Table of Contents

[Week\_1\_W 3](#_Toc505021264)

[Git 3](#_Toc505021265)

[GitHub 3](#_Toc505021266)

[GitHub for course 3](#_Toc505021267)

[Course on Site 3](#_Toc505021268)

[Pull and push 3](#_Toc505021269)

[Websites are really just collections of files: 3](#_Toc505021270)

[HTML 3](#_Toc505021271)

[HTML Anatomy for Meta 4](#_Toc505021272)

[HTML Anatomy for Body (Marking our content for semantics) 4](#_Toc505021273)

[Comments 5](#_Toc505021274)

[Older browsers 5](#_Toc505021275)

[Goals before next class: 5](#_Toc505021276)

[Questions 5](#_Toc505021277)

[Week\_2\_M 5](#_Toc505021278)

[CSS 5](#_Toc505021279)

[Color 5](#_Toc505021280)

[RGBA colors 6](#_Toc505021281)

[Adobe color system 6](#_Toc505021282)

[Fonts 6](#_Toc505021283)

[Size 6](#_Toc505021284)

[Question 6](#_Toc505021285)

[CSS 7](#_Toc505021286)

[Add CSS to your page 7](#_Toc505021287)

[Images 7](#_Toc505021288)

[Relative path 7](#_Toc505021289)

[Absolute path 7](#_Toc505021290)

[Full URL to an external system 7](#_Toc505021291)

[Read a link 8](#_Toc505021292)

[Image file formats 8](#_Toc505021293)

[Answers to questions 8](#_Toc505021294)

[Week\_2\_W 9](#_Toc505021295)

[Div 9](#_Toc505021296)

[Div p 9](#_Toc505021297)

[CSS can be hooked onto HTML’s: 9](#_Toc505021298)

[Class 9](#_Toc505021299)

[ID 10](#_Toc505021300)

[Nesting 10](#_Toc505021301)

[Selectors\*\*\*\*\* 10](#_Toc505021302)

[Select multiple elements 10](#_Toc505021303)

[Only select an immediate child inside a parent 11](#_Toc505021304)

[Select exact match for both tag and class combination 11](#_Toc505021305)

[Use case 11](#_Toc505021306)

[Font weight 11](#_Toc505021307)

[EM and Strong 11](#_Toc505021308)

[CSS 11](#_Toc505021309)

[To check if something will work in all browsers 11](#_Toc505021310)

[Font licensing 11](#_Toc505021311)

[CSS normal or reset 12](#_Toc505021312)

[JavaScript and Web fonts 12](#_Toc505021313)

[Box Model 12](#_Toc505021314)

[Content box model 12](#_Toc505021315)

[Week\_3\_M 12](#_Toc505021316)

[div 12](#_Toc505021317)

[span 13](#_Toc505021318)

[Width of the screen 13](#_Toc505021319)

[Sketch 13](#_Toc505021320)

[Box Model 13](#_Toc505021321)

[Margin 13](#_Toc505021322)

[Dev tools 13](#_Toc505021323)

[To see the actual size of an image: 14](#_Toc505021324)

[Hero 14](#_Toc505021325)

[Resources 14](#_Toc505021326)

[Footer 14](#_Toc505021327)

[Browsers 14](#_Toc505021328)

[Normalized 15](#_Toc505021329)

[Multiple classes 15](#_Toc505021330)

[Semantic html (e.g., header, footer, nav) 15](#_Toc505021331)

[Class & tags 15](#_Toc505021332)

[Balancing between being generic and precise 15](#_Toc505021333)

[Layout methods 15](#_Toc505021334)

[For office hours, ask: 16](#_Toc505021335)

# Week\_1\_W

## Git

* Is a version control system
* Repository – a folder with all the files you are working on that stores your edits
* Not great at any large image

## GitHub

* Website that stores gits
* Bitbucket is another platform that stores gits
* Once you put something in it, it’s hard to get it out
* Can do desktop or command line version
* You can see what has changed
* *Always have a summary of what you changed*
* Can delete/destroy repositories BUT not files (easily)

## GitHub for course

* <https://github.com/ga-students/fewd-dc-36>

## Course on Site

* <http://fewd.kasigi.com/FEWD-DC-36/>

For homework copy over files from GitHub into your own homework folder

* (so you can commit changes)

