

# Intro to Web Development



Nina Zakharenko  
Brian Holt



[bit.ly/1jXij63](https://bit.ly/1jXij63)



**Tools**

# **A good toolkit is important.**

Why?

The tools of the trade are designed to help us solve problems faster.

# The Tools you Need

- A good text editor
- A way to debug your HTML, CSS, and Javascript

# Our Toolbox



# Sublime: Syntax Highlighting

Colors help differentiate between different parts of your code.

```
<html></html>
```

```

```

# Sublime: Match Parenthesis

```
alert( 'Welcome!' ) )
```



```
alert( 'Welcome!' ) )
```

# Sublime: Replace All

I have 5 apples.  
My favorite fruit is apples.

Find What:	apples	Find	Replace
Replace With:	oranges	Find All	Replace All

I have 5 oranges.  
My favorite fruit is oranges.

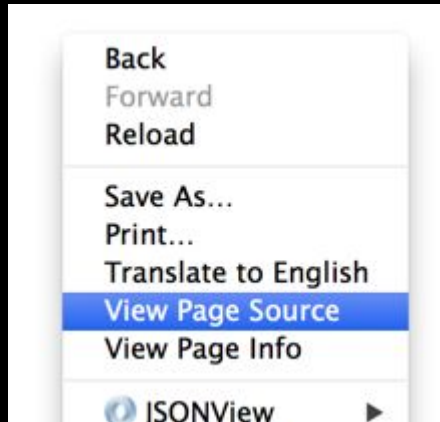


# Debugging a web page

# Chrome: View Source

Visit [www.google.com](http://www.google.com)

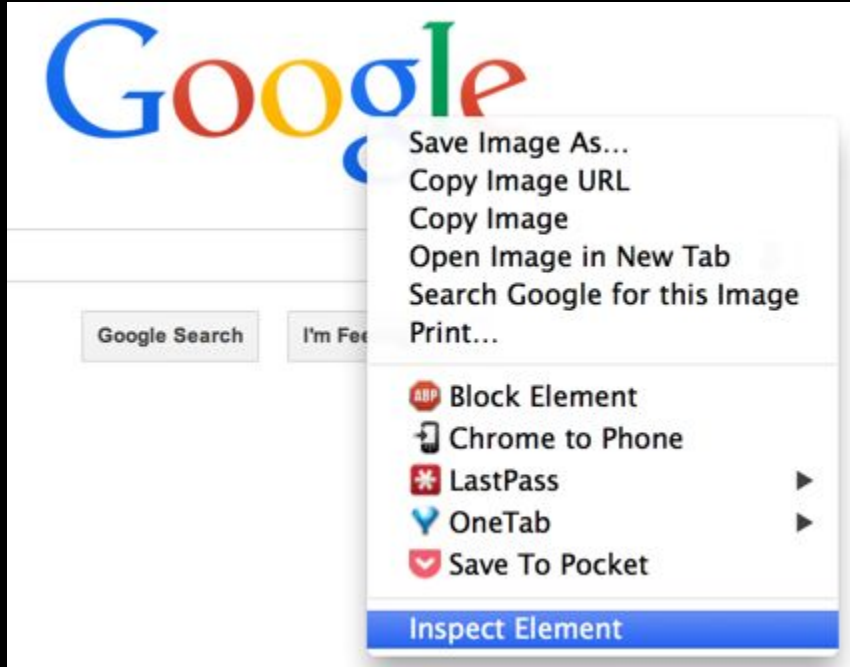
Right Click -> View Page Source



# Whoa.

```
3,10200134,10200160,10200248,10200318,10200330,10200334,10200353,10200387,10200391,10200393,10200396,10200398,10200408,10200440,10200477',
{e:'4791,17259,3300053,3300101,3300134,3300137,3300164,3310648,3310699,3312180,3700209,4000116,4007661,4008142,4009033,4009641,4010806,401
12373,4012504,4013414,4013591,4013723,4013787,4013823,4013967,4013979,4014016,4014093,4014431,4014515,4014636,4014671,4014805,4014991,4015
5633,4015772,4016127,4016279,4016283,4016309,4016367,4016373,4016487,4016824,4016976,4017162,4017204,4017280,4017284,4017544,4017578,40175
681,4017694,4017710,4017742,4017818,4017894,4017913,4017980,4017982,4018009,4018019,4018106,4018126,4018159,4018181,4018363,4018416,40184
21,4018638,4018874,4018914,4018923,4018933,4019014,4019074,4019084,4019142,4019191,4019205,4019207,4019225,4019268,4019337,4019339,401938
8,4019494,4019564,4019590,4019661,4019664,4019740,4019779,4019789,4019800,4019801,4019816,4019827,4019843,4019856,4019888,4019891,4020014,
8300012,8300021,8300027,8300033,8300039,8300042,8300054,8300057,8300066,8500223,8500256,8500272,8500393,8500433,8500509,8500516,8500553,
0318,10200330,10200334,10200353,10200387,10200391,10200393,10200396,10200398,10200408,10200440,10200477',ei:'Uk7PU6PyNKW9iwKvrcWcGg'},auth
{en:1,bv:21,pm:'p',u:'8048fb21'}};google.kHL='en';})();(function(){google.lc=[];google.li=0;google.getEI=function(a){for(var b;a&&(!a.getA
(b=a.getAttribute("eid"))));)a=a.parentNode;return b|google.kEI};google.https=function(){return"https:"==window.location.protocol};google.
{return(new Date).getTime();};google.log=function(a,b,d,h,k){var c=new Image,f=google.lc,e=google.li,g="",l=google.ls||"";c.onerror=c.onloa
f[e]};f[e]=c;d||-1!=b.search("&ei=")|| (g="&ei="+google.getEI(h));a=d||"/"+(k||"gen_204")+ "?atyp=i&ict="+a+"&cad="+b+g+l+"&zx="+google.time
(google.ml(Error("a")),!l,{src:a,glmm:l}),delete f[e]};(c.src=a,google.li=e+1));google.y={};google.x=function(a,b){google.y[a.id]=[a,b];ret
(google.x({id:a+m++},function(){google.load(a,b,d)});var m=0;})();
google.j.b=!location.hash&&!!location.hash.match('[#&]((q|fp)=|tbs=simg|tbs=sbi)'); (function(){google.sn="webhp";google.timers={};google.
{google.timers[a]={t:{start:google.time()},bfr:!!b};window.performance&&window.performance.now&&
(google.timers[a].wsrt=Math.floor(window.performance.now()));};google.tick=function(a,b,c)
{google.timers[a]||google.startTick(a);google.timers[a].t[b]=c|google.time();google.startTick("load",!0);
try{google.pt=window.chrome&&window.chrome.csi&&Math.floor(window.chrome.csi().pageT);}catch(d){});})();
(function(){'use strict';var g=this,k=Date.now|function(){return new Date};var r=function(c,d){return function(a){a| (a=window.event);ret
navigator&&Macintosh/.test(navigator.userAgent),w="undefined"!=typeof navigator&&!Opera/.test(navigator.userAgent)&&WebKit/.test(naviga
{A:13,BUTTON:0,CHECKBOX:32,COMBOBOX:13,LINK:13,LISTBOX:13,MENU:0,MENUBAR:0,MENUITEM:0,MENUITEMCHECKBOX:0,MENUITEMRADIO:0,OPTION:13,RADIO:3
TABLIST:0,TREE:13,TREEITEM:13},y={CHECKBOX:1,OPTION:1,RADIO:1};var z=function(){this.o=this.i=null},B=function(c,d){var a=A;a.i=c;a.o=d;re
c=this.i;this.i&&this.i!=this.o?this.i=this.i.__owner||this.i.parentNode:this.i=null;return c};var C=function(){this.p=
[];this.i=0;this.o=null;this.s=!1};C.prototype.k=function(){if(this.s)return A.k();if(this.i!=this.p.length){var c=this.p[this.i];this.i+
(this.s=!0,B(c.__owner,this.o));return c}return null};var A=new z,E=new C;var G=function(){this.w=[];this.i=[];this.o=[];this.s={};this.k=
[];F(this,"_custom")},H="undefined"!=typeof navigator&&iPhone|iPod|iPad/.test(navigator.userAgent),I=/\s*/,\K=function(c,d){return fun
if(!a.detail||!a.detail_type)return;b=a.detail_type;if("click"==b&&(u&&a.metaKey||!u&&a.ctrlKey||2==a.which||null==a.which&&4==a.button
f=a.which||a.keyCode||a.key:w&&3==f&&(f!=13);var e=a.target||a.srcElement,m=
```

