# Class 2: Styling

3D buttons: using gradients

CSS – select everything: \*

## Document Object Model (DOM)

Everything that can have something be applied to it.



## Nested Selectors

Two selectors at the same time. Like an address. A set of instructions telling CSS property as they are looking for a CSS property.

thead tr {

{

select only tr in the thead

<em> emphasis

font-style -> normal or italic

font-weight -> bold or not

user agent stylesheet -> browser stylesheet

## Layout

<div> - means “document part/section.” In practice, it’s a part of looking at something like a box.

<header>

<footer>

<aside>

<section>

<article>

## Class and ID:

<tag name attribute name=”value”>

IDs are unique

Classes are not unique

Usually use classes. IDs are really best used for navigation and javascript situations

In CSS:

.className

#idName

Dot means class.

Much better to put a class on elements rather than globally apply.

### HTML Structural Elements

Header tags – h1 especially – if it’s just sitting loosely on the page, you can have precisely 1 of them if you want them found by google. The exception is putting it in <header> <aside> <footer> etc.

## Floats

Document flow: top to bottom, left to right

Float is a CSS positioning property, used to layout a web page. Float takes it out of the document flow.

Has no height impact on the flow. Doesn’t move where you expect sometimes.

### Dimensions

px

% - percentages relative to its parent container

em – relevant to the font-size of its parent. Hard to use because of

in – inches. Print css

cm – centimeter

pc - pica

rem – relative to the html tag. Not properly supported

CSS:

Overflow: hidden; -> lops off overflow so it hides any that the container cuts off

## Reset

Reset and Normalize.

Goes first, before stylesheet.

<link rel=”stylesheet” href=”css/reset.css”>

<link rel=”stylesheet” href=”css/stylesheet.css”>

Webfont

Link font

<link href=’http://fonts….>

Googlefonts and typekits

<https://www.google.com/fonts>

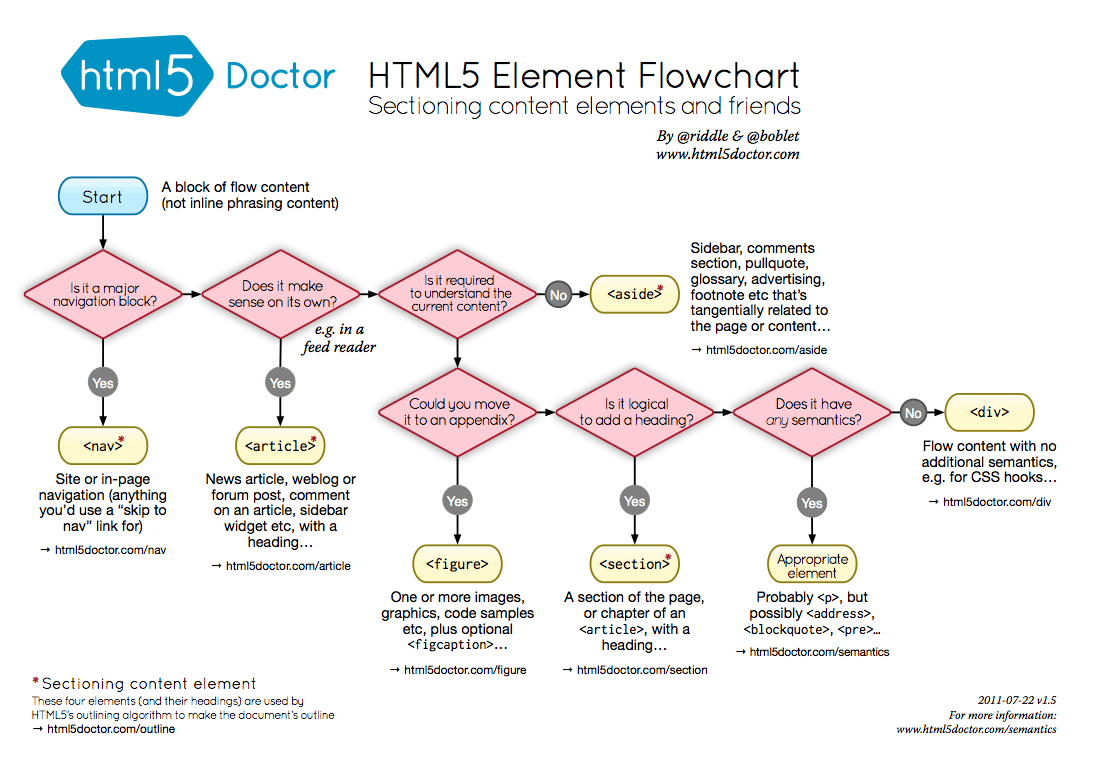
<link href='https://fonts.googleapis.com/css?family=Roboto' rel='stylesheet' type='text/css'>

