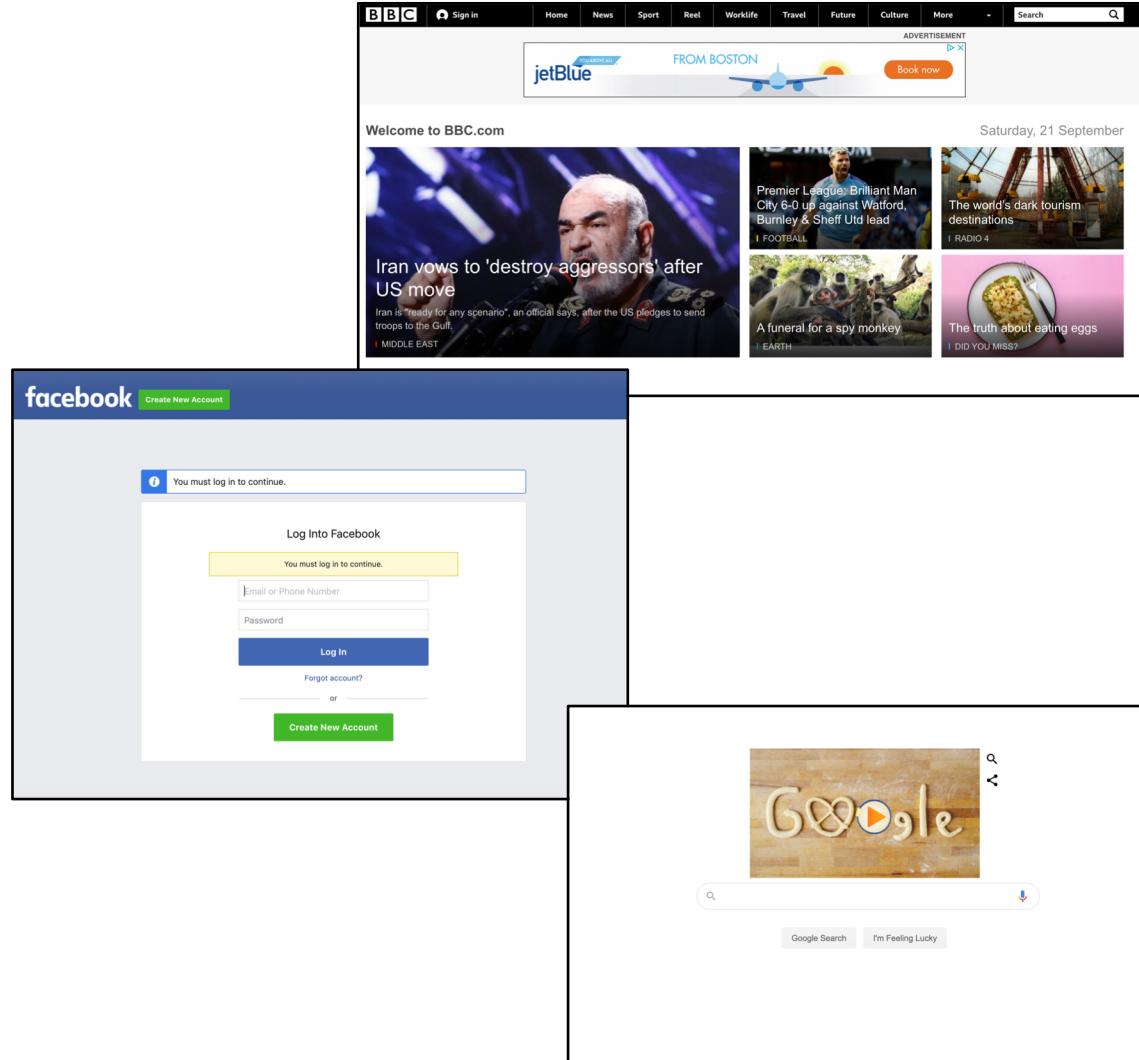


# 1.125 – Software Engineering and Architecting

Lectures 5-6  
NodeJS and Backend Development

# The Web – Our point of view

Front End  
Websites/App interfaces



# A Modern Web-App

Front End  
Websites/App interfaces

HTML - Structure  
CSS - Style  