# Better Way - Inspect Element



# Much Better.

```
window.lol&&lol()">
```

```
lity='hidden':</script>
```



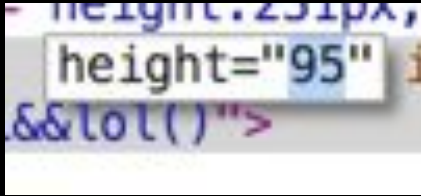
269 × 207 pixels (Natural: 538 × 190 pixels)

```
op:zop.
```

```
" src="/images/srpr/logo11w.png" style="padding-
```

# Chrome: Updating Values

Double click on the field you want to change.



Enter the new value, and press enter to see your changes.

# Update Values

Note: Your updates are only on your local machine.

If you refresh the webpage, your updates will be gone.

# Updating Values: Exercise

Let's stretch out the google logo.

1. Navigate to <http://www.google.com>
2. Inspect Element on the Google Logo
3. Find the width value, set it to 700.
4. Find the height value, set it to 30.

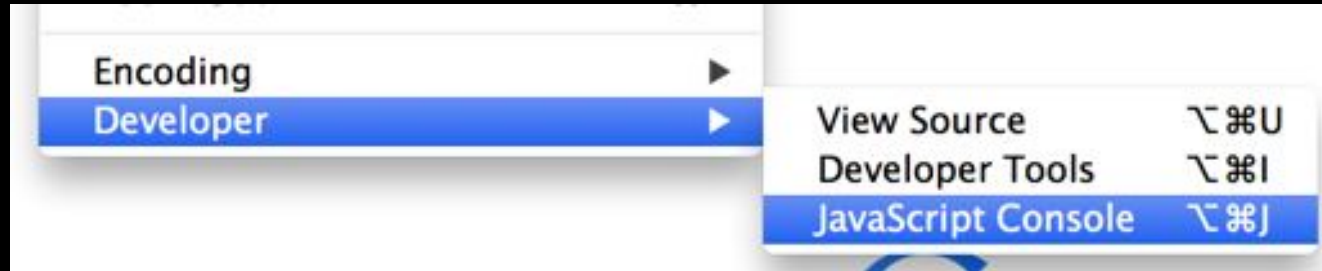


# Updating Values: Results



# Chrome: Javascript Console

View -> Developer -> Javascript Console



# Interactive Javascript

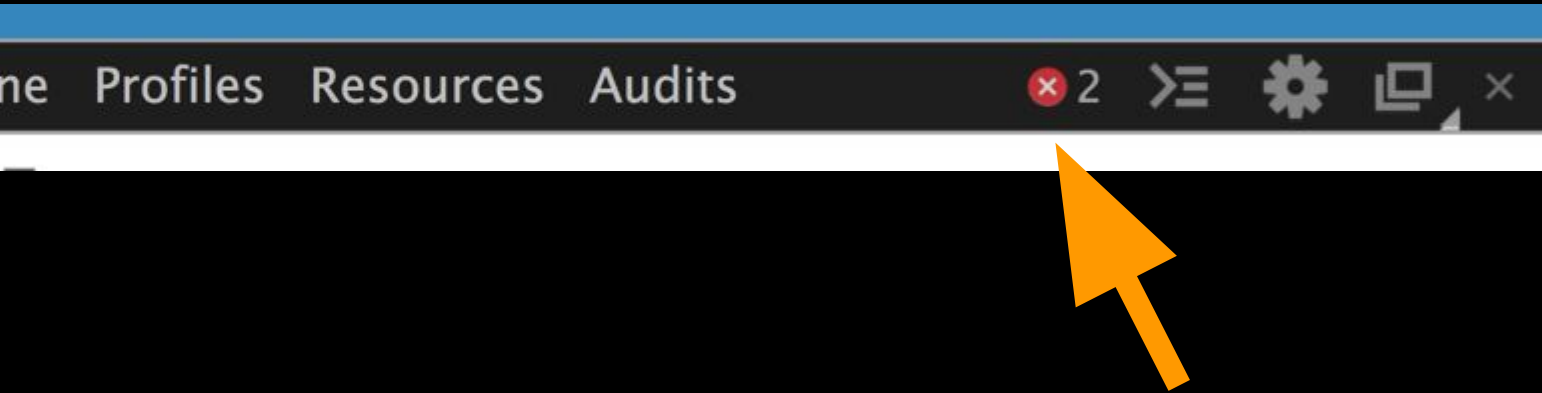
Open the Javascript Console.

```
> console.log("hello world");
```

```
> console.log("hello world");  
hello world
```