<https://fortawesome.github.io/Font-Awesome/>

paste BEFORE your style sheet – all fonts, resets, icons from the web

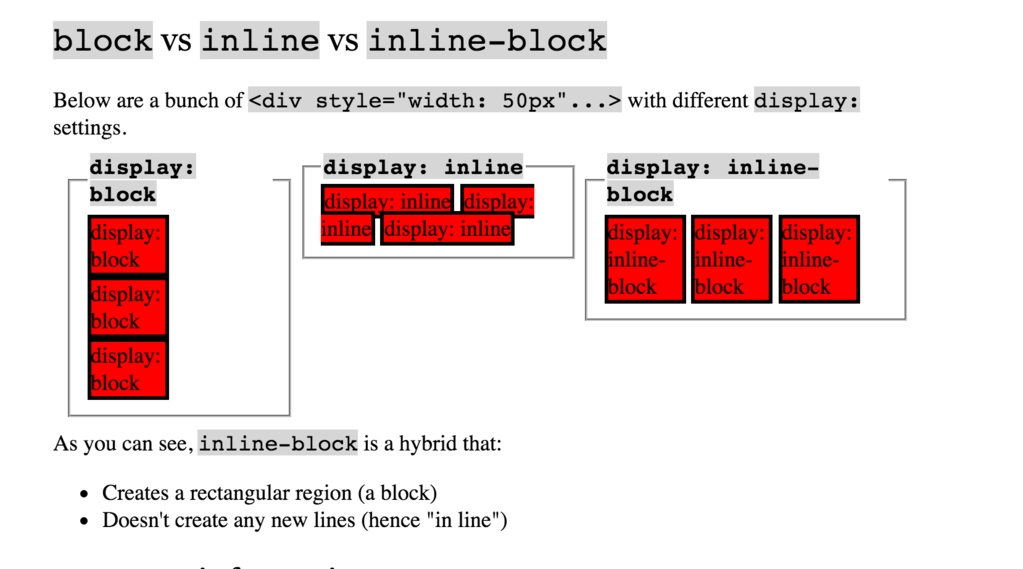
../ -> up a folder

# Class 3: LayouT



Block-level element makes it go across the page

* Inline constantly flows with your text (like span, anchor)



Containers/wrapper/wrap

Margin: 0 auto; Needs a width

background-attachment:fixed; -> gives a parallaxes effect

a:hover -> pseudo selector

also there’s focus pseudo selector (when you focus a selector/input field

To

Footer nav li a {

Padding0riight:5px;

Padding-left: 5px

Border-left: solid 1px black;

Transition: background .25s ease;

}

First-child:

Footer nav li:first-child a {

Padding-left:0;

}

footer nav li:last-child a {

border-right: none;

}

# Class 4: Intro Programming

JavaScript: acts like an object-oriented programming language

Collapsible/Accordion

Video-handling -> javascript

Fancybox.net (modal – popup of image, turns background dark grey) – hard for disabled or for google search or users, etc.

jQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML.