JavaScript - Control



Back End  
Server/Databases

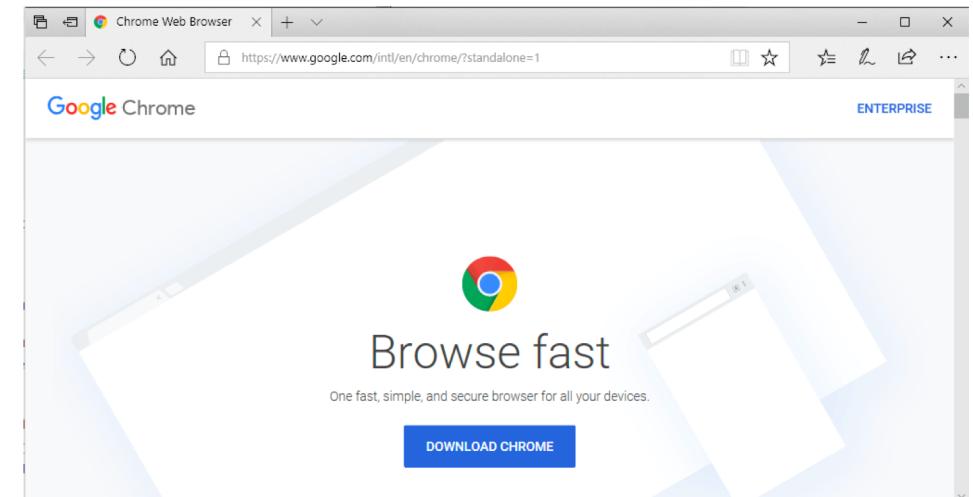
NodeJS - Control  
MongoDB - Storage  
The Cloud - Portability



# One Language to rule them all

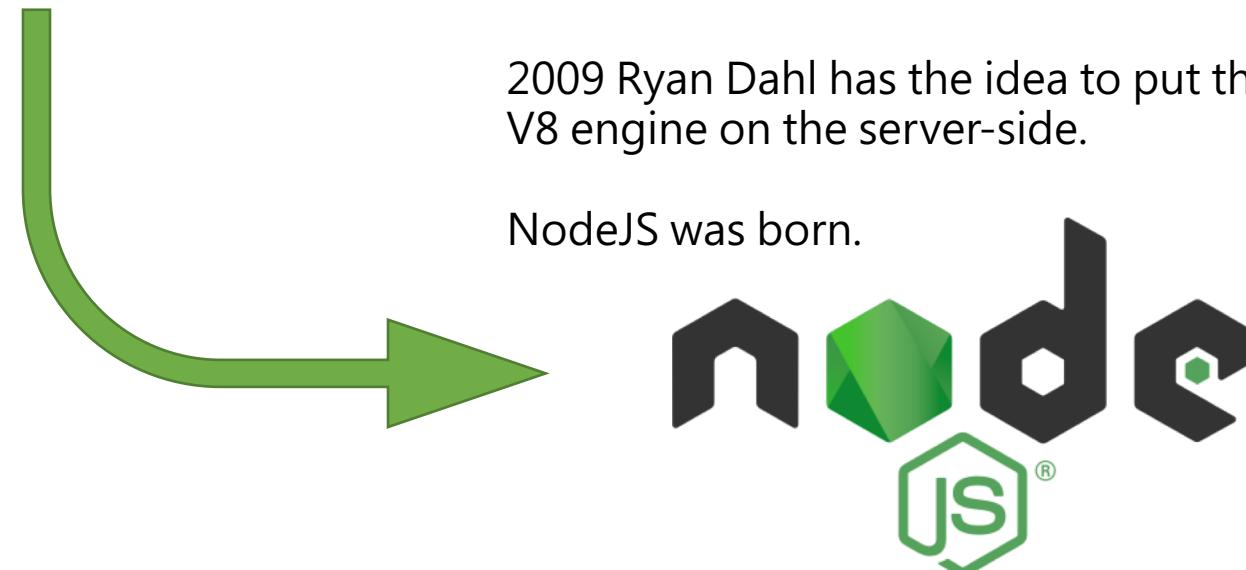
2008 Google creates a new browser from the ground up.