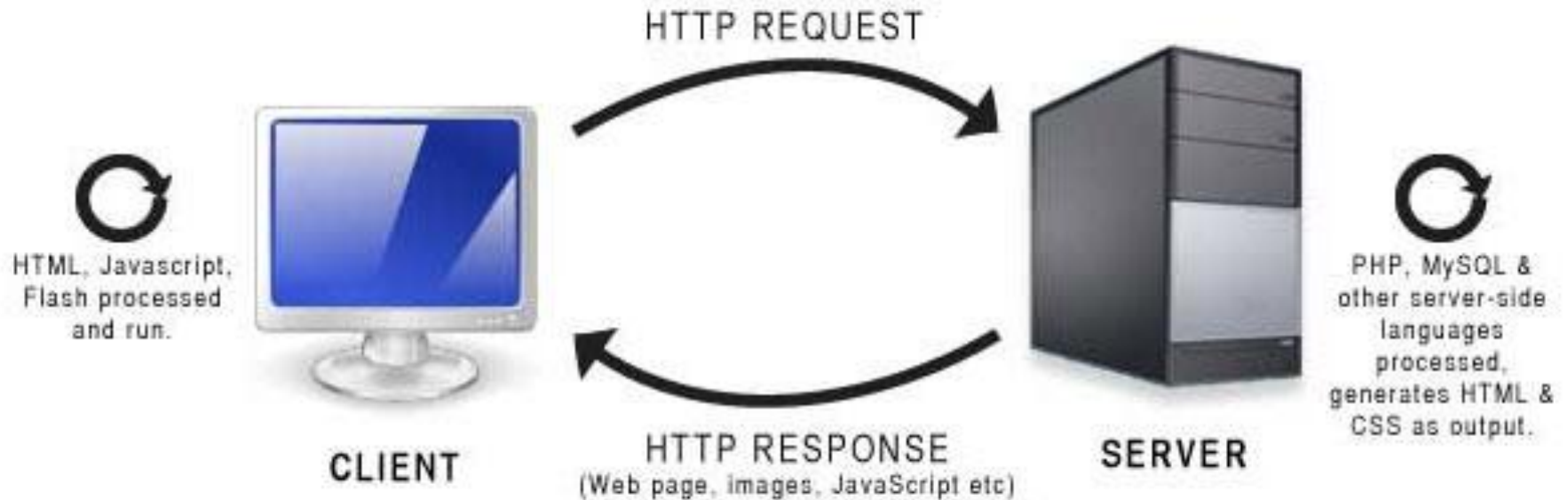
# Chrome: Javascript Errors

If your Javascript has errors, you'll see an indicator in the developer toolbar.



# Client & Server

# HTTP Request & Response



# What happens?

When a link is clicked, the browser creates an **HTTP Request** that is sent to the server.

```
GET http://weather.example.org/oaxaca HTTP/1.1  
Host: weather.example.org  
Accept: application/xhtml+xml
```

# The Server handles the request





### 3. The Server Responds

```
HTTP/1.1 200 OK
Content-Length: 45178
Content-Type: application/xhtml+xml; charset=utf-8

<!DOCTYPE html PUBLIC "...
<html xmlns="http://www...
<head>
  <title>5 day forecast . . .
```

# The Browser Displays the Page



# 404

PAGE NOT FOUND

[GO BACK TO HOME](#)

# Common HTTP Status Codes

**200** - OK

**404** - Resource not found

**500** - Server Error

# Status Code Ranges

**100s** are informational

**200s** are successes

**300s** are redirection (something moved)

**400s** are client errors

**500s** are server errors



Hypertext Markup Language

# What is HTML?

It is the content of your web page. If you have a blog, the HTML will describe the different sections of your page and will actually have the words of your blog post.

# What is HTML?

HTML is not style. It does not describe how the page is arranged, the color, or the background image. That is CSS.



# What is HTML?

HTML is made up nested tags. You start with the most general and nest it, becoming more specific.

```
<car>
  <engine>
    <transmission></transmission>
    <radiator></radiator>
  </engine>
  <stereo>
    <cd-player></cd-player>
    <fm-radio></fm-radio>
  </stereo>
</car>
```

```
<team>
  <defense>
    <defensive-backs>
      <corner-back>Richard Sherman</corner-back>
      <free-safety>Earl Thomas</free-safety>
      <strong-safety>Kam Chancellor</strong-safety>
    </defensive-backs>
  </defense>
  <offense>
    <wide-receivers>
      <wide-receiver>Doug Baldwin</wide-receiver>
      <wide-receiver>Golden Tate</wide-receiver>
    </wide-receivers>
  </offense>
</team>
```

```
<html>
```

```
  <head>
```

```
    <title>My first web page</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>My very first web page</h1>
```

```
    <p>
```

```
      This is pretty much the most awesome thing ever. Seriously.
```

```
    </p>
```

```
  </body>
```

```
</html>
```

# HTML Tags - Meta

- **<html>** - Encompasses your entire document.
- **<head>** - Where all your meta-data goes. Nothing in here gets displayed.
- **<body>** - Where all your content goes. This is the stuff that will be displayed.

# HTML Tags - Content

- **<h1>**, **<h2>**, ... **<h6>** - Headers or titles.  
h1 is the most important or “top level.”
- **<p>** - Denotes a stand-alone paragraph.
- **<div>** - A division or container of content.  
Used to group like objects together.

# HTML Tags - Content

- **<ul>** - An unordered list. These bullet points are an unordered list. Implies no order.
- **<ol>** - Ordered list. Any list that implies some order (usually has leading numbers.)
- **<li>** - An element of a list. In this case, an individual bullet point.

```
<h1>My favorite social media sites</h1>
```

```
<ol>
```

```
  <li>reddit</li>
```

```
  <li>Twitter</li>
```

```
  <li>Instagram</li>
```

```
</ol>
```



# Exercise

1. Create a new page that has a title.
2. Create an unordered list of things you look for in a car.
3. Create an ordered list of your favorite cars.
4. Give each a title

<http://codepen.io/btholt/pen/axvAd>

# HTML Tags - Inline

- **<strong>** - Usually bold. Used for something you want to stand out.
- **<em>** - Usually italics. Used for something you want emphasis on.
- **<span>** - Like div for inlines. Used for something you want to separate from other things. Becomes useful with CSS.

This is an `<strong>awesome</strong>`  
class. I `<em>love</em>` it.

<http://codepen.io/btholt/pen/wInbm>

# HTML Tags - Void Tags

Some tags don't need a closing tag; they can't have anything in them. In these cases, the tags close themselves. A good example is an input tag.

```
<input />
```

# HTML Tags - Attributes

Sometimes tags need additional meta-data.  
The `img` tag is a great example of that.

```

```

<http://codepen.io/btholt/pen/vewaA>

# HTML Tags - Grouping

- Like we said before, it's a good idea to group tags by some idea.
- If you were doing a blog post, you would group together individual blog posts.

<http://codepen.io/btholt/pen/njGdE>



# HTML - Classes

Those previous groupings were useful, but to someone who had never seen the code before, you wouldn't know what it was a group of. For that, we'll use classes.

```
<div class="picture-group">
```

```
  
```

```
  
```

```
</div>
```

<http://codepen.io/btholt/pen/ywAjc>

# HTML - IDs

- Classes can (and should be) used multiple times throughout a page. If you have multiple blog posts, you should multiple blog-post classes used.
- IDs are unique. There can only be one of an ID on a page. You would only have blog-post-1 ID or one blog-post-2 ID.

```
<div id="group-1" class="picture-group">
```

```
  
```

```
  
```

```
</div>
```

```
<div id="group-2" class="picture-group">
```

```
  
```

```
  
```

```
</div>
```

# HTML Tags - Naming

It's tempting to give your classes name like left-group or purple-container. Don't give them "presentational names." What if you need to move the left-group to the right? Or the purple-container is now green? You don't want to have to rename everything.

