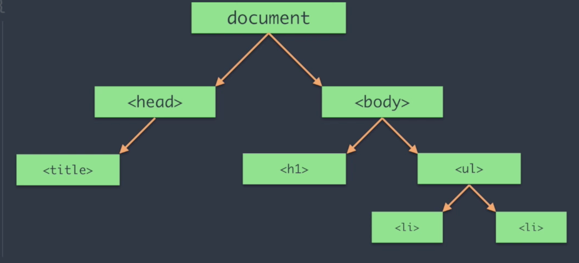
JS Tutorials

Section 1

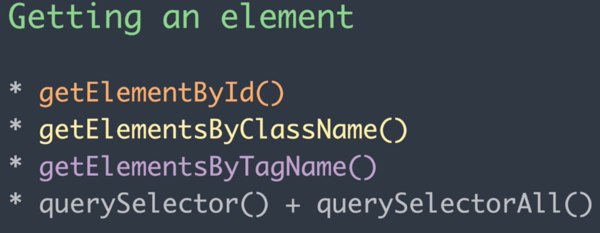
DOM:

The bridge btw the HTML and JS



Section 2

JS Common Queries



// Getting an Element()

document.getElementById('head') // displays all id with head

document.getElementsByClassName('para')//displays all class

document.getElementsByTagName('p')//displays all Tag

document.querySelector('.para')//displays the first one

document.querySelectorAll('.para')//displays all

document.querySelectorAll('.para')[0].innerHTML= 'Changed it' //changing the text

//Creating Things in the Documents

document.createElement()// to create an element

var p = document.createElement('p')

p.innerText = "This is me"

document.body.appendChild(p)

document.createAttribute() //to create attributes like id and class, styles etc

var att = document.createAttribute('id')

att.value = "created"

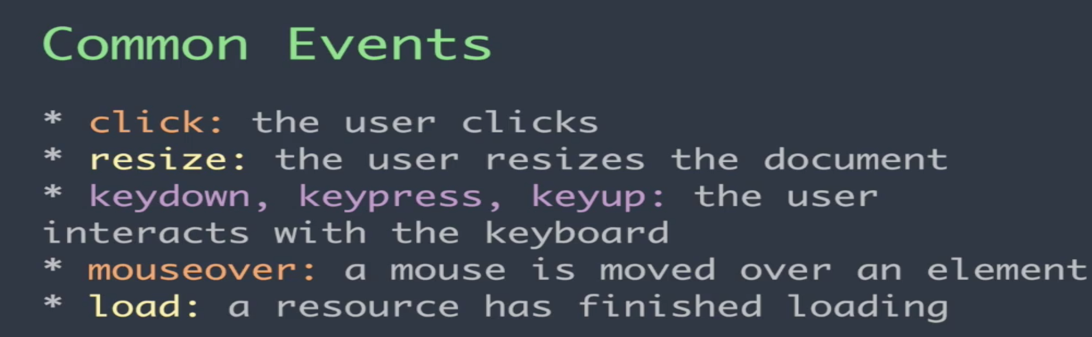
p.setAttributeNode(att) //sets the new p tag to have an id called created

