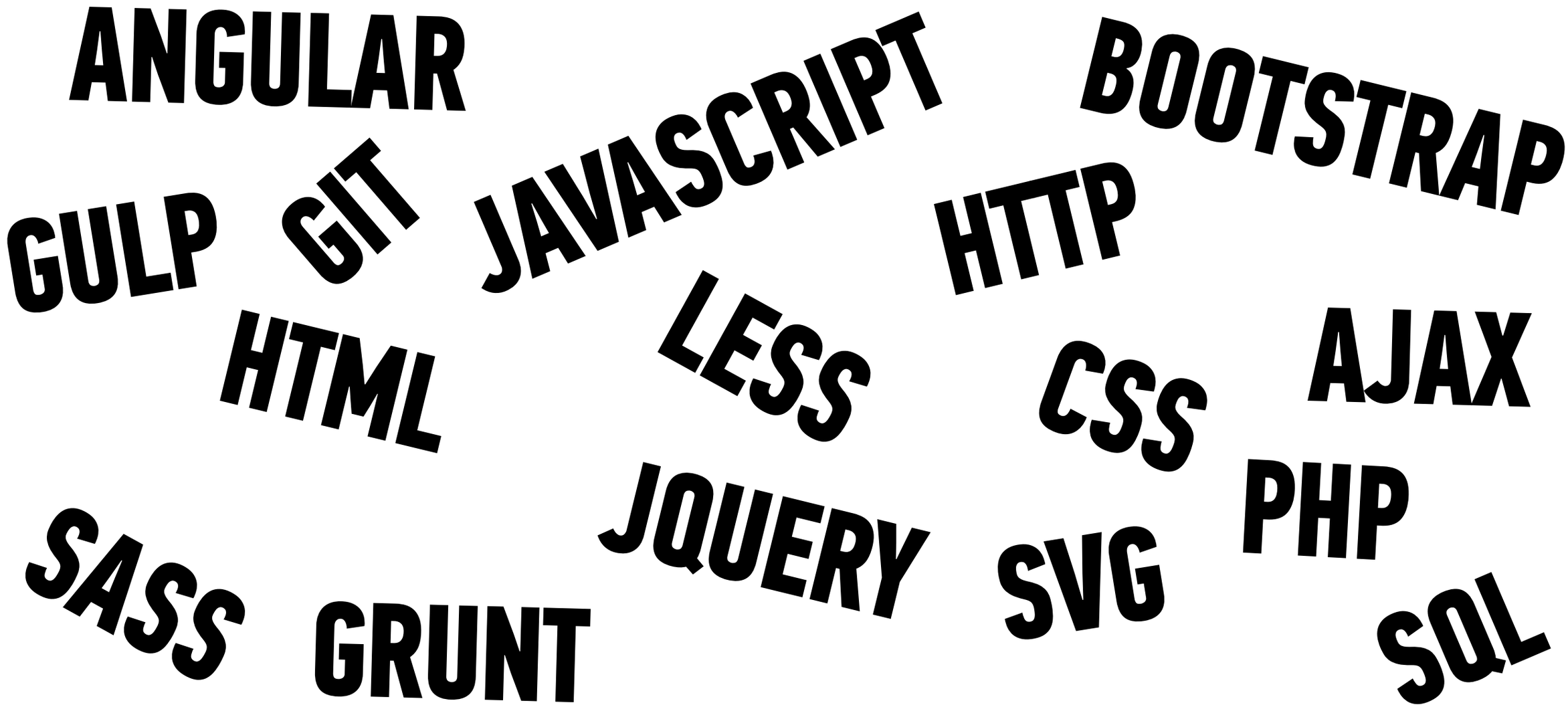
GENERAL ASSEMBLY

HTML & CSS BASICS: HANDS-ON CODING

Matt Soria, Front-end Web Developer

SO MUCH!!! WHAT IS ALL OF THIS?!!



WHAT'S THE PLAN?