# Exercise - Giving Classes

1. Open <http://codepen.io/btholt/pen/njGdE>
2. Give appropriate class names to all the h1, p, and divs.
3. Give the blog posts appropriate IDs.



Cascading Style Sheet

# CSS - What is CSS?

- While HTML is the content, CSS is the style, the presentation of the content. CSS dictates how the HTML looks.
- CSS is a collection of rules. If certain are met, then a style is applied to it.



```
body {
```

```
  color: red;
```

```
}
```

<http://codepen.io/btholt/pen/eFoIn>

```
body {
```

```
  color: red;
```

```
}
```

```
p {
```

```
  color: green;
```

```
}
```

<http://codepen.io/btholt/pen/GFI mi>

# CSS - Better Practices

What we've done works so far. However, try adding another `<p>` to the previous pen. It will also be green. What if we want one `<p>` to be blue and one to be green?

<http://codepen.io/btholt/pen/vlryG>

# CSS - Classes

- Classes are used extensively for styling. Notice the leading period in `.leading-p:` that denotes that it is a class.
- There are different rules for styles “winning out,” or stated differently, when two styles conflict, which one gets applied. We’ll talk about it in a sec.

<http://codepen.io/btholt/pen/echKA>

# CSS - Text Properties

- `color` - Change the color of the text.
- `font-weight` - Change it from normal to bold.
- `font-style` - Change the text from normal to italic.
- `text-align` - Left, right, or center.
- `text-indent` - Indent paragraphs.
- `font-size` - How big the text is

# CSS - Measurements

CSS has a whole bunch of measurements for legacy reasons: px, pts, em, ex, rem, in, mm, cm, etc. With few exceptions, px and em are the two you want to use (depending what context you're in.)

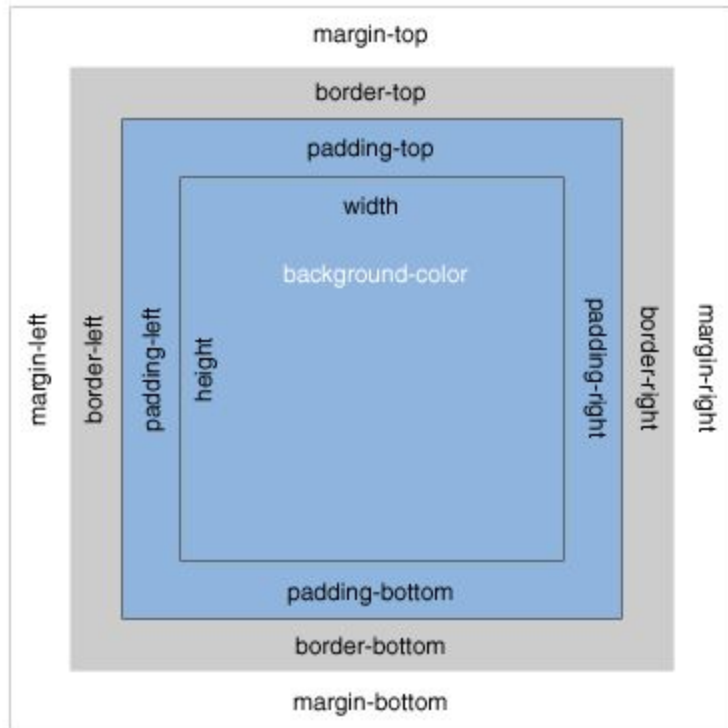
# CSS - Measurements

- px - Pixels. While not necessarily uniform across browsers (Firefox has a different pixel measurement vs Chrome) it will be uniform across your page.
- em - M's. It's how wide an 'm' is in the current font. Good for relative sizes that scale with a font.



# CSS - Boxes

- width / height - Obvious, hopefully.
- border - The border around your box. Examples:
  - 1px solid black
  - 3px dashed gray
  - 5px dotted #663399
- background-color - The color of the background
- background-image - A URL to the background image



<http://codepen.io/btholt/pen/DlsLE>

<http://codepen.io/btholt/pen/echKA>

# CSS - Document Flow

CSS has a concept called float. You can have divs “float” next to each other instead of each one on a new line. The browser will try to fit as many divs next to each other until there isn't any room left, at which point it'll wrap to the next line.

<http://codepen.io/btholt/pen/HamIj>

<http://codepen.io/btholt/pen/ebkJG>

# Exercise: Styling a Page

1. Open <http://codepen.io/btholt/pen/uHzfL>
2. Recreate <http://i.imgur.com/0dnCF58.png> without changing the HTML
3. Leave the box-sizing. It makes it easier.



**Picture #1**



Lorem ipsum dolor sit amet,  
consectetur adipisicing elit.  
Temporibus sapiente fuga,  
quia?

**Picture #2**



Commodi reprehenderit sunt,  
voluptatum, eius voluptates  
repellat optio similique  
dolore eos laborum!

**Picture #3**



Dolorem, fugiat voluptas  
nemo est error, rerum.  
Possimus unde non autem  
repudiandae!

**Picture #4**



Ipsa recusandae voluptates  
eligendi consequuntur ex  
unde reprehenderit corrupti  
assumenda, enim corporis.

**Picture #5**



Accusamus totam fuga  
ducimus aperiam, placeat  
similique vero voluptatibus  
repellendus quam eligendi.

**Picture #6**



Odio nesciunt aliquam,  
veritatis et esse magnam vero  
illo sint praesentium  
explicabo.

**Picture #7**



Tempore porro, consectetur  
amet illo accusantium  
voluptas culpa aut corporis  
incidunt doloribus!

**Picture #8**



Accusantium blanditiis,  
provident repellat. Earum  
optio natus, similique quis  
odio officia, neque!

[codepen.io/btholt/pen/uHzfL](https://codepen.io/btholt/pen/uHzfL)

[i.imgur.com/0dnCF58.png](https://i.imgur.com/0dnCF58.png)

**Solution:**  
[codepen.io/btholt/pen/kcuea](https://codepen.io/btholt/pen/kcuea)

# CSS - Conclusion

CSS, like HTML, has a lot of stuff to it. It's impossible to keep it all in your head. Don't expect to memorize all of it. Use sites like the [MDN](#), [CSS-Tricks](#), and [Stack Overflow](#) when you need help.

# JavaScript



## An Introduction to Programming

# How is JS different from HTML?

**HTML** tells your browser how a web page is going to look.

**Javascript** is **interpreted** by your browser.

# What is it used for?

Javascript is used for anything from creating a pop-up when a user clicks on something, to full fledged games.

# Statements

Each line in JavaScript is an instruction.

When the browser reads it, it executes that line of code.

```
console.log("Hello");
```

# Statements

; - what's this thing??

; is a semi-colon

think of a javascript statement as a sentence, and a semi-colon as a period.

# JavaScript - Comments

```
// Comments start with two forward slashes.
```

```
/*
```

```
Multi-line comments
```

```
can be long
```

```
*/
```



# Comments - Rule of Thumb(s)

- Comments can be used to explain complex or complicated code.
- There is no need for obvious comments
- Remember, when you go back and update your code, comments should be updated too.

