Intro To Javascript

Intro to Web Development

Redbrick

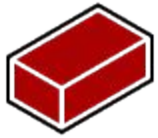
Preamble

- I don't like JavaScript
- But with JS your feelings don't matter.
- Browsers don't work well with any other language so you have to write code that either:
  - Uses JavaScript as a compile target
  - Is JavaScript
- Js Suffers from there's an App for that Syndrome.
  - **208,067** total packages on www.npmjs.com
- So If you're planning on writing Plain old JS have fun getting Help from StackOverflow.
- The following examples have been stolen shamelessly from
  - Gary Bernhardt Wat (CodeMash 2012)

Redbrick

On the Brightside

It’s not PHP.



Redbrick

JavaScript != Java

JavaScript and Java are not even remotely related languages.
While they share some structural similarities this has more to do Java and Javascript's syntax being inspired by C code.

- JavaScript is an **interpreted** Language Whereas Java is **compiled**.
- Java uses **block-based scoping**, JavaScript uses **function-based scoping**
- JS **Functions are variadic** (You can throw too many or too few variables at a function and it wont cry) Java gives you a big nice compiler error.

Java and **Javascript** are similar like **Car** and **Carpet** are similar.

http://stackoverflow.com/questions/245062/whats-the-difference-between-javascript-and-java

Redbrick

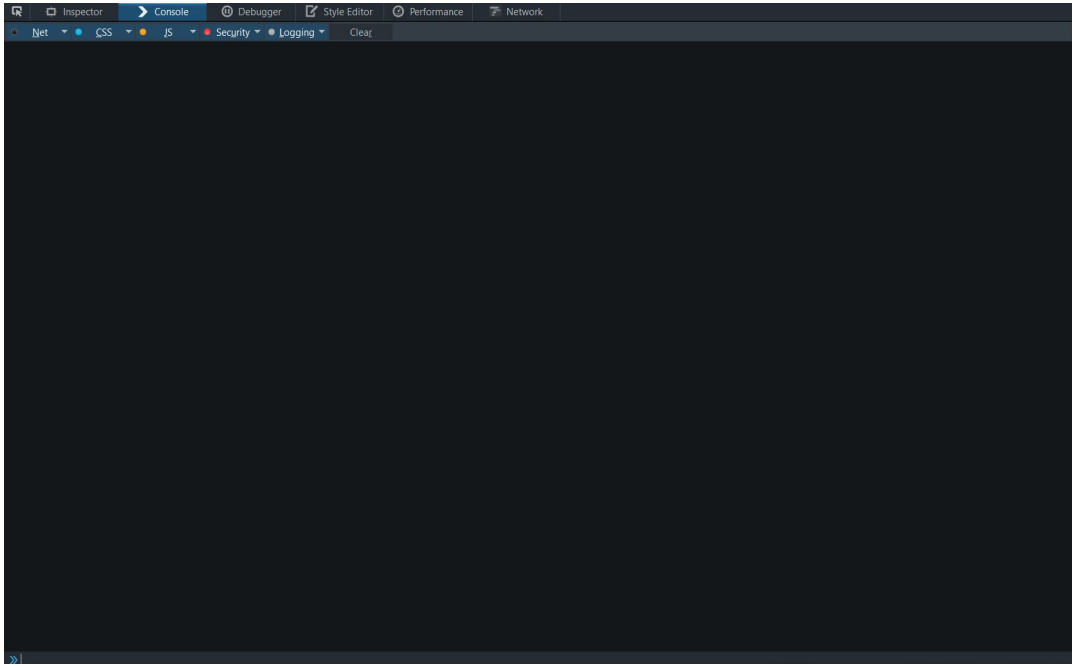
The Browser Console

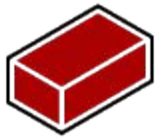
Your browser comes with a developer console.

To Open it press CTRL+Shift+K

In here you can run JS code.