AGENDA

- Web Development Overview
- HTML
  - What is it?
  - Anatonmy
  - Code-along!
- CSS
  - What is it?
  - Anatonmy
  - Code-along!
- Lab!
- What's next?
- Where can I learn more?
- Let's talk! (Q&A)

OBJECTIVES

- Understand the components of web development
- Examine Front End Web Development in more detail
- Gain an understanding of HTML & CSS syntax
- Build a basic web page!
- Discover some resources to help with moving forward
- Have fun!

WHAT YOU NEED:

DOWNLOAD SUBLIMETEXT: SUBLIMETEXT.COM/3

DOWNLOAD CHROME: GOOGLE.COM/CHROME

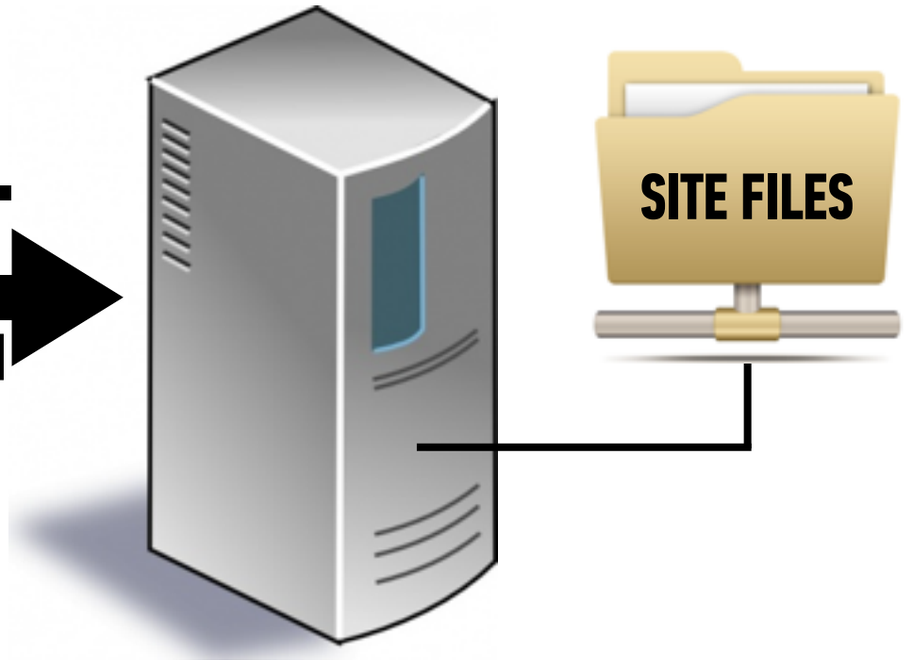
DOWNLOAD THIS TEMPLATE FOR A ROUGH STARTING POINT: GITHUB.COM/POOPSPLAT/GA-STARTHERE

HOW DO WE MAKE A WEBSITE?

HOW DOES A WEBSITE WORK?

HTTP REQUEST

[VIA INTERNET CONNECTION]