## Pull and push

* Pull – pulls from internet and brings down to the local
* Push – pushes from local and pushed to the internet
* You should pull on a regular basis (every day)

## Websites are really just collections of files:

* HTML (.html) content & structure
* CSS (.css) style and visual design
* JavaScript (.js) interaction & behavior

## HTML

* HTML5 is HTML with a few additions
* The DOCTYPE tells you if the page is HTML5 ready
* <!DOCTYPE html>
* It was originally designed to share academic papers and mark them up
* The big change occurred in 2009 – when HTML5 came out
* W3C governs this (and CSS) – how you’re supposed to do it but not necessarily implemented (e.g, there is CSS for the hearing impaired)

## HTML Anatomy for Meta

* Everything is tagged – it has a beginning and an end
* DOCTYPE doesn’t follow normal HTML standards but it tells you what dialect of HTML you are speaking
* <html></html> -- open and close
* <head> contains information about your document (often just copied from document to document)
* <html lang=”en”> is an attribute saying this is English
* charset – character set – so universal typeface
* One tag you need in the head is the title
* Second most important is styling sheet
* Meta tag is for search engine

## HTML Anatomy for Body (Marking our content for semantics)

* HTML syntax open comment close
* Attributes are always attribute name=”value”
* *You can have as many attributes as you want and the order of them doesn’t matter*
* H1 –H6 – there’s no such thing as a H7
* Don’t have a header 2 if you don’t have a header 1 – for 508 compliance and google
* <p></p> paragraph tag
* <code> This is some computer code</code>
* <address>This is street address</address>
* <ul>This is a list but it’s unordered</ul>
* <Li> Second item </li>
* <ol>Communicates these things have to be in this order</ol>
* <a href="https://en.wikipedia.org/wiki/Cattle">Wikipedia</a>
* Image tag <img src=”” alt=”” > (a self-closing tag)
  + Source says where the image is (because not actually in html – you have to link to where it lives) – “images/CH\_cow\_2\_cropped.jpg”
    - If you start with a slash that means start with the basics of the website
    - Always assume that it is case sensitive
  + Alt is the text description of the image (you must have it as it gives 508 compliant and also for government) -- if the image is purely decorative, then you don’t have to have it
    - It has to be 125 characters

## Comments

* Html <!-- -->
* If working in a complicated problem, instead of writing everything out in advance, write out comments and then type things out

## Older browsers

* Government
* Korea
* Won’t render webpages correctly

## Goals before next class:

* Push and pull 3 repositories
* Fix all the things that are on the resume
* Do homework!

## Questions

* How do computers actually grab html and CSS and javascript
* Where does python and php fit in to all this

# Week\_2\_M

## CSS

* CSS – cascading style sheets
* Separates content from structure
* Rule: whatever the selector is gets this set of properties and values
* So, affect all paragraphs, this is what you want to affect and this is how you do it
* Order of punctuation {:;}
* You can describe it in sets so you have multiple declarations in one rule
* Color property – color effects test

## Color

* Can specify color by word (but you can’t always trust a browser to show the same red – so not as precise)
* RGB (0,55, 55)
  + (0,0,0) that’s black
  + (255,0,0) it’s red
  + (0,255,255) it’s teal
* hexcolors - #first two digits red, second two green, third two blue
  + #005555

## RGBA colors

* Takes a 4th value called the alpha – where totally visible to partially visible
* 0 is invisible
* 1 is solid
* background-color (rgba (255,0,0,0.5)

## [Adobe color system](mailto:https://color.adobe.com/create/color-wheel/)

* Use this to find color values
* It shows you the RGB and Hex

## Fonts

* Font family
* Font weight – sets the thickness of the font, values 100 -900
* Font style – normal or italic
* Size
* Text decoration – e.g., there is no underline tag in html

h1 {

color: #005555;

font-family: Times, Times New Roman;

text-align: center;

font-size: 16px;

}

h1 span {

text-decoration: underline;

color: rgb (200,100,0);

}

## Size

* Pixels (px) for fonts
* Percentages – use for boxes
* Inches – use it for printing invoices