Because Space has a salt crystal for a heart. He's only added info to open a console in Mozilla Firefox





Redbrick

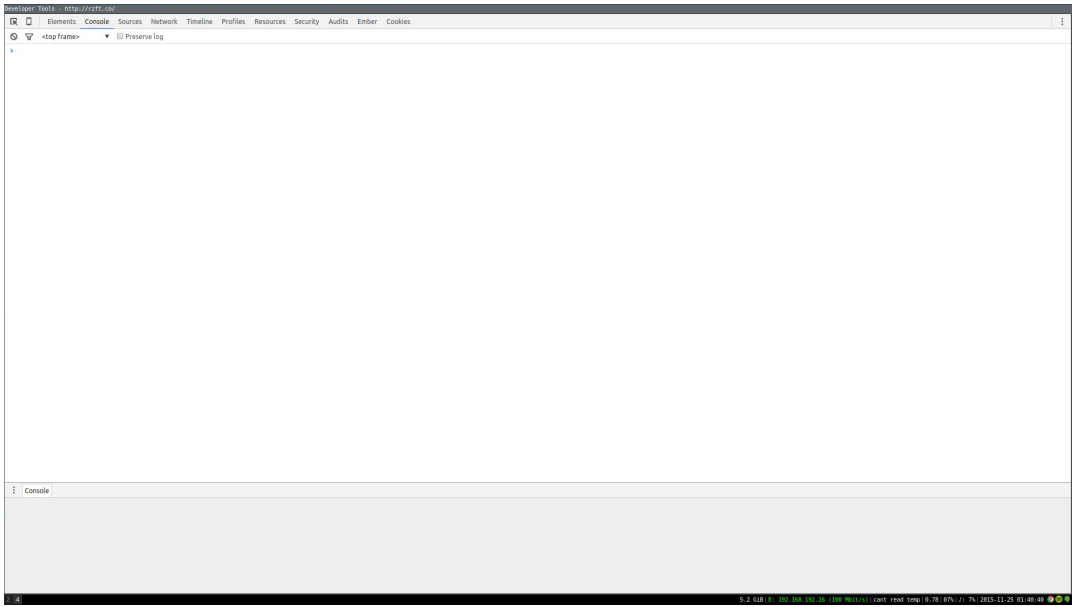
The (Chrome) Browser Console

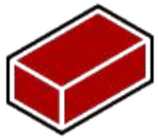
... But voy had mercy on your poor soul

To Open it press F12

In here you can run JS code just as in Firefox (but better)

(It's the same)





Redbrick

Hello World

To run a hello world type into the console

console.log(“Hello World”);

Javascript uses semicolons to end statements. Much like Java or C.

Redbrick

Success! (hopefully)

```
» console.log("Hello World");
← undefined
  Hello World
```

Redbrick

Variables

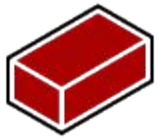
JS is a Dynamically Typed Language (Like Python)

- This means the type assigned to a Variable is able to change.
- However this is not recommended for performance reasons

JS Supports the Following types

- Integers
- Booleans
- Strings
- Arrays
- Objects*

*Everything is an object in JS



Redbrick

Declaring some Variables

```
var length = 16;                                    // Number

var lastName = "Johnson";                           // String

var cars = ["Saab", "Volvo", "BMW"];                // Array

var x = {firstName:"John", lastName:"Doe"};         // Object
```

Redbrick

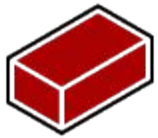
JavaScript Functions

Because JS is loose about its type. **Functions can return any value**, For languages like java you would have to supply the type of the return

```
function myFunction(a, b) {
        return a * b;
}
```

Functions in JS can be recursive

```
function factorial (number) {
  if (number <= 0) { // terminal case
    return 1;
  } else { // block to execute
    return (number * factorial(number - 1));
  }
}
```



Redbrick

JavaScript Objects

For the first year's Objects sound scary. Don't worry though they are super simple to understand

```
{
        talk        : “Intro to Javascript”,
        speaker : “Sean Healy”
}
```

Objects are organized into keys. In this object we have two keys, talk and speaker. Keys can contain any type of variable including other objects.

Redbrick

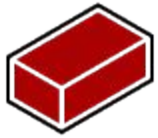
Try it in your console.

var rb= {title:"Intro to Javascript", speaker:"Sean Healy"};

to get the data out of an object you simply call like this

objectName.*keyValue*

Try to combine it with your previous knowledge and get the console to print out this talks title :)



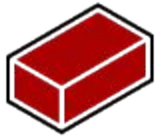
Redbrick

JSON - JavaScript Object Notation

Some smart people realized that JavaScript object-like structure would be a great way to transfer files across the internet

Advantages:

- Lighter than XML
- Easier to read
- Most web based clients are JS based so reading a JS object is easy for them.