CLIENT

SERVER

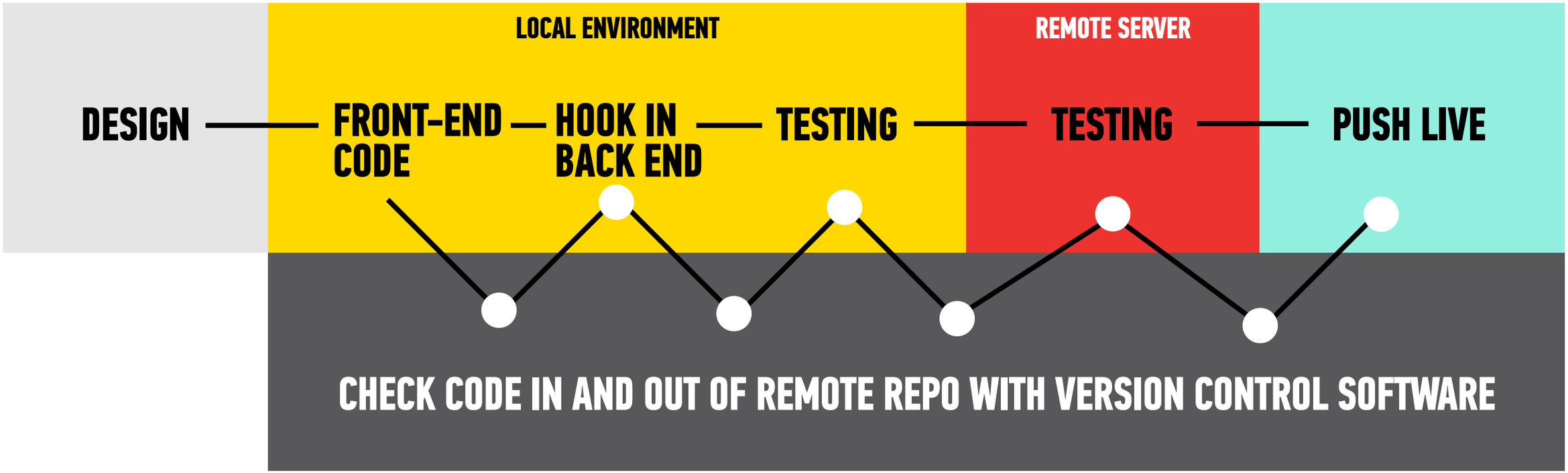
FRONT END

- Code is rendered on the client (your web browser)
- Describes content, presentation and behaviour
- Examples: HTML, CSS, JavaScript