## Question

* What does syndication actually look like?
* How do you grab it?
* How do domain names work?
* How do you convert a photo from PNG to JPEG to SPG?
* Get the color from another source with like an eyedropper

## CSS

* Top to bottom
* Least specific to more specific

## Add CSS to your page

* Inline CSS – there is an attribute in html called style where you can manual set the style
  + There is a use case where you have to do it this way -- email
* Style tags – supposed to be in the header – useful for troubleshooting but not ideal
  + Another use case for this is when you are working in a content management system – it will build a lot of the pieces over and over again (e.g., Drupal, Wordpress)
* External files with a link tag e.g., style.css
  + <link rel=”stylesheet” href=”style.css”> -- this is self-closing
    - rel tells you what kind of relationship it is
    - href is your file
    - type is a historic relic – dates back to when a lot of pieces of browsers were questionable
* Can have multiple CSS files for same website

## Images

* Is one of the most common self closing tags
* Two attributes – the src (where to find the image) and the alt (which describes the image) – can’t have tags in alt, just plain text
* Order to the attributes doesn’t matter

## Relative path

* A relative path – how to find it relative to the file you are editing
* You can find it easily – except if you move your files!
* You can write ../ which means go up a level – if your file had moved up a level

## Absolute path

* Professional websites use this but not personal
* It is a massive link from your hardrive

## Full URL to an external system

* This is always valid
* Downsides are you are relying on an external system – so what if they take that url down
* Also security! You are relying on outside files – so you might be worried that websites could be stealing your customers files

## Read a link

* Protocol, domain name, file path
* Protocol – http and https – the s stands for secure
* http – all that data goes through many serves on the internet and is not encrypted
* https aren’t that much slower anymore

## Image file formats

* PNG – they have loss less compression – so this means you can shrink the image but get it back exactly the way you had it before – also support transparency – when you want a detailed logo that is transparent on the website, you would use this BUT they are bigger in size so a bigger download (also wonderful for shapes) – make images translucent – good for curves (e.g, aliasing)
* GIF – animated gifs (they can move – which PNGs can’t)
* JPEG is great for photos – can get the photo down nice and small, can control how aggressive it is, and most of the photos on the internet are JPEG
* SPG – unlike the others which are rastor files, these are vector – and it means that you can scale this really, really big – since it is math driven, it is hard for computers – because it’s so big – SPGs shine for use as icons, but not really as backgrounds

## Answers to questions

* Developer tools
* Waterfall
  + Every single element on your page
  + Shows how long it takes to do things
  + Network speed – waiting time – the server had to get the pieces
    - Content management job is to assemble the pages
  + Download time how fast your internet is
* PHP and Python run on the server
  + Python is more heavy duty than PHP
  + PHP is really only for websites
  + Python you need to learn more to use it
  + Your choice of web hosts is much more limited
  + PHP is the wonder bread of the web world
* Syndication
  + RSS
  + Your website might spit out an xml file (looks a lot like html but tags are different)
  + Takes your website data and spits the data out in a structure you understand
  + Take the data out of your database and assemble it and spit it out
* For colors
  + If on a mac go to applications, then utilities, digital color meter application – whatever you hover on you get the rgb values
  + For precision branding work, inspect it
* PNG to a JPEG
  + Photoshop is personal favorite, much easier
  + ImageMagick – it’s a great program that can be frustrating -- this is good for bulk conversation – you’ll learn a lot about color spaces – you might see things converting weirdly and you might notice color shifts, photoshop in command line basically
  + Command called convert cover sloth.jpeg sloth.png
  + When you go to png to jpeg, you’ll want to think about compression settings
  + Convert sloth.png sloth2.jpeg—quality=100

# Week\_2\_W

## Div

* The div tag doesn’t have strong specific meaning
* It makes sections
* You can put other tags inside of them
* Shell around something we can group
* Useful if you want to turn a whole bunch of paragraphs green