Redbrick

So what does json look like?

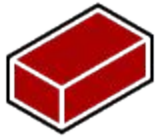
Exactly like a JavaScript Object


```
{
   name: "Sean Healy",
   about: "Sometime in the near future Mayo will win Sam. If you care about me please File a missing persons report when this happens :)
   somewhere between Dublin and Mayo",
   dob: "19/04/1994",
   sex: "Male",
 ▾ qualifications: [
     ▾ {
           title: "Webmaster",
           group: "Redbrick",
           start: "April 2015",
           end: "present"
       },
     ▾ {
           title: "Events Officer",
           group: "Redbrick",
           start: "April 2014",
           end: "April 2015"
       },
     ▾ {
           title: "Events Officer",
           group: "Redbrick",
           start: "April 2014",
           end: "April 2015"
       }
   ]
}
```


I'm using a pretty cool browser addon to make Json reader pretty

https://addons.mozilla.org/en-us/firefox/addon/jsonview/

#FireFoxMasterRace #ChromeSux



Redbrick

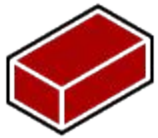
Flow Control

If For and while exist in Javascript and they work similarly to **Python**. But they are **syntactically similar to C** based languages in their structure

```
if (booleanValue){
        // Do something
} else {
        //Do Something else
}
```

```
while (booleanValue){
        //Do Something
}
```

In JS the braces are used to Show what is contained inside a block. **Python uses indentation** to do same thing. Standard JS Style still means you should still indent though



Redbrick

Async

It’s ok to start Drinking now.

Redbrick

Async

One of the major advantages of JS is its support for Asynchronous programming.

This gives us the concept of doing multiple things at the same time.

Languages like Java and Python are naturally synchronous consider this python script.

```
print “Hello”
time.sleep(10000000000000000000000000000000)
print “World”
```

Async

The Expected output in this case is “**Hello**” wait a boat load of time. “**World**”

In any user interface based code **this is unacceptable**. As the Client does the Long Command the client is **essentially unresponsive**, Nothing else can be done until this long command is finished.

- Async allows us to **run multiple things in parallel**.
  - This makes us far **more flexible**. As long running commands and functions can run in their own time without the user realizing.

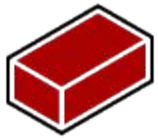
Async example

```
window.setTimeout(function() {
    console.log("World");
}, 100000);
console.log("Hello");
```

In a sync world this function would return “World Hello”