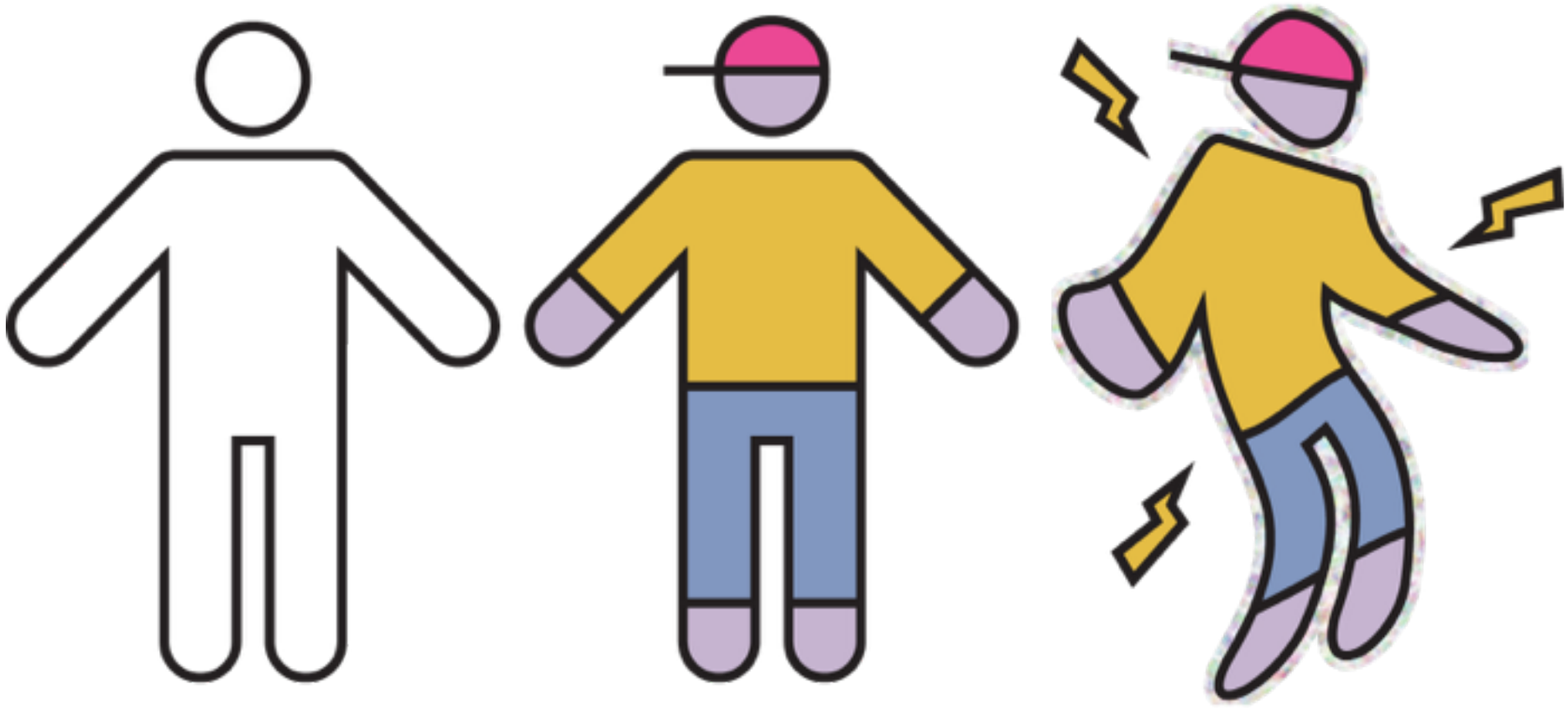
BACK END

- Code is interpreted on the server
- Handles state, data stores, other business logic
- Passes content to the front-end before rendering in the browser
- Examples: Ruby, Python, PHP

SIMPLE STANDARD WORKFLOW



SEPARATION OF FRONT-END TASKS



HTML + CSS + JAVASCRIPT

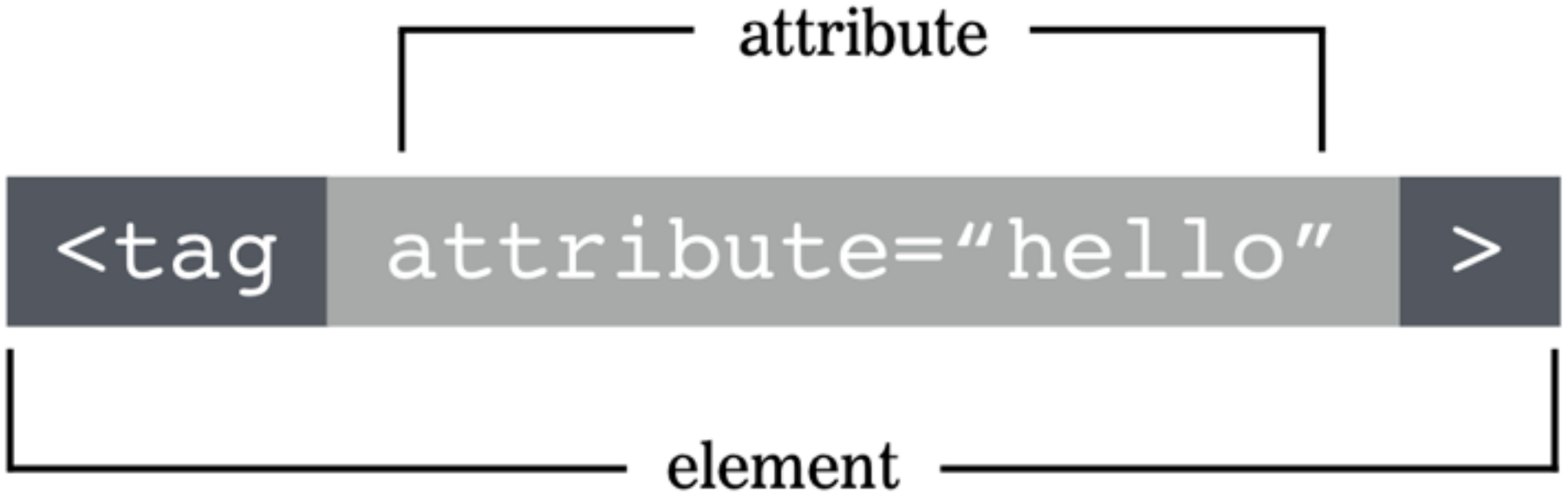
HTML: HYPERTEXT MARKUP LANGUAGE

WHAT IS THIS HTML THING?