## Div p

* Only style the paragraph inside the div

## CSS can be hooked onto HTML’s:

* Tag
* Class
* ID

## Class

* Are not unique
* Generally, use classes instead of IDs
* Can be uppercase – is usually case sensitive
* Don’t use generic names
* Can have multiple classes, separated by a space
* Defined by a . (e.g., .somethingSpecial)
* Could have a class called sidebar so that every sidebar on the site is the same
* Good to have styling associated with class (not tags)
* Can do .somethingSpecial
* Can do .somethingSpecial p if nesting

## ID

* Can only use 1 time on the page
* Can have both a class and ID on the tag
* ID must be unique per page
* Defined by a #

## Nesting

* You are nesting a selector to get really specific
* A global target will give you a hard time
* **Don’t style tags globally!!!!! It creates technical debt.**
  + Technical debt – when you do something badly you have to make modifications that creates a risk and additional price to pay down the road
  + More work by making a bad choice in the past
* You will want to use a nested selector instead

## Selectors\*\*\*\*\*

* Select Multiple Elements

Use a comma

.myClass, p, h2 {}

* Select element only inside another element

.myClass p {}

* Select element only when it's an IMMEDIATE child of a particular element
* .myclass > p {}
* Select elements that are exact match for both tag AND class combination
* p.myclass {}
* Find “error” class inside “message” class

.message .error

## Select multiple elements

* Can select multiple elements with a comma
* .somethingSpecial p, .somethingSpecial li {} – the comma works as an “and” (but they are fully independent)
* A plus sign is a little more unique than the comma, that just applies to multiple elements at once

## Only select an immediate child inside a parent

* .somethingSpecial > p
* somethingSpecial .nothingIsSpecial >p (this is the targeting of a paragraph in the class nothingIsSpecial but only when inside somethingSpecial)
* The > means how many levels deep the nesting is
* Greater than sign is about more precision

## Select exact match for both tag and class combination

* P.nothingIsSpecial means find a paragraph that is in this
* Find me a paragraph of class nothingIsSpecial

## Use case

* When you would do that

## Font weight

* Can do boldest, bolder, bold, and light
* But can also do numbers – up to 900
* The font has to have that face avalaibile

## EM and Strong

* Bold and italtics don’t use it
* Instead use em – emphasis (italics) or strong (bold)

## CSS

* If they are equal the farther down will win
* If they are more precise, the more precise one will win

## To check if something will work in all browsers

* Use “Can I use \_\_\_\_?”

## Font licensing

* Adobe type kit allows you to use fonts on the web (but it caps out how many vistors can see the font)
* Google fonts – free and have no limits
  + When you select, you customize – select all the weights you think you will need
  + Each weight is a copy of the font, so it slows down your load time
  + Pretty rare to need thin (usually you want regular, bold, and light)
  + Embed text – this is a stylesheet – you can copy it and then put it in your html!
  + Add it above your own style sheets – because that makes it available for us in your styles
  + Then to assign the font to a style you have to do font-family
  + But it needs the web – so will it always work on just the phones? Nope.
  + You also don’t have control over that system – so you wouldn’t want to put banking information in there per say

## CSS normal or reset

* Every browser has its own default stylesheet built into it
* This problem was noticed in 2005
* Developers made this giant CSS blob that makes everything common
* You link to it like a stylesheet
* But it must be the FIRST thing that you link too – because if it is second or third thing you think, then that will override yoru changes
* Reset zeroes out your spacing
* Normalize – your bullets will still be your bullets
* Most major web projects have something like this linked in the very beginning!
* Usually put this in the CSS folder because don’t ever want this to change

## JavaScript and Web fonts

* Link external because so large

## Box Model

* Every element in web design is a box.
* Margin (e.g., space outside of your borders)
* Border (e.g., always at least 1 pixel)
* Padding (e.g., space between your content and your border – also part of your background color)
* Content/image (e.g., div)
* **Total width space wise is the sum of all these!**

## Content box model

* Margin – 10 to the left and 10 the right

# Week\_3\_M

## div

* Generic container
* Used to group elements for styling purposes

## span

* Equivalent to a div but within line

## Width of the screen

* Target for the smallest screen that your users are using
  + This is why there is a lot of whitespace on your screen
* Most people use 1280px by 800px

## Sketch

* Design tool (mac only)
* Used for wireframes
* Has H1 and H2s (etc.)