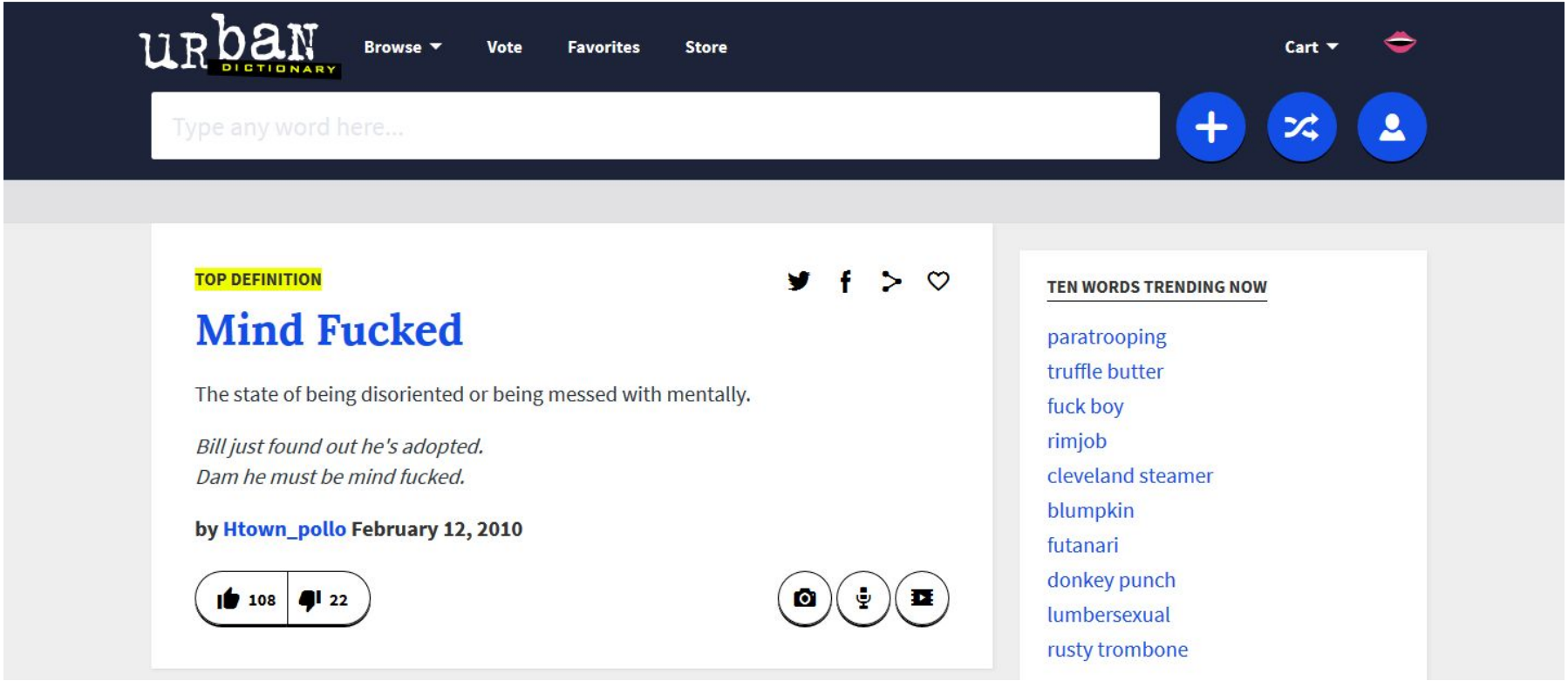
To write Async code we use what are called **Callback functions**. This is basically a function that does nothing until it has what it needs to succeed.

The code in this case realizes that it cannot fulfill the setTimeout() call right now, so it skips it until that code is able to be used (after the timeout has passed.)



Redbrick

This is an appropriate Reaction right now



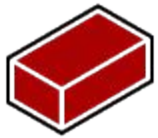
Redbrick

Dangers of Async

- Callback Hell

```
function hell (win) {
  // for listener purpose
  return function () {
    loadLink(win, REMOTE_SRC+'/assets/css/style.css', function () {
      loadScript(win, REMOTE_SRC+'/lib/async.js', function () {
        loadScript(win, REMOTE_SRC+'/lib/easyXDM.js', function () {
          loadScript(win, REMOTE_SRC+'/lib/json2.js', function () {
            loadScript(win, REMOTE_SRC+'/lib/underscore.min.js', function () {
              loadScript(win, REMOTE_SRC+'/lib/backbone.min.js', function () {
                loadScript(win, REMOTE_SRC+'/dev/base_dev.js', function () {
                  loadScript(win, REMOTE_SRC+'/assets/js/deps.js', function () {
                    loadScript(win, REMOTE_SRC+'/src/'+win.loader_path+'/loader.js', function () {
                      async.eachSeries(SCRIPTS, function (src, callback) {
                        loadScript(win, BASE_URL+src, callback);
                      });
                    });
                  });
                });
              });
            });
          });
        });
      });
    });
  };
}
```

- Good luck with closing them...



Redbrick

Dangers

Some async functions return values occasionally as they progress through their task. (Think stuff like a stream buffer).

If you dont expect the occasional returns you can end up getting junk output that doesnt make any sense in your head.

Dangers: Example

```
var data = {}

data = waitForServerToRespond()


console.log(data)
```

WHAT WILL HAPPEN?

Dangers: Example

ANSWER: {}
Better solution would be

```
var data = {}
waitForServerToRespond(function(newData){
    data = newData;
    console.log(data)
});
```

```
function waitForServerToRespond(callBack){
    // do stuff
    callBack(returnData)
}
```

XMLHttpRequest

The Main reason you will use async is for **XMLHttpRequests**.

That names a bit of a mouthful so let's call them **AJAX requests**.