- It describes our page's content in a way that the web browser can understand and derive semantics from
- It's made up of "tags" that surround pieces of our content to describe their context
- It's hierarchical – tags can contain other tags
- Part of mastering HTML is understanding which tags should be used for which types of content

TAG SYNTAX

SELF-CLOSING TAG SYNTAX



BASIC HTML MARKUP

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>My First Website!</title>
5    </head>
6    <body>
7      <h1>Hello World!</h1>
8      <p>This is a paragraph on my website!</p>
9      <a href="...">This is a link to another website!</a>
10   </body>
11 </html>
```

DOCUMENT TAGS

Tag	Usage	Notes
doctype	`<!doctype html>`	Tells the browser that the document is HTML5. Should be the first tag in the document
html	`<html>...</html>`	Parent element for all other elements of the page
head	`<head>...</head>`	Parent element of all metadata – anything here doesn't get rendered to the screen
body	`<body>...</body>`	Parent element of page content – anything here can get rendered to the screen

META TAGS

These tags should live inside the <head> element.

Tag	Usage	Notes
title	<title>My Page</title>	Sets the title for the web page. Rendered by the web browser in the tab, and by search engines
meta	<meta charset="utf8">	Can set various metadata attributes of the page, such as character set.
meta	<meta name="viewport">	Used to set device width, initial scale, and other viewport-related settings

CONTENT TAGS

These tags should live inside the <body> element.

Tag	Usage
p	`<p>Hello, this is a paragraph.</p>`
code	`<code>bleep, bloop</code>`
anchor	`<a href="http://generalassemb.ly">General Assembly</a>`
strong	`<strong>This is so bold.</strong>`
em	`<em>This is so emphasized!</em>`

HEADING TAGS

These tags should live inside the <body> element.

Tag	Usage
h1	`<h1>First level heading.</h1>`
h2	`<h2>Second level heading.</h2>`
h3	`<h3>Third level heading.</h3>`
h4	`<h4>Fourth level heading.</h4>`
h5	`<h5>Fifth level heading.</h5>`
h6	`<h6>Sixth level heading.</h6>`

LIST TAGS

These tags should live inside the <body> element.

Tag	Usage

unorder list	`<ul>`
	`<li>I'm an unordered list item!</li>`
	`</ul>`

ordered list	`<ol>`
	`<li>I'm an ordered list item!</li>`
	`</ol>`

list item	`<li>I'm a list item!</li>`

HTML5 STRUCTURAL TAGS

These tags should live inside the <body> element.

Tag	Usage
header	`<header><h1>A site or section header!</h1></header>`
nav	`<nav>…(list of site links)…</nav>`
main	`<main>…(the main stuff)…</main>`
section	`<section>A distinct, heading-worthy section</section>`
article	`<article>Independent content (think blog post)</article>`
footer	`<footer>…(Footer-type content for page or section)…</footer>`

CODE ALONG: BASIC MARKUP

IMAGES

HTML IMAGES

- Images use the `<img>` tag
- They have one required attribute: `src`
- The `src` attribute tells the browser where the image is located
- The `width` and `height` attributes can resize images in the browser
- An `alt` attribute provides alternative text to non-visual clients

```
<img src="hurley.jpg">
```

IMAGES

Plain ol' image

```
<img src="hurley.jpg">
```

Image with alternative text – better!