## Box Model

* Content box model – everything the content outward – adds up content, padding, border, margin
  + Box-sizing: content-box;
* Border box model – everything from the border inward (so now only dealing with width and margin) – here it’s everything but the margin so the content, padding, and border are included
  + Box-sizing: border-box;

## Margin

* Clockwise (so top, right, down, left)
* Margin: 20px 5px 5px 5px

## Dev tools

* Been part of the main browsers for about 8 years
* Best ones are chrome
* Right click on an element, click inspect, we have our toolkit
  + In here, we have a whole collection of tabs
    - Elements
      * On the left - html – can see how this page has been assembled
      * On the right can see the styles
        + Ones with the strikeout in them means that it has been overridden
        + The most specific is more powerful than the last CSS
    - Styles
      * Can actually go into the styles and change elements, just through typing in “inspect” – allows you to preview and debug
      * Also, can turn the rules on and off – by unchecking elements
        + Can be debug to see if this is causing your problem
        + “Inherit” means take whatever my parent’s element is
    - Computed
      * Browser says these are the styles I’m actually applying to that element – you can see the padding and dimensions for that element
    - Console
      * Javascript can be done in this
    - Network
      * This tab is useful because it is a list of every single file on this page in the order loaded and how long it took to load
      * Green – backend being slow
      * Blue – front end being slow (e.g., image size)
        + Can optimize with Acrobat (not enough saying file save – want to run through Adobes optimizers to make it smaller)
        + Should be 6 megabites or less for images

## To see the actual size of an image:

* Open in a new tab
* CSS can shrink and design images (go to console and kick on the box model to see how larger it is with the CSS on it)

## Hero

* Main image/slider
* Is essentially the “junk drawer” – where you can put anything on
* You stuff content that is important to the organization – but doesn’t belong elsewhere (e.g., replacing core mission or log in)

## Resources

* Where internal stakeholders can store things that don’t damage your content structure

## Footer

* Generally, people only go to the footer when searching for something specific
* Good for putting things for internal political movements

## Browsers

* All browsers are different
* No standard for what a h1 should look like
* Developing consistent experience == headache
  + You want to reset or normalize
    - Reset strips all default styles (e.g., mobile application)
    - Normalize isn’t as dramatic – the format is just the same across browsers
    - IMPORTANT: It’s like stretching your canvas – you don’t want to change this once you’ve done this. You just want it – in the very beginning – so your site is the same across browsers.

## Normalized

* Download and save the file – save it in your folder
* <link href=”normalize.css” rel=”stylesheet”>
  + It ALWAYS goes first – before your other CSS files.

## Multiple classes

* Can only have 1 attribute in tag so it looks like class=”message error” even though message and error are two different classes

## Semantic html (e.g., header, footer, nav)

* In each of these it resets your h1 runners:
  + header
  + footer
  + nav
  + main (only use it once on your page! It is the point of the webpage)
  + section (divide your page into sections)
  + aside (related content, ads)
  + article – general guidance is that if you take the article tag out of the website – it is a cohesive unit (don’t need to know anything else on the page for that to make sense)
* These behave exactly like divs

## Class & tags

* Always have a class involved
* Never style tags naked

## Balancing between being generic and precise

* .error .criticalCode
* .error code.criticalCode (so tied to the tag “code”)

## For office hours, ask:

* Go through the selectors we discussed
  + Mnemonics to remember?
* Github on the command line
* Sketch
  + How to create a wireframe?

## Layout methods

* Tables
* Floats
* Flexbox
* CSS Grids

## Tables

* Tables are for email layout only
* Don’t use it for webpages

## Floats

* It used to be that this was the only way to do this
* Now there are more advanced things (e.g., flexbox)
* A lot of websites and components are using floats (e.g., need to know for when you’re the mechanic on old websites)
* When you float something, you take it out of the document flow – and you say “float to the right” and ‘float to the left”
* You can have a div that wraps around 2 other divs so that other divs sit on top and below them

## Flexbox

## CSS Grids

* Too new, really cool
* Only the current versions of browsers support them
* Not enough browsers support this yet

## Document flow

* All your elements sitting against each other
* It reads:
  + Top to bottom
  + Left to right