//<p id="created">This is it</p>

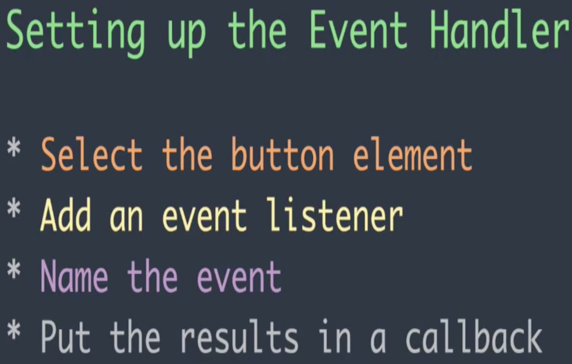
Section 3

Events and Call-backs

Events: are occurrences that happens in a browser



Callbacks: is an event handler. It’s a function that runs when an event occurs

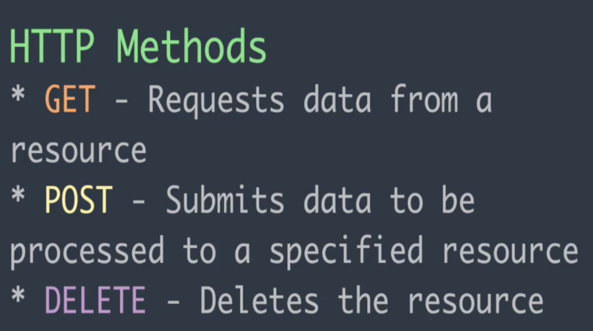
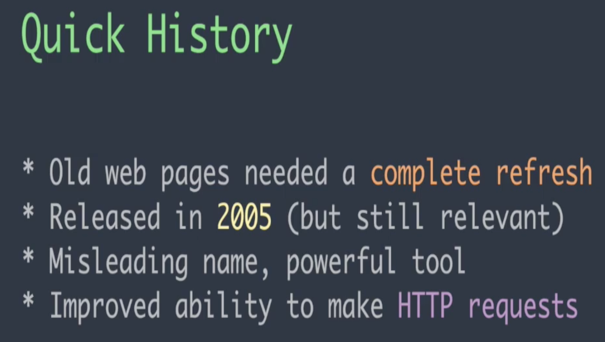
load is the event, function is the call-back. So the function runs when the event occurs

got to folder 3 for examples

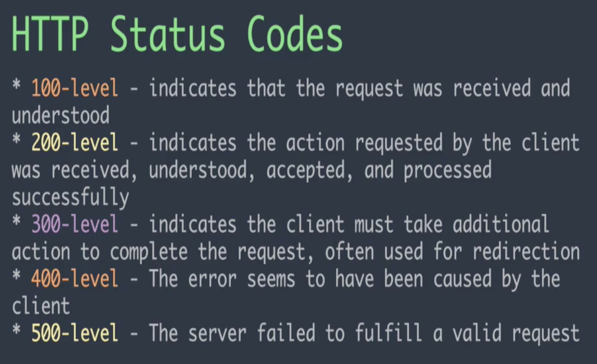
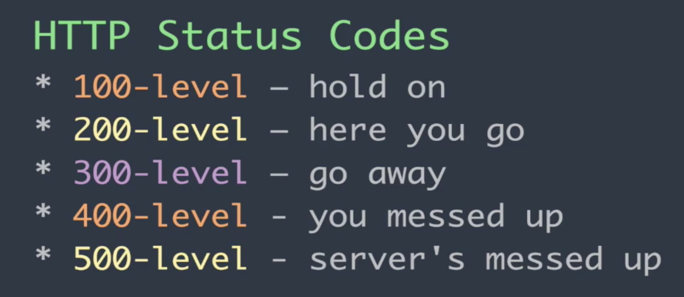
Section 4

AJAX & HTTP Requests

AJAX makes asynchronous request: meaning other things can still be done while waiting for the response to our requests