```
<img src="hurley.jpg" alt="Hugo 'Hurley' Reyes">
```

WHY IS ALT TEXT IMPORTANT?

- It provides search engines context to what the image is – good for SEO
- An image may fail to load for a variety of reasons:
  - There was a connection error
  - The file was not found
  - The user is using a screen reader – they may be visually impaired
  - The user is using a text-based browser, such as an old WAP phone or a non-graphical browser like Lynx

CODE ALONG: IMAGES

CSS: CASCADING STYLESHEETS

CSS: CASCADING STYLESHEETS

- Adds the presentational layer to our pages
- Provides the browser with rules for how our content should be displayed, by attaching styles to the HTML we write
- All our CSS needs to go in a .css file, and that file be referenced from our HTML
- Is the most fun of the front-end languages

ATTACHING OUR CSS FILE

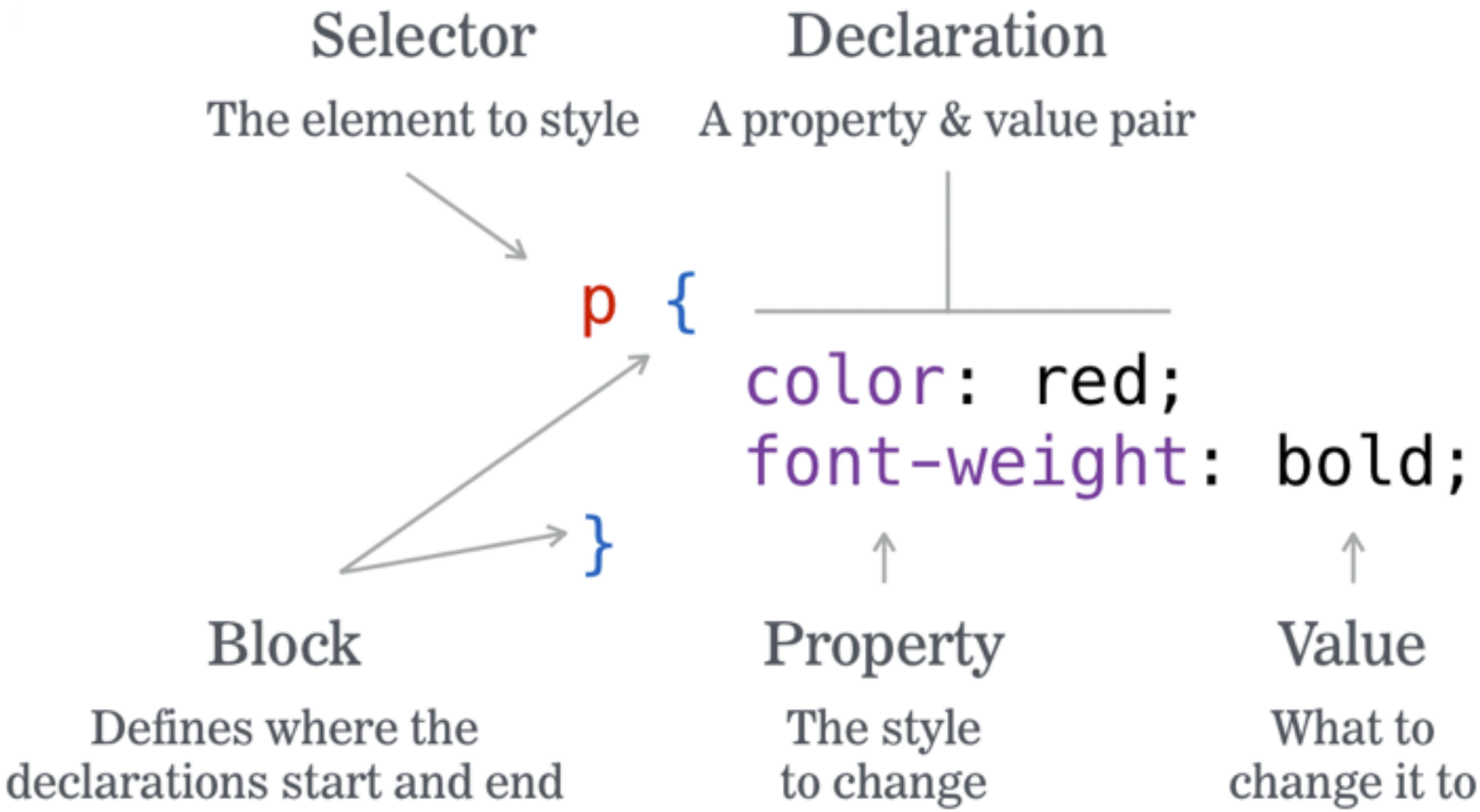
- The HTML document needs to be told of the presence of a CSS file
- A document can reference multiple CSS files
- We use the `link` tag in the `head` element to link the files
- The link tag needs its `rel` attribute set to stylesheet, and its `href` attribute pointing to a CSS file