## Example of float

* <main class=”floatWrapper”>
* <div class=”floatColumn”
  + <p></p>
* </div>
* <div class=floatColumn”
  + <p></p>
* </div>

In CSS:

* Make them smaller
* .floatWrapper {width: 50%; float: left; overflow: hidden}
  + This means he is kicking things to the right

Easiest way to get everything to align

* Float everything in the same direction

## Floating

* Original reason floats exist is that you might want text to wrap around a picture
  + But because they were the best option for layout – a lot of designers were using them for layout
* Taking an object out of the document flow and floating to the right or the left
* Left column 150 px and right column is 150 px
* So the height of the document flow is 0 now – because everything is taken out of the document flow
* What this means is that if you hide overflow – all your content/all your boxes disappear because your document float is 0 – making everything else invisible

## Clearfix

* Prevents things from disappearing on you
* You have to use this when you bump into a problem with your document flow not showing anything
* You use this by writing .floaterWrapper:after {content:””; display: block; clear: both;} – this is inserting a space under the last floated element – so that way the document flow has to expand to contain it
* This example works on internet explorer 8
* You can also just have a class called clearfix – so you can do “floatwrapper Clearfix”

## Overflow

* What happens if a div contains something that is larger than itself?
* If a div is 50% wide and inside the div you have a paragraph – how wide is it going to be? What happens if the paragraph thinks it should be wider than the div?
* If you say overflow hidden, it says none of my children can be bigger than me
* You need to constrain the boxes to make sure nothing is being knocked out

Draw your layout before coding!

* That way you can figure out where your wrapper should go

## How to center things

* Margin: 10px auto 10px auto;
  + If things have a specific width because it’s saying top -10 right -auto bottom-10 auto-left

# Week\_3\_W

## Max-Height and Min-Height

* Safe way to shrink images down and make them larger
* It won’t break the layout if the image is too small or too big and won’t break the layout

How to center margins

* 16px auto; [auto means margins are automatically scaling]

Inspect to find errors!

## \*

* Wild card
* *Why do you need a hack for border box?*

# Week\_4\_M

## Container to contain

* div-class = “hero”
* job is to provide a space where you know what the width will be because different otherwise depending on display screen
* Like placing all the elements inside a fish tank
* Common classes – how can you be lazy?

## Placeholders

* Lorem Ipsum
  + Dates back to print errors (faux Latin text)
  + [www.lipsum.com](http://www.lipsum.com)
  + You can tell it how many paragraphs/items you want
* Placeholder images
  + <http://via.placeholder.com>
  + Neutral and bland
  + Won’t offend anyone
  + Shows designers the space they have

## Pseudo Elements

* Can apply a style to a link if visited
* Anything with a beaker ignore because they are experimental features
* Check all these before you use – on the website Can I Use

/\*

Target the first letter of the first paragraph

Give it a font weight of 800

A size of 80px

and a line-height of 80px

\*/

.headerParagraph:first-letter {

font-weight: 800;

font-size: 80px;

line-height: 80px;

float: left;

color: red;

}

/\*

Target the first paragraph. It should have a weight of 600 and be the color darkgoldenrod

Says find me any tag that is a direct child of pseduo text -- find me any that you find.

\*/

.pseudoText > \*:first-child {

color: darkgoldenrod;

}

/\*

Target the links.

Links should have the color teal with a weight of 600;

A visited link should have the color mediumvioletred.

\*/

.pseudoText a:link {

color: teal;

font-weight: 600;

}

.pseudoText a:visited {

color: mediumvioletred;

}

## Display

* Block
* Inline – margin doesn’t work well here
* Inline-block –
* Flex
* By default, every element will have a default display

## Flexbox

* Display: flex
* Flex-basis – what is the basis for the browser to make the columns (e.g., Flex: 1)
  + 1 unit wide in this case for each
  + Whatever your flex is times the number of divs, section, etc. for it
* Make the first child 2 times the other – you can do
* Can reverse the order of the flexboxes

.colorBox: first-child {

Flex:3;

}

## Justify Content

* Space between so the place between articles or images is equal
* Space around is similar – equal space on all sides
* Center

## Liquid & Responsive

* Liquid always moves
* Responsive moves within specific sets