The JavaScript engine, V8 is extremely fast at running JavaScript



2009 Ryan Dahl has the idea to put the V8 engine on the server-side.

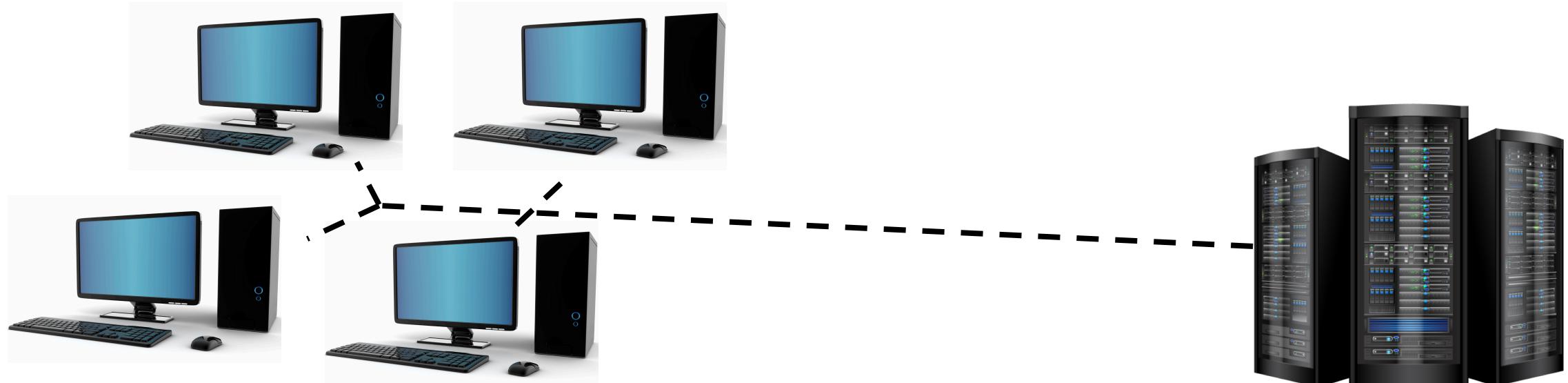
NodeJS was born.



# One Language to rule them all

Why is it a big deal?

1 Language. Front end. Back end. Full Stack.



# NodeJS Install

<https://nodejs.org/en/>

# A soon-to-be-familiar sight

```
callbacks.js — Teaching_1125_Fall19
js callbacks.js ×

exercises > javascript_basics > js callbacks.js > ...
1
2 // Callback functions are a crucial part of asynchronous programming.
3 // These functions essentially wait for some other function to complete before they run.
4
5 // A simple example
6 function myfunc(variable){ // regular function, no callback
7   console.log(variable);
8 }
9 myfunc('hello');

10
11
12 function myfunc_w_callback(variable,callback_func){
13   callback_func(variable); // this will apply the func that is passed as a parameter
14 }
15 function upperandlog(variable){ // we can define the function separately
16   var upperVariable = variable.toUpperCase();
17   console.log(upperVariable);
18 }
19 myfunc_w_callback('hello',upperandlog);

20
21
22 myfunc_w_callback('hello',function(variable){ // we can also define the function directly inside the callback
23   var upperVariable = variable.toUpperCase();
24   console.log(upperVariable);
25 });
26
27 // Arrays contain special formats for these callbacks
28 var array = [-5,-4,-3,-2,-1,0,1,2,3,4,5,6];
29 // forEach
30 var newarray = [];
31 array.forEach(function(item){ // essentially forEach is just a shorter for loop
32   newarray.push(item *=-1);
33 });
34 console.log(array);
35 console.log(newarray);
36
37 // map
38 var newarray2 = array.map(function(item){ // map returns a new array with the mapped values
39   return item *=-2;
40 });
(base) ➜ javascript_basics node callbacks.js
hello
HELLO
HELLO
[ -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6 ]
[ 5, 4, 3, 2, 1, -0, -1, -2, -3, -4, -5, -6 ]
[ -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6 ]
[ 10, 8, 6, 4, 2, -0, -2, -4, -6, -8, -10, -12 ]
[ -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6 ]
[ 1, 2, 3, 4, 5, 6 ]
[ -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6 ]
6
```