# Seeing Results

```
// will popup a dialog
```

```
alert("hello popup");
```

```
// will output directly to the html page
```

```
document.write("hello document");
```

```
// will display in the console
```

```
console.log("hello console");
```

# Exercise: Seeing results

<http://codepen.io/nnja/pen/Inypi?editors=001>

# JavaScript - Variables

Variables are used to store values.

Declare and initialize on one line:

```
var y = 2;
```

or two:

```
var x;
```

```
x = 5;
```

# **Don't forget the var!**

**Always use the var keyword when declaring a variable for the first time.**

If you don't other parts of the code can accidentally overwrite it.

# JavaScript Variables - Naming

**Variables are case sensitive!**

Naming convention:

- Variables start with lower case letter, \$ or \_
- Don't contain special symbols % # !
- written in camelCase

# JavaScript - Reserved Words

Some words are reserved and can't be used as variable names.

Examples:

case

catch

for

let

private

this

with

in

do

# JavaScript - Variable Types

**Strings** - Used to store a series of characters.

```
var myName = "Nina";
```

**Numbers** - With or without decimal.

```
var year = 2014;
```

```
var pi = 3.14;
```



# JavaScript - Variable Types

## Boolean

True (Yes)

```
var isNice = true;
```

False (No)

```
var hasCake = false;
```

# JavaScript - Special Types

The value of a variable which is not yet declared is called **undefined**.

```
var cars; // Value is undefined
```

Variables can be emptied by setting the value to **null**.

```
person = null; // Value is null
```

# Our First Javascript

```
// Let's use Javascript to popup something about  
us
```

```
var myName = "Nina";
```

```
alert("My name is " + myName);
```

# Exercise

<http://codepen.io/nnja/pen/nzspH/?editors=0>  
01

# Operators

+

Addition

-

Subtraction

\*

Multiplication

/

Division

%

Modulus

# JavaScript - Expressions

Variables can also store the results of expressions.

```
var x = 2 + 2;
```

```
var y = 5 * 6;
```

```
var name = "Nina";
```

```
var greeting = "Hello " + name;
```

# Increment & Decrement

++

Increment

--

Decrement

```
var x = 5;
```

```
x++; // x will be 6
```

```
x--; // x will be 5
```

# JS - Comparisons

`==`

equal to

`===`

equal value and type

`!=`

not equal to

`!==`

not equal value or not equal type

`>`, `>=`

greater than, greater than or equal

`<`, `<=`

less than, less than or equal



# JavaScript - Equality Comparison

```
var numApples = 3;
```

```
numApples == 3 // true
```

```
numApples == "3" // true
```

```
// Strict Equality
```

```
numApples === "3" // false
```

# JS - Logical Operators

The image shows the JavaScript AND operator symbol, which consists of two ampersands (&&) in a yellow-orange color, enclosed within a dark blue square.

AND

The image shows the JavaScript OR operator symbol, which consists of two vertical bars (||) in a yellow-orange color, enclosed within a dark blue square.

OR

The image shows the JavaScript NOT operator symbol, which consists of an exclamation mark (!) in a yellow-orange color, enclosed within a dark blue square.

NOT

# JS - Logical Operators Example

```
var x = 5;
```

```
var y = 3;
```

```
console.log(x < 4 && y < 4);
```

```
>> false
```

```
console.log(x < 4 || y < 4)
```

```
>> true
```

# JavaScript - Arrays

Arrays are a list of variables.

They are written with square brackets. [ ]

```
var fruits = ["Peach", "Orange", "Apple"];
```

# JavaScript - Arrays

Arrays have a length property, so we can find out how many items are in them.

```
var fruits = ["Peach", "Orange", "Apple"];
```

```
fruits.length
```

```
>> 3
```

# JavaScript - Access Items in Array

Use square brackets to access an item in an array.

```
var myFruit = fruits[0];
```

Array access is **zero based**. To access the first element, use **[0]**.

```
console.log(myFruit);
```

```
>> Peach
```

# Exercise: Arrays

<http://codepen.io/nnja/pen/zCHtf?editors=00>

1

# JS - Change / Add item in List

```
var fruits = ["Peach", "Orange", "Apple"];
```

```
// Change item in 2nd position.
```

```
fruits[1] = "Pineapple";
```

```
console.log(fruits);
```

```
>> ["Peach", "Pineapple", "Apple"]
```

```
// Add Orange to List of Fruits
```

```
fruits.push("Orange") // Returns the new length of the array.
```

```
["Peach", "Pineapple", "Apple", "Orange"]
```



# JavaScript - If Statement

Use `if` to tell JS which statement to execute, based on a condition that is true.

```
var apples = 8;  
if (apples > 0) {  
    console.log("Eat An Apple");  
}
```

# JS - If Else

```
if (apples > 0) {
```

```
    console.log("Eat An Apple");
```

```
}
```

```
else {
```

```
    console.log("No apples left.");
```

```
}
```

# JS - If, Else, Else If

```
if (apples > 0) {
```

```
    document.write("Eat An Apple");
```

```
}
```

```
else if (apples < 3) {
```

```
    document.write("Go to the store.");
```

```
}
```

```
else {
```

```
    document.write("No apples left.");
```

```
}
```

# Exercise - The If statement

<http://codepen.io/nnja/pen/qfIBK?editors=00>

1

# JS - For Statement

```
for (<counter> ; <counting to> ; <increment counter>) {  
    <expression>  
}
```

- The **<counter>** is a variable used to keep track of what step you're on.
- The **<counting to>** is the goal. It's how many total steps we want to take.
- The **<increment counter>** is how we change the variable to get to the goal.

# JS - For Statement

```
var text = "numbers: ";
```

```
for (var i = 0; i < 5; i++) {
```

```
    text += " " + i;
```

```
}
```

```
console.log(text);
```

```
>> "numbers:  0 1 2 3 4"
```

# Iterating over Array

```
var fruits = ["Peach", "Orange", "Apple"];  
for (var i=0; i<fruits.length; i++) {  
    console.log(fruits[i]);  
}
```

*// We're counting up to fruits.length. It doesn't matter how long the array is.*

# Exercise - Iterate over an Array

<http://codepen.io/nnja/pen/zaqBl?editors=00>

1



# Functions

Functions are a way of repeating the same action multiple times.

Functions can be called multiple times.

# Anatomy of a Function



```
var printList = function(list) {
```

The diagram illustrates the components of a JavaScript function declaration. Four orange arrows point to the following parts of the code: the variable name 'printList', the keyword 'function', the parameter 'list', and the opening curly brace '{'.