GET: Pulls, Post: creates Delete: Remove

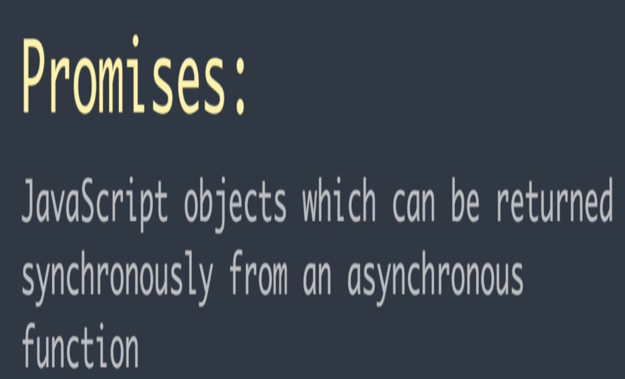
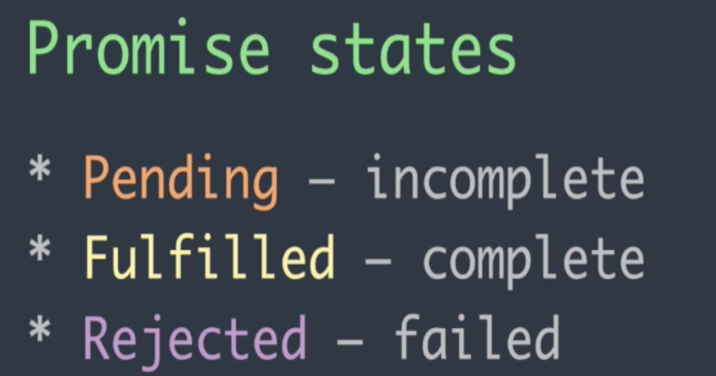
 

PROMISES:

<https://developers.google.com/web/fundamentals/primers/promises>

<https://dev.to/damcosset/i-promise-i-wont-callback-anymore-cp3>

<https://developers.google.com/web/updates/2015/03/introduction-to-fetch>

**FETCH()**

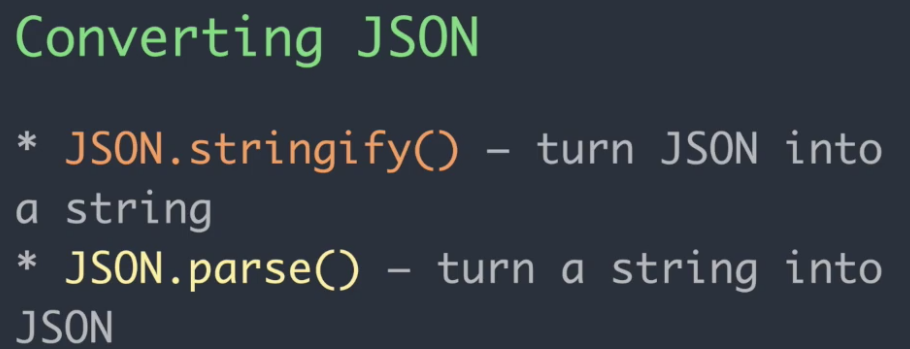
This is a better alternative for events and call backs.

Fetch uses promises to get the desired output.