1. Let’s you target the page the exact same way you do with CSS (easy way to query your doc)
2. AJAX is when your page asks your server for the content and puts it on your page without reloading. Like google when you’re typing in the search bar, autofills in. Had to write your code differently for different browsers. JQuery came up with common AJAX things that worked for all browsers. AJAX issue has lessened over time.
3. Provides animation (before html 5 and css 3)

Adding jQuery to your website: <script src=”js/jquery-1.8.3.min.js”></script> - adding the file or

<<script src=”//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery….> -> find google CEN version

CEN- content delivery network.

Whenever possible, grab the CEN.

Check for bugs.

Grabbing 1 – older/stable version. Grabbing 2 – new and trendy, but isn’t supported on all browsers. 2 is faster.

Put it above your script.

jQuery can take plugins.

jQuery syntax - select: selectors are just like css

$(“.class”).on(‘click’);

Note: document.getElementById(‘thing’).onclick =doSomething;

Function doSomething() {

//something happens

}

In jQuery, this might look more like:

$(‘selector’).on(‘click’,doSomething);

function doSomething() {

//something happens

}

jQuery click event: .on(‘click’)

$(“#stopLight”).css(“background-color”,”teal”);

OR you can write in no conflict mode:

jQuery(“#stopLight”).css(“background-color”,”teal”);

-Recommendation is to do the jQuery on bc it will ALWAYS work, whereas $ sometimes work

nesting:

jQuery(“body”).children(‘.bulb’).css(“background-color”,”black”)

Content does NOT belong in the JavaScript. Content belongs in your HTML.

Function name() {

jQuery(“body”).css({

“background-color” : “gray”,

“color” : “white”

});

# class 5: programming

Variable is a bucket you can put data in. “Data” can mean anything.

Primitive data types:

* boolean: true or false
* null – assigned a value of “no value”
* undefined – variable has been declared but not assigned a value
* numbers
* string – anything between single or double quotes

## Variable Conventions:

* variables start with a lower case letter
* if they contain multiple words, subsequent words start with an upper case letter (camelcase)

Data Types can be converted

Var intString = “4”;

Var intNumber = parselInt(intString);

Var floatString = “3.14159”;

Var floatNumber = parseFloatNumber….

Equal (==)

Not equal (!-)

Strict equal (===)

String not equal (!==)

Greater than (>)

Great than or equal (>=)

Less than (<)

Less than or equal (<=)

## Conditional

If (condition is true) {

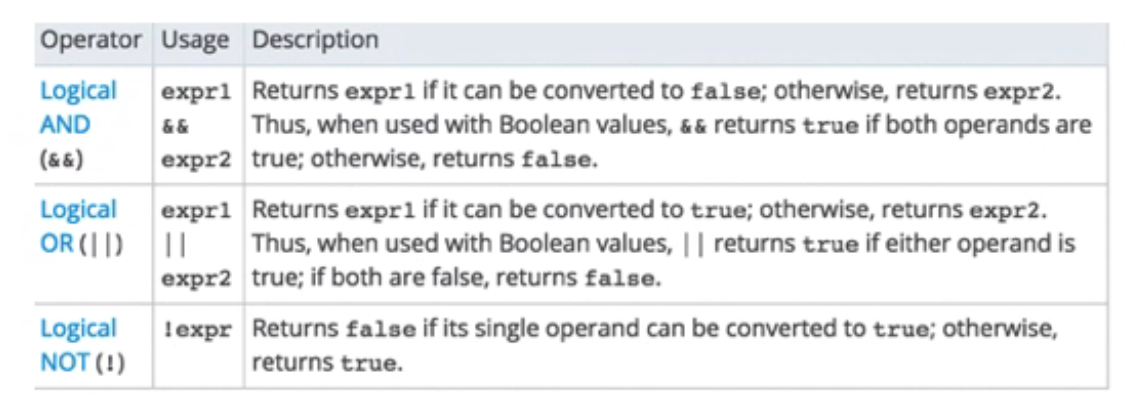
//Do cool stuff

} else {

//Do other cool stuff

}

## Logical Operators



if(name ==”GA” && password == “YellowPencil”)

{//Allow access to internet}

if (day == “Tuesday” || day == “Thursday”) { //We have class today}

## functions

### function calls

Function helloWorld () {

Console.log(“Hello Functions”);

}

helloWorld(); //Prints “Hellow Functions” to the console

The brackets execute the function.