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

The diagram illustrates the components of a JavaScript function declaration. Four orange arrows point to the following parts of the code: the variable name 'printList', the keyword 'function', the parameter 'list', and the opening curly brace '{'. An additional orange arrow points to the 'for' loop statement.



```
    console.log(list[i]);
```

The diagram illustrates the components of a JavaScript function declaration. Four orange arrows point to the following parts of the code: the variable name 'printList', the keyword 'function', the parameter 'list', and the opening curly brace '{'. An additional orange arrow points to the 'for' loop statement. Another orange arrow points to the 'console.log' statement.



```
  };
```

The diagram illustrates the components of a JavaScript function declaration. Four orange arrows point to the following parts of the code: the variable name 'printList', the keyword 'function', the parameter 'list', and the opening curly brace '{'. An additional orange arrow points to the 'for' loop statement. Another orange arrow points to the 'console.log' statement. A third orange arrow points to the closing curly brace '}' of the function body.



```
};
```

The diagram illustrates the components of a JavaScript function declaration. Four orange arrows point to the following parts of the code: the variable name 'printList', the keyword 'function', the parameter 'list', and the opening curly brace '{'. An additional orange arrow points to the 'for' loop statement. Another orange arrow points to the 'console.log' statement. A third orange arrow points to the closing curly brace '}' of the function body. A final orange arrow points to the semicolon ';' at the end of the function declaration.

# Running Functions

Declaring a function is different from a statement.

The code in a function will not be run until you explicitly call it.

# Functions Exercise

[http://codepen.io/nnja/pen/aumdH?editors=](http://codepen.io/nnja/pen/aumdH?editors=001)  
[001](#)

# Calling a function

```
var fruits = ["Peach", "Orange", "Apple"];
```

```
var printList = function(list) {
```

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

```
    console.log(list[i]);
```

```
  };
```

```
};
```

```
printList(fruits);
```

# Scope

Variables defined in functions can only be used in those functions.

So what happens when you forget to use `var`?

<http://codepen.io/nnja/pen/ocLDG?editors=0>  
01

<http://codepen.io/btholt/pen/pzBCI>

# JavaScript Objects

- Objects are collections are of properties. They can contain numbers, strings, functions, arrays, even other objects.
- They're useful for containing like-properties.



# JS Objects

```
var car = {  
  make: "Telsa",  
  model: "Model S",  
  acceleration: 30,  
  accelerate: function() {  
    this.accelerate += 10  
  }  
}
```



# JavaScript - Context

- Context refers to what this means.
- What this means depends on what context the function is called from, much like the preceding sign.
- Depending on where this called, it means different things. It can be source of bugs. Be careful.

<http://codepen.io/btholt/pen/BLtiv?editors=001>

<http://codepen.io/btholt/pen/iFodm?editors=001>

# Other topics not covered here

- do / while loops
- Inheritance
- Switch statements
- A boatload of DOM interactions (other than document.write)
- Newer “ES6” syntax
- Much, much more. JavaScript has been around a while and has a lot to it.



Using JavaScript to manipulate the DOM

# jQuery - What is jQuery?

- jQuery is JavaScript *toolkit*. It's JavaScript someone else wrote to make writing it easier.
- Its main purpose to make common tasks in JavaScript (like changing info on the page) easier and shorter.
- Used on some 80% of the top sites in the world.



# jQuery - Telltale Sign

If you see something looks

```
$(‘... stuff in here ...’)
```

then it's probably jQuery. jQuery uses the \$ a lot.

# jQuery - Simple example

```
$( '.caption-text' ).text( 'Magic!' );
```

<http://codepen.io/btholt/pen/aopKv>

# jQuery - Chaining

```
$( '.caption-text' )  
  .text( 'Magic!' )  
  .css( 'background-color', 'orange' );
```

<http://codepen.io/btholt/pen/wdgiB>

# Exercise - Set text

1. Open <http://codepen.io/btholt/pen/lbwul>
2. Using jQuery, make it so when the button is clicked, whatever text in the `<input/>` is set as the text of the `<p>`.
3. Do not modify the HTML.
4. Hint: you'll need `.text()`, `.val()`, and `.click()`.

Solution: <http://codepen.io/btholt/pen/Gyqri>

# jQuery - Responding to Users

```
$('.alert-btn').click(function() {  
    alert('Hey there!');  
});
```

<http://codepen.io/btholt/pen/ujlEz>

# jQuery - Get text from an input

```
var name = $(' .name-input' ).val();  
alert(name);
```

<http://codepen.io/btholt/pen/Elfjs>

# jQuery - Other Cool DOM Functions

- `.css()` - Get or update CSS values.
- `.html()` - Set the inner HTML of an element.
- `.show()` / `.hide()` - Displays / hides an element.
- `.addClass()` / `.removeClass` - Add or remove a class.
- `.append()` - Add an element to the existing elements in an element

# jQuery - Other functions

jQuery has over 300 functions. It's huge. And it has great documentation and a great community. If you want to know how to do something, just search for "jQuery how to hide div" and it'll come up. Stack Overflow is great.



# AJAX

- Stands for Asynchronous JavaScript and XML. It's really a misnomer. It's just a buzzword that stuck.
- Means making requests to a server without refreshing the page.
- jQuery makes it dead simple with its `.ajax()` method.

<http://codepen.io/btholt/pen/FArdh>

[http://codepen.io/btholt/pen/mjBk](http://codepen.io/btholt/pen/mjBkq)  
q

Putting some concepts together

# Exercise: Displaying Data from reddit

1. Open [http://www.reddit.com/r/aww/search.json?q=puppy&restrict\\_sr=true](http://www.reddit.com/r/aww/search.json?q=puppy&restrict_sr=true)
2. If you want to see the API documentation, it's here: [http://www.reddit.com/dev/api#GET\\_search](http://www.reddit.com/dev/api#GET_search)
3. Open the CodePen <http://codepen.io/btholt/pen/Aejsl>

## Hints:

- Solution: <http://codepen.io/btholt/pen/fErah>
- You'll need the jQuery methods `.click()`, `.append()`, and `.ajax()`
- You shouldn't need to touch HTML or CSS.
- The data is fairly nested. The data that concerns you is `data.children[i].data.thumbnail`. `children` is an array you'll loop over.