AJAX = **Asynchronous JavaScript and XML**

This is a bit of a lie because we won't be using XML as our interchange format. Mainly because JSON is a significantly better than XML (Just take my word for this, or ask ANYBODY in the industry... apart from IBM*)

*Look up JSONx - rzft.co/JcCv6

Redbrick

Request Types

GET - I want this Data you have at $URL.

POST - I have Data I want to give to $URL.

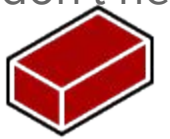
PUT - Remember that Data I had given you? Well I want to change it.

DELETE - I want to remove this Data at $URL

Redbrick

Sample Get Request

```
function getRequest() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() { // async callback
        if (xhttp.readyState == 4 && xhttp.status == 200) {
              console.log(xhttp.responseText);
        }
  };
  xhttp.open("GET", "http://www.redbrick.dcu.ie/~space/test.php", true);
  xhttp.send();
}
// We will only be covering GET requests in this tutorial. We don't need the others.
```



Redbrick

How does that work?

xhttp.open("GET", "http://www.redbrick.dcu.ie/~space/test.php", true);
Is a long running process. This will take somewhere in the **region of 10-400ms** to complete **in optimal circumstances**. Not a lot of time for people but **tonnes of time to a computer.**

xhttp.onreadystatechange = function() { } // async callback
This listens for our open command to terminate. Once open has terminated our callback function is invoked. And its contents are called.

Redbrick

How does that work? (Cont)

if (xhttp.readyState == 4 && xhttp.status == 200) {}

xhttp.readyState has **multiple states**.

- 0: request not initialized
- 1: server connection established
- 2: request received
- 3: processing request
- 4: request finished and response is ready

We wait for 4 because that is when we are done.

xhttp.status

Http Requests come with a response Code. **200** indicates a **successful request** the one's you are probably familiar with is either **404 (Data not Found)** or **500 (Internal Server Error)**

Redbrick

The DOM

DOM? Whose DOM?

Dom or Document Object Model is hard to define.

> The short answer is the data that you end up seeing in the browser is the DOM
>
> Javascript is really good at modifying the DOM.

It's not,

- Your HTML file(though they are similar)

Redbrick

DOM manipulation

90% of every JS app you write will be some sort of app that

- Receives and Sends Data from a server.
- Modifies the DOM based on the response.

Js gives us access to the document object so we can work with it.

Redbrick

document Functions

Finding HTML Elements

- document.getElementById(*id*)
- document.getElementsByClassName(*name*)
- document.getElementsByTagName(*name*)
- document.querySelectorAll(CSSselector);

Modifying HTML Elements

- document.createElement(*element*)
- document.removeChild(*element*)
- *element*.innerHTML = *new html content*
- *element*.setAttribute*(attribute, value)*

Redbrick

Getting stuff

1. `var myElement = document.getElementById("title");`
   a. get the data contained inside the tag with the id of title.
2. `var x = document.getElementsByTagName("p");`
   a. Returns an array with the data contained in all the p tags in a html file.
3. `var x = document.getElementsByClassName("table_row");`
   a. See 2.A(Except it works with classes instead of tags)

Redbrick

Changing stuff in element.

1. document.getElementById("ID").innerHTML = "This has been changed.";
   a. Changes the contents of the html element with an ID of ID to “This has been changed.”
2. document.getElementById("image").src = "landscape.jpg";
   a. In this case imagine an image, We are changing the src attribute to another image.

Redbrick

Do you want to Build a Js CV?

You can have a look at a JSON file I've provided on

www.redbrick.dcu.ie/~space/test.php

I've written a shitty HTML file for you to get started

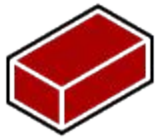
http://www.redbrick.dcu.ie/~space/HTMLSample.txt

JS has a security feature called **CORS** so if you want to write your own json file and have javascript read it you can use this, I hate it you will too :)

http://www.redbrick.dcu.ie/~space/JsonScript.txt

If you are having problems with your own json file lint it

http://jsonlint.com/



Redbrick

If you want to be fancy.

Create some extra HTML elements in JS.

JavaScript not your thing?

LEAVE.

or stay...

Your Alternatives:

- Dart (dartlang.org) - We will be using it for new RB website
- CoffeeScript (coffeescript.org)
- TypeScript (typescriptlang.org)

All of the above compile to JS so it can be ran in any browser