Functions also don’t need “;” at the end

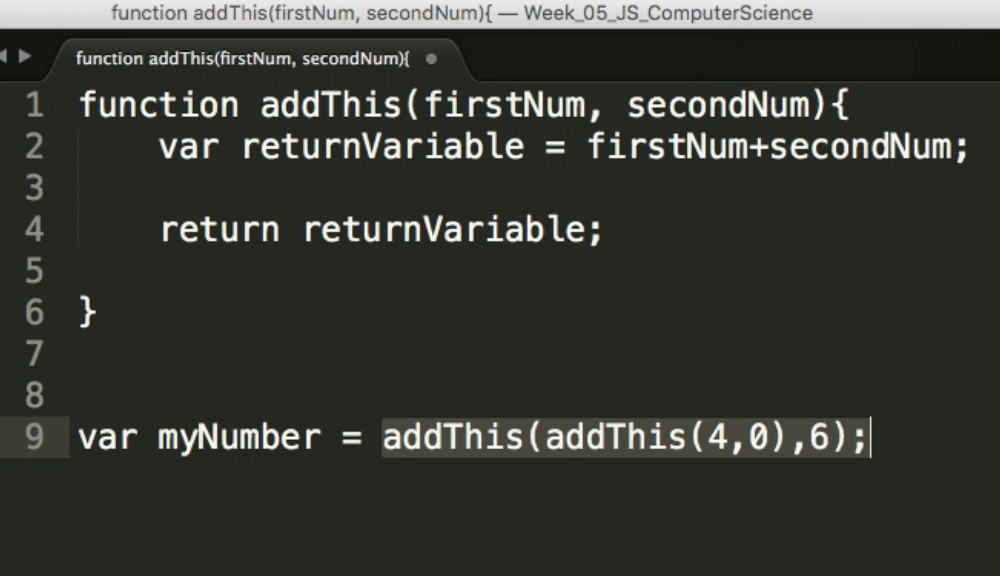
### Function arguments



try not to use optional arguments bc it blocks up code.

### Return Variables

Allow us to nest variables



### Organizing Function

Functions must exist before you call the function

## Array

Arrays are buckets with multiple variables

Var daysThisYear = 366;

Var myNewArrayofHappiness = [3,255,12,daysThisYear];

You can nest arrays within arrays

How to access array nameOfArray [ position# ]: i.e.: myNewArrayofHappiness [1]

There Is .length: myNewArrayofHappiness.length would give us 4

No associative arrays in javascript.

### Loops

Loops let you iterate – let you run code while something is true. For instance, let’s add a number to every part of our array:

For(var q=0;q<myNewArrayofHappiness.length;q++){myNewArrayofHappiness [q] +=22;}

in the { // do something to myArray[q]}

Don’t use .forEach bc it’s slow and doesn’t work on older browsers!

# Class 6: Review Refractor

Arrays and objects

Arrays: everything is condensed to a number. Arrays have []

Object can have properties, like color, a word, etc. Objects have { }. Objects can contain objects, arrays, etc.

Var productData = {“NikonCamera”:

{

“features”:[“digital”, “full frame”, “20 megapixel”],

“name”: “D700 DSLR”

},

“CanonCamera”:

{

“features”:[“digital”, “full frame”, “24 megapixel”],

“name”: “EOS 1D mkIII DSLR”

},

}

productData.NikonCamera

productData[“NikonCamera”]

* Both those work to get the features from those objects
* Class for loops don’t work on objects
  + Two ways of doing a “each” loop

jQuery.each(productData, function(index,value){

for(f=0; f<value.featuers.length;;f++){

console.log( “Product: “+index+” – “+value.features[f]”);

}

});

* Index would be Nikon camera and canon camera; value would be the properties

## JSON

Jsonviewer.stack.hu