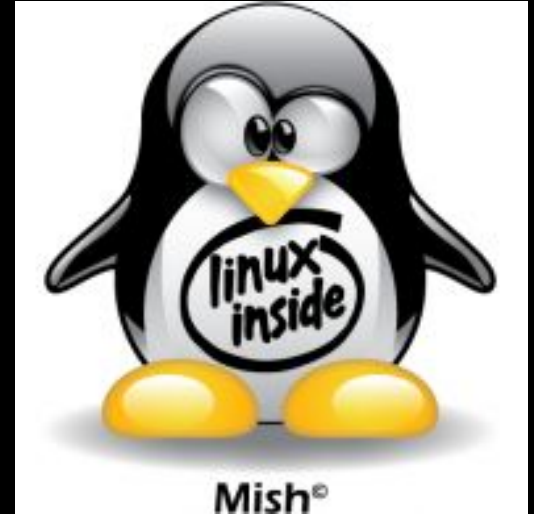
# Intro to Command Line

```
SLMIN .SYS 12P 20-Dec-85  VH .SYS 3P 13-Aug-86
XL .SYS 4P 20-Dec-85  LD .SYS 8P 21-Aug-86
SP .SYS 8P 13-Aug-86  WL .SYS 8P 13-Aug-86
RT11SJ.SYS 78P 13-Aug-86  BU .SYS 8P 13-Aug-86
WL .SYS 2P 13-Aug-86  TT .SYS 2P 13-Aug-86
SD .SYS 5P 31-Aug-86  BLOCPC.SYS 71P 21-Nov-84
BASIC .SAV 56 24-Aug-79  BLDCOM.SAV 24 20-Dec-85
BATIME.SAV 1 20-Dec-85  DIR .SAV 13 20-Dec-85
DWP .SAV 1 20-Dec-85  DWP .SAV 47 20-Dec-85
TSXDD.SAV 78 27-Nov-82  FORTAL.SAV 206 21-Aug-85
HARRIS.SAV 2 21-Dec-81  LET .SAV 5 20-Dec-85
START .P3G 151P 15-Aug-88
EXHA .STR 15P 02-Feb-83  RETRO .OBJ 22 17-Mar-82
STAMP .LIN 12P 15-Aug-83  TSXUCL.TSL 1P 04-Sep-85
EXHA .STR 19 11-Feb-83  JGFLIP.SAV 30 08-Mar-86
JGFLIP.FOR 3 08-Mar-86  RT11FS.SYS 8P 20-Dec-85
ANNOY .SAV 30 18-Apr-83  TSXP23.MEM 1200P 27-Nov-82
DUO .DIR 18 16-Jul-86  RL .DIR 7 16-Jul-86
DIR .DIF 26 16-Jul-86  DEHOF6.OBJ 1 1
DEHOF6.OBJ 1  -BIO-  EVAN .ID 1
114 Files, 8949 Blocks
14433 Free blocks
.OE
```

BU-85 12U 748  
digital VT100



# Windows vs Mac vs Linux

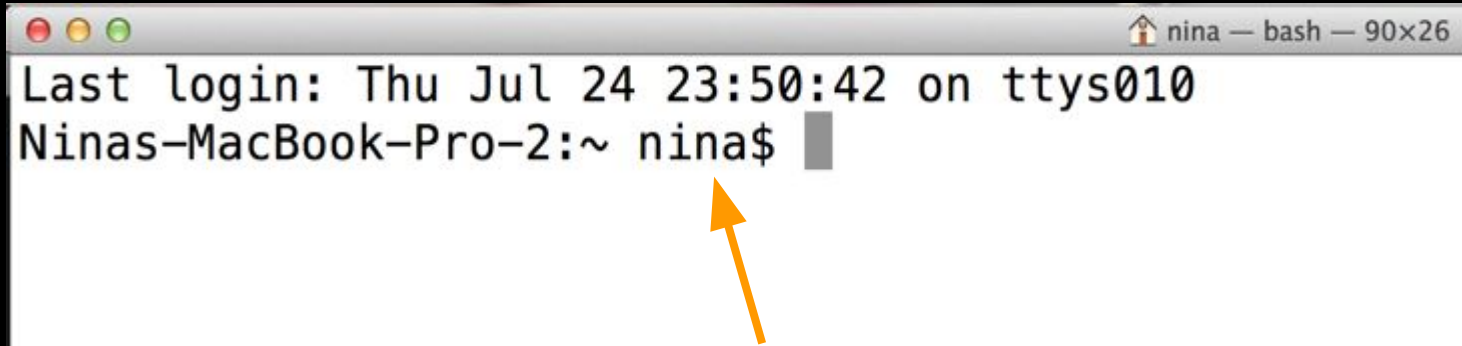


# Why do we still use it today?

- Completing repetitive tasks is faster
- Tasks can be automated
  - This is called scripting
- Parameters can be specified
  - Parameters are a way of customizing how a command is run.



# The Command Prompt



A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left and a status bar on the right that reads "nina — bash — 90x26". The terminal content shows the login message "Last login: Thu Jul 24 23:50:42 on ttys010" followed by the prompt "Ninas-MacBook-Pro-2:~ nina\$". A grey rectangular cursor is positioned at the end of the prompt. An orange arrow points from below the prompt up to the space between the tilde (~) and the username (nina).

```
Last login: Thu Jul 24 23:50:42 on ttys010
Ninas-MacBook-Pro-2:~ nina$
```

# Home Directory

Mac/Linux:

```
$ cd
```

```
/Users/<username>
```

Windows:

```
C:\Users\<username>
```

# What directory am I in?

## Linux/Mac OS

```
$ pwd
```

```
/Users/nina
```

```
(pwd stands for  
print working  
directory)
```

## Windows

```
$cd
```

```
(cd stands for  
current directory)
```

# Listing Directory Contents

```
$ cd ~/Desktop
```

```
$ ls
```

```
folder/
```

```
file.txt
```

```
$ dir C:\windows
```

# Case sensitivity

File names in linux and mac os are case sensitive.

That means `myfile.txt`  $\neq$  `MyFile.TXT`

UNLESS, you use windows.  
Windows don't care.



# Navigate to a different Directory

linux/mac os

```
$ cd Desktop
```

windows

```
$ chdir C:\windows
```

tip: cd stands for  
change directory

# Navigate up one directory

```
$ cd ~/Desktop/painting/
```

```
$ pwd
```

```
/Users/nina/Desktop/painting
```

```
$ cd ..
```

```
pwd
```

```
/Users/nina/Desktop
```

# Paths - Relative/Absolute

The starting for relative paths is the directory you are in.

The starting for absolute paths is the root directory. It's just `/` on linux, or `C:\` on windows.





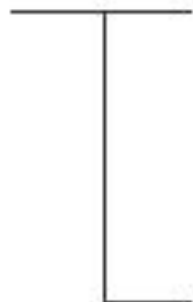
WWW



a.html



XXX



b.html



c.html

# Git / Github

Git is an open source version control system.

- Advantage: Maintain history of changes
- Can use a remote server for backup

**Github.com** is a popular website with features built on top of this tool.

You can find tons of free projects!

# Forking

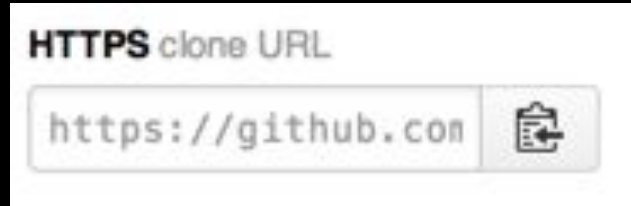
<https://github.com/btholt/pull-requests>



# Cloning a Repository

Navigate to a github.com Repository.

Copy Clone URL:



# Cloning a Repository

Open your terminal, and navigate to a folder you'd like to save files in.

type:

```
git clone <your repo url>
```

This will copy the files to your computer.