Editor + Console

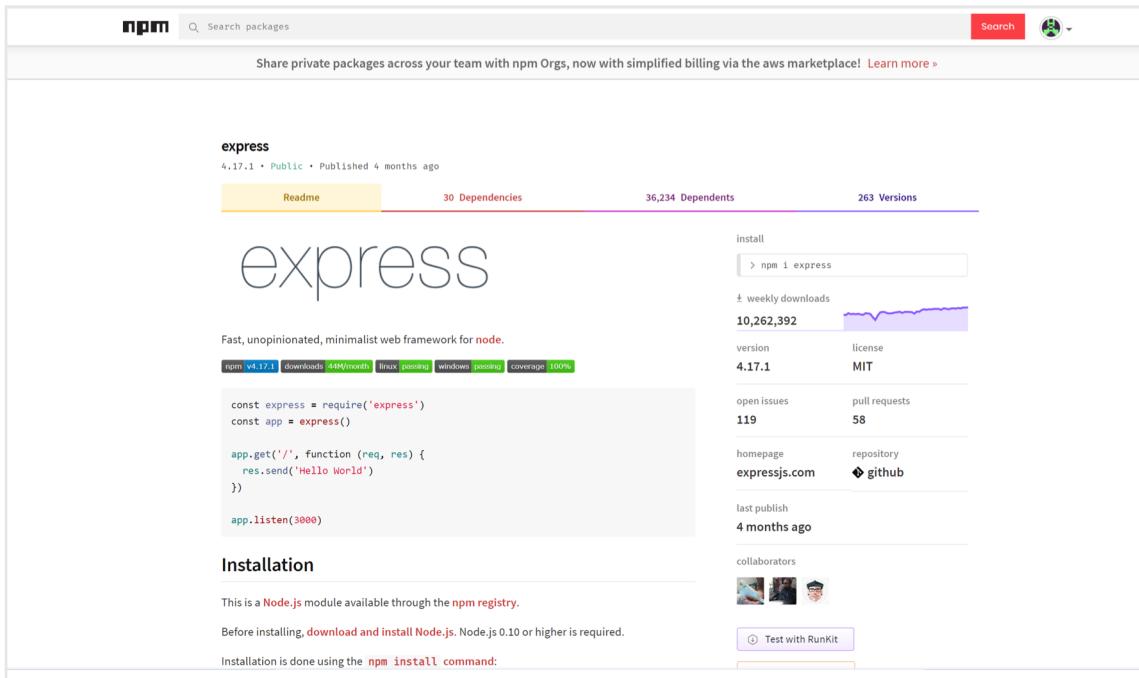
# In Class Activity

1. Create a new repo on your GitHub called: *nodeExamples*
2. Clone this empty repo to your computer
3. Create a new file with the filename: *nodeExamples1.js*
4. Solve the following problems:
  1. Hash-Pyramid: create a function that will take as an input the height of a half pyramid and print out to the node console a pyramid of hashtag signs.  
e.g. `pyramid(4)` would yield the following output.  
`#`  
`##`  
`###`  
`####`
5. After you have finished, update your github repo

# The Node Package Manager

## Downloads Last Week

13 BILLION... can you think of anything else that happened 13 Billion times in the last week?



What a typical homepage for a library looks like

# package.json

Lists the packages your project depends on

Specifies versions of a package that your project can use using [semantic versioning rules](#)

Makes your build reproducible, and therefore easier to share with other developers

```
1  {
2    "name": "expressdockerimage",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "start": "node server.js"
9    },
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "express": "^4.16.4"
14   }
15 }
16 }
```

Inside a package.json file

# require()

How do we use these packages?

They are available as objects with internal functions (like the console.log() ) after we add the 'require()' function

```
1  var express = require('express');
2
3  var app = express();
4
5
6  app.get('/',function(request,response){
7    |   response.sendFile(path.join(__dirname + '/index.html'));
8  });
9
```

# Your First Web Server

Create a new repo on GitHub – Use the Node git ignore

Use npm init to create a new node package

Install express and path with npm

Create the following files:

index.html

frontend.js

server.js

Upload the files to GitHub.