Check folder “Ajax(4)” to see the example

**SECTION 5**

**JSON**



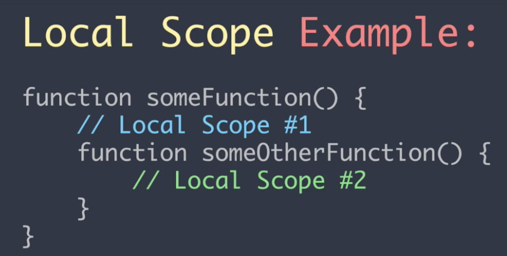
check folder 5 json

SECTION 6

SCOPES AND THE VARIABLE THIS

TWO TYPES OF SCOPE

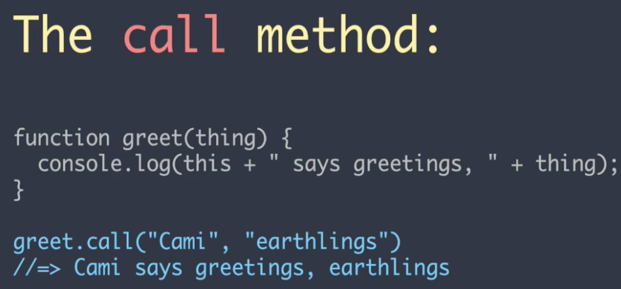
1. LOCAL SCOPE: IN A FUNCTION
2. GLOBAL SCOPE: NOT IN A FUNCTION

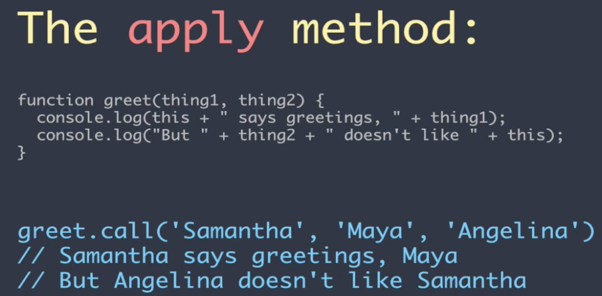
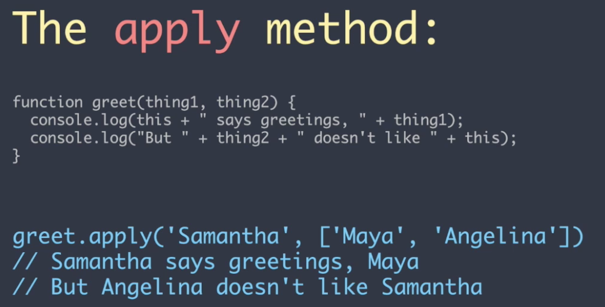


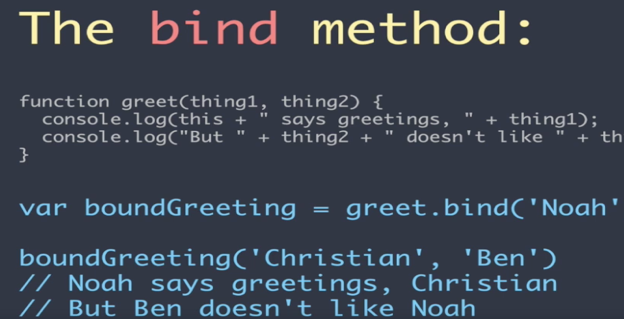
THIS VARIABLE

THIS VARIABLE HAS 3 METHODS

1. CALL: USED TO CALL USING THIS AS THE FIRST VARIABLE
2. APPLY: FOR MULTIPLE USING ARRAY, THIS VARIABLE COMES FIRST AND NOT IN THE ARRAY
3. BIND: ASSIGN THIS TO A VARIABLE



<https://medium.com/@omergoldberg/javascript-call-apply-and-bind-e5c27301f7bb>

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

<https://codeplanet.io/javascript-apply-vs-call-vs-bind/>

SECTION 7

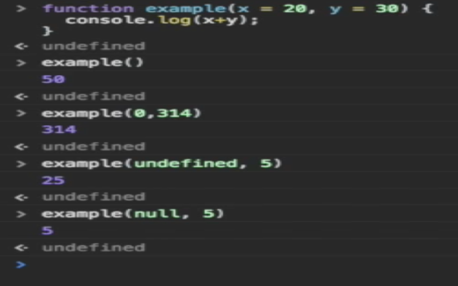
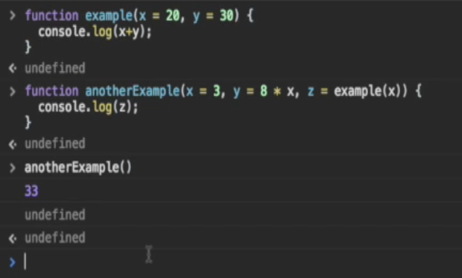
VAR, LET AND CONST

<https://hackernoon.com/js-var-let-or-const-67e51dbb716f>

SECTION 8

DEFAULT VALUES

Basic default values it can also take expressions and functions

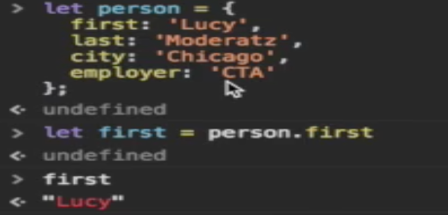
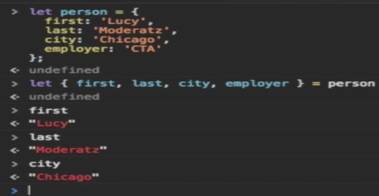
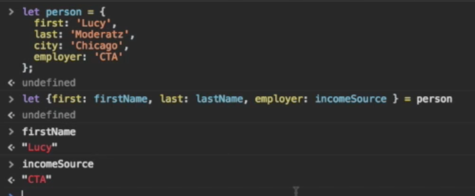
 