# Git Status

This is the git command you will use the most.

```
$ git status
```

```
On branch master
```

```
Your branch is up-to-date
```

# Unstaged Changes

Create a file called <your first name>.txt in the pull-requests directory. Write a fun fact about you in the file, and save it.

```
$ git status
```

```
Untracked files:
```

```
nina.txt
```

# Git Add

```
$ git add nina.txt
```

```
Changes to be committed:
```

```
new file:   nina.txt
```



# Git Commit

```
$ git commit -m "yay! my first commit"  
[my_branch 1edbb31] my first commit  
1 file changed
```

# Origin


At this point, our version controlled changes are only on our local machines.

We want to push our code up to github.

```
$ git push origin my_branch
```

```
* [new branch] my_branch -> my_branch
```

# Creating a Pull Request

 Create pull request

Choose different branches or forks above to discuss and review changes.



# Initializing a New Repository

```
$ mkdir project-folder
```

```
$ cd project-folder
```

```
$ git init
```

```
$ Initialized empty Git repository in  
/Users/nina/project-folder/.git/
```

```
$ git status
```

```
On branch master Initial commit
```

# Feature Branches

```
$ git checkout -b my_branch
```

```
Switched to a new branch 'my_branch'
```

```
$ git status
```

```
On branch my_branch
```

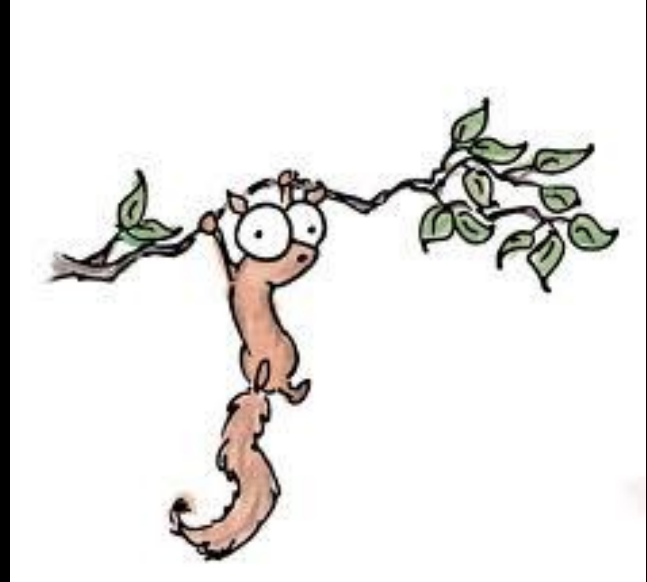
```
nothing to commit, working directory  
clean
```

# Feature Branches

```
$git branch
```

```
master
```

```
* my_branch
```





JavaScript on the Server

# JS on the server? wut?

- node.js takes Google Chrome's JavaScript engine (V8) and uses it to control libenv, a library that allows you to create and run a server.
- It's super rad and pretty fast.



# Who uses node.js?

- Walmart
- eBay
- PayPal
- LinkedIn
- HP
- New York Times
- Klout
- Microsoft
- Groupon
- Yahoo!
- Uber
- Pinterest
- Mozilla
- Flickr

# node.js - What is it?

- We're going to use JS to code the server. This is useful because you only need to learn one language
- You're using node.js instead of Python, Ruby, Java, PHP, etc.
- node.js is quite different from other server-side languages.

# Hello World

Let's do the most basic app, Hello World.  
Folder: `node-exercises/basic`

# npm - node package manager

- Remember how we used jQuery which is just code written by someone else to make your life easier? npm makes it even easier to bring in other people's code.
- Great news! You already have it because you installed node.js.

# npm

- Let's make it easier to make changes to our program using npm.
- In the terminal, type:

```
npm install -g nodemon
```

# nodemon

- nodemon is a development tool that every time you make code changes, nodemon restarts your server, making it easier to code.
- npm, by default, installs everything locally. We want to be able to use nodemon anywhere, so we used the -g flag, which makes it install globally.

# Express

- Like jQuery, Express is a library designed to make writing code easier.
- Express makes writing node.js easier at by a billion. Maybe a trillion.
- Folder: node-exercises/express

# Exercise - cheer and jeer

1. Make an app that has two routes:  
/cheer.txt and /jeer.txt.
2. /cheer.txt should send back something positive to say.
3. /jeer.txt should send back something negative to say.
4. Solution is in node-exercises/cheer-jeer
5. Remember to npm install express



# Static Assets

- HTML, CSS, JavaScript, images, and fonts are considered to be “static assets” because they aren’t changed by the server; they are served exactly as they are saved.
- As such, it would be a pain to write a route for every image. Instead we have “static” or “public” directories which items are served exactly as they are.
- Let’s look at `node-exercises/static-assets`
- Notice that even the CSS and JavaScript are loading okay!

# Receiving Parameters

- Often we have 1 page that will serve multiple routes.
  - Example: /team/jazz/ and /team/timberwolves/ will both use the same page but load different information on those pages
- node.js makes that pretty easy.
- Look at node-exercises/params

# npm - more

1. npm can track your app's dependencies, or libraries that must be there for your app to work.
  - a. Example: Our apps all depend on Express.
2. When you have a lot of dependencies, you can't remember them all. Good thing npm can keep track.
3. Let's create a dependency file for params.
4. Delete node\_modules
5. Run `npm init`
6. Run `npm install express --save`

# npm - even more

1. Now that params has a package.json file (it's always called that in node.js), let's see why that's useful.
2. Delete the node\_modules directory
3. Run `npm install`
4. Notice you didn't tell npm to install Express; it just knew which one to install.
5. Take a look at package.json

# node.js - POSTing

- Often we want to send more than a few parameters in a request to the server. We can do this via POST.
- We'll use jQuery to send up a user's login credentials.
- Folder: node-exercises/posting

# node.js - POSTing

- Notice that we had another dependency, body-parser. Express aims to be flexible and allows you to use different parsers. Most of the time you'll just use body-parser as it does a great job
- We're also using HTTP status codes. Remember 404 - Not found? That's a status code. Some other useful ones 200 - request successful, 401 - unauthorized request, and 403 - forbidden request. There are a lot.

**Our App: Twitter**

copy the link from  
<https://github.com/btholt/intro-to-webdev-app>  
p

\$ git clone  
<https://github.com/btholt/intro-to-webdev-app>  
p.git





# Requirements

- Write a server in node.js. The server will
  - Serve the HTML, CSS, and JS necessary to run our app
  - Accept GET requests of the latest tweets
  - Accept POST requests to post a new tweet and store it to be served later via a GET request
  - For each new tweet, the server will attach the time of when it was posted

# Requirements

- Create a web app that will:
  - Make an AJAX request to GET new tweets
  - Have the ability to accept user input to POST a new tweet to the server
  - Will display in a pretty way the tweets when the server loads
  - After posting a new tweet, it will display the new tweet too.

# Things we will not do

- Store anything in a database. If the server crashes or gets shut down, we will lose all our tweets.
- Have any notion of users, follows, retweets or anything like that. Just anonymous tweets.

# Bye!

Nina Zakharenko  
@nnja



Brian Holt  
@holtbt

