)

$.ajax({

type: 'GET',

url: 'https://restcountries.eu/rest/v1/all'

}).done(function(response) {

// .done is like the "onreadystatechange thing" + check successful request

// jQ already does JSON.parse on response so it is an objet => JQUERY OBJECTS ARE ARRAYS!!

$('#region-select').empty();

$('#region-select').append('<option value="default"> -- Select region -- </option>')

$.each(response, function(index, item) {

if ($.inArray(item.region, regions) == -1 && item.region.length >=1) {

regions.push(item.region);

$('#region-select').append('<option value="' + item.region + '">' + item.region + '</option>')

}

})

})

}

function getCountries(event) {

var regionName = $(this)[0].value // remember jQuery returns an Array !

$.ajax({

type: 'GET',

url: "https://restcountries.eu/rest/v1/region/"+ regionName

}).done(function(response) {

$('#country-select').empty();

$('#country-select').append('<option value="default"> -- Select country -- </option>')

$.each(response, function(index, item ) {

$('#country-select').append( '<option value="' + item.name + '">' + item.name + '</option>')

})

})

}

function getCountryData() {

var countryName = $(this)[0].value;

$.ajax({

type: 'GET',

url: 'https://restcountries.eu/rest/v1/name/' + countryName

}).done(function() {

$('#results').empty();

$('#results').append("<h4>" + countryName + "</h4>");

$('#results').append("<ul id='description'></ul>");

})

}

$(document).ready(function(){

$('#get-regions').on('click', getRegions);

$('#region-select').on('change', getCountries);

$('#country-select').on('change', getCountryData)

})

### Shorter syntax for GET and POST requests

$.get("/articles").

done(function (data) {

console.log(data);

});

$.post("/books", {

book: {

title: "The Giver",

author: "Lowis Lowry"

}

}).

.done(function (data) {

console.log(data);

});

```

# NODE.js - NODE.js - NODE.js

*See w8\_d1 classwork for the basics ; see homework solution for more advanced way to wrap ajax requests in objects*

Server-side code: the JS is not run on the client, we can’t see it in the browser

Non-blocking: bits of code are asynchronous, they do not stop the execution of the rest of events

Run ‘node’ in the terminal

‘node debug file’ to stop at the debugger points

‘repl’ to work on console outside of the context of the file code

npm = node package manager (similar to ruby gem)

Node is very long-winded to creat routes -> EXPRESS is what we use to create all the various routes in Node (“web framework”)

In nodes would be sth like

var http = require('http');

http.createServer(function (req, res) {

// req = request; res = response

console.log(req.method, req.url);

if (req.url === '/' && req.method === 'GET') {

res.writeHead(200, {'Content-Type': 'text/plain'});

res.end('Hello World\n');

} else if (req.url === '/about' && req.method === 'GET') {

res.writeHead(200, {'Content-Type': 'text/plain'});

res.end('About Jeremy\n');

// } else if (req.url === '/class/' + name) {

// res.writeHead(200, {'Content-Type': 'text/plain'});

// res.end('Hello' + name + '\n');

} else {

res.writeHead(404, {'Content-Type': 'text/plain'});

res.end('Error page not found\n');

}

}).listen(1337, 'localhost');

console.log('Server running at <http://localhost:1337/>');

## Express – web framework

npm = node package manager (similar to ruby gem)

npm install --save express body-parser ejs morgan socket.io

body parser used for post requests – it makes the req.body easy to read object

ejs for the views

morgan for middleware

socket.io if I need the real-time engine

npm install –g nodemon

no need to refresh the server – make this one global with –g

so now we run the server we `nodemon app.js`

Will install express into my package.json file (like the gemfile in rails)

So if you send the code someone they just run npm install

<https://nodejs.org/api/http.html#http_http_incomingmessage>

var express = require('express');

var app = express();

var port = 3000'

var burgers = ['Double cheese', 'Big Mac', 'Whooper'];

app.get('/', function(req, res) {

res.send('hello world');

});

app.get('/burgers', function(req, res) {

res.send(burgers.join(', '));

});

// MAKE IT DYNAMIC

app.get('/burgers/:id', function(req, res) {

console.log(req);

res.send('we all love ' + burgers[req.params.id - 1]);

});

// QUERY - localhost:3000/thank?name=Jeremy

app.get('/thank', function(req, res) {

console.log(req);

var name = req.query.name;

res.send('hey ' + name)

})

// START THE SERVER

app.listen(port, function() {

console.log('server started on http://localhost:3000');

});

## EJS – Embedded JavaScript

That’s how we get views in express

npm install ejs –save

npm install morgan –save, to handle middleware

app.set('/views', './views'); to say that our views are in the folder ./views

app.set('view engine', 'ejs'); we use ejs

app.get('/', function(req, res) {

res.render('index', {header: 'index!'});

});

*// this is passing an instance variable, that I can call in my view using <%- header %>*

## Twitter

Store the tokens as env variables in ~/.zshrc so we can access them as env[token] and no one can see them

# Templates

In the header, Add link tag with the link to mustache (google mustache js cdn)

<script type="text/javascript" src='https://cdnjs.cloudflare.com/ajax/libs/mustache.js/2.1.1/mustache.js'></script>

**Then below the body, create the template – KEY TO GIVE IT AN ID**

<script type="x-tmpl-mustache" id='tweet-template'>

<div class="row">

<div class="col-md-6 col-md-offset-3 tweet">

<img src="{{ user\_profile\_image }}" alt="" class='avatar pull-left'>

<div class="names">

<span class="full-name">{{ name }}</span>

<span class="username">@{{ screen\_name }}</span>

</div>

<div class="contents"><span class="text">{{ text }}</span></div>

</div>

</div>

</script>

**Finally, render that on the page**

<script type="text/javascript" src='/socket.io/socket.io.js'></script>

<script type="text/javascript">

var socket = io();

socket.on('statuses', function(status) {

// grab the template html we built just above

var template = $('#tweet-template').html();

Mustache.parse(template); // speeds up by caching when receiving loads of requests

// populate the template with the object from our request

var rendered = Mustache.render(template, status)

})

</script>

socket.on('connect', function() {

console.log('Connected!');

});

# Jasmine Testing

<http://jasmine.github.io/2.3/introduction.html>

ToBe true, null

ToEqual

ToMatch (regex)