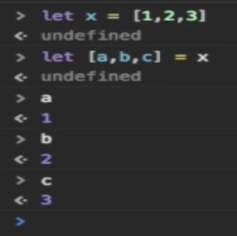
SECTION 9

DESTRUCTURING

Is an expression that allows us to pull data from objects and arrays into their own variable

Different ways of destructuring

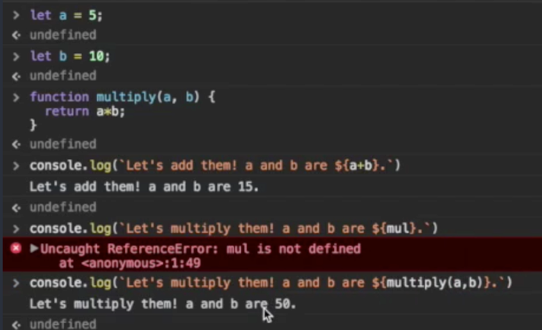
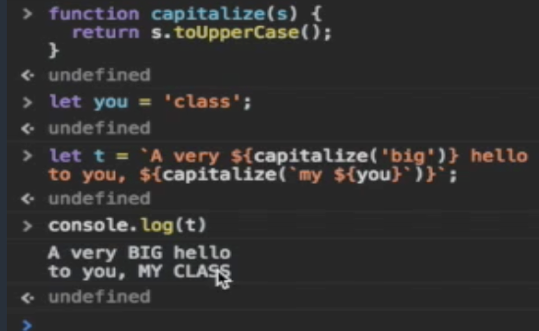


SECTION 10

LITERALS

TWO TYPES OF LITERALS

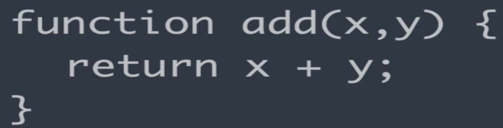
* OBJECT LITERALS
* TEMPLATE LITERALS

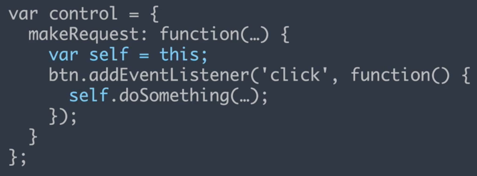
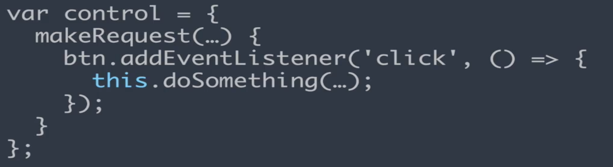
 

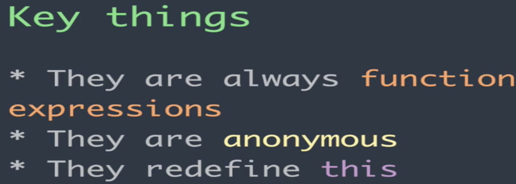
SECTION 10

ARROW FUNCTION

BEFORE NOW(ES6)



INSTALL DEPENDENCIES

Package managers

install nodejs, it automatically installs npm

Brew install yarn

**HOW TO USE NPM**

Create new project

Mkdir

Npm init

**Dependencies:**

Npm install vaca

**HOW TO USE YARN**

Create new project

Mkdir

yarn init

**Dependencies:**

yarn add dogefy

Npm install -g grunt-cli

Npm install -g gulp-cli

Npm install –g webpack