```
<link rel="stylesheet" href="css/styles.css">
```

ANATOMY

```
p {
    color: red;
    font-weight: bold;
}
```

ANATOMY


Selector
The element to style
Declaration
A property & value pair
p {
color: red;
font-weight: bold;
}
Block
Defines where the declarations start and end
Property
The style to change
Value
What to change it to

CODE ALONG: BASIC CSS

CSS REVIEW

- The C in CSS stands for “cascading”. Rules affect the children of elements they’re applied to
- More specific rules will “win out” over less specific ones. “More specific” rules are longer: `ul li p { ... }` over `p { ... }`
- Making a rule “important” will override specificity. Just **don’t** do this: `li { font-weight: bold !important; }`

CSS EMBEDDING

Name	Usage	Best practice?
External Stylesheet	`<link rel="stylesheet" href="…">`	Right on!
Embedded styles	`<style type="text/css"> ... </style>`	There's no reason to…
Inline styles	`<p style="font-weight: bold;> ... </p>`	Please no.

OTHER COMMON BASIC CSS PROPERTIES

Property	Example usage
font-family	`'Helvetica', Arial, sans-serif;`
margin	`10px auto 20px; (top left/right bottom)`
padding	`20px; (top/right/bottom/left)`
border	`2px solid #000; (thickness style color)`
max-width	`800px; (maximum width)`
border-radius	`5px 5px 5px 5px; (TL TR BR BL)`
text-decoration	`underline; (basic text-decorations)`
transition (bonus!)	`background .2s ease-out; (property duration easing)`

LAB:
HURLEY'S FIRST WEBSITE


WE DID IT!

SO WHAT'S NEXT?

BEST PRACTICES TO ADOPT FROM DAY 1

- Write lean, clean code (DRY, consistent formatting)
- Document your code (comments/README)
- Version Control your code (get on Github!)
- Code for semantics (use tags as they were intended)
- Code for accessibility (can everyone view your site?)
- Do it right the first time (it will save time in the end)

TECHNIQUES & PHILOSOPHIES TO CONSIDER

- Responsive Design

  Design and code your sites to be fluid, adapting to the size of the viewport, not dependent on specific devices or sizes
- Mobile First

  Design and code your sites from the smallest viewport layout first and adapt it as the viewport gets larger, rather than the other way around
- Progressive Enhancement & Gracefule Degradation

  Code your sites to be compatible with the least capable browsers/scenarios first, and layer on complexity if the technology is capable, or code for the most capable technology but ensure less-capable browsers/scenarios are still functional

THE SOFTWARE I USE EVERYDAY AS A DEVELOPER

- Text Editor - Sublime Text
- Browser Code Inspectors (like Chrome DevTools)
- FTP - Filezilla & Transmit
- SQL - Sequel Pro
- Command Line - iTerm2 or Console
- Image Compression - ImageOptim or PNGGauntlet
- Virtual Machines - VirtualBox
- Design - Adobe Creative Suite, Sketch, or alternatives

WHAT'S NEXT?

BOILERPLATES:

- HTML5 Boilerplate
- Base
- Start Here (my own custom)
- Kraken
- Fireshell

FRAMEWORKS:

- Bootstrap
- Foundation
- jQuery
- Compass (Sass)
- UIkit

A BOILERPLATE IS A STARTING POINT FOR PROJECTS, USUALLY FOCUSED ON PROJECT STRUCTURE, TOOLS, AND MARKUP, BUT DOESN'T ASSUME STYLES

A FRAMEWORK ESTABLISHES A LIBRARY OF REUSABLE MODULES, STYLES, AND FUNCTIONS (AND SOMETIMES THE SAME THINGS AS A BOILERPLATE TOO)

TOOLS & TECH TO IMPROVE WORKFLOW, SPEED, AND EXTEND FUNCTIONALITY

- CSS Pre-processors - Sass or Less
- JavaScript Task Runners - Grunt.js or Gulp.js
- GUI Task Runners - CodeKit or Prepros
- Code Shortcuts - Emmet
- Open Source it - Github
- Live Code Injection - LiveReload
- Shortcuts and Workflows - Alfred App
- Window organization - Divvy

WHERE CAN I LEARN MORE?

WHERE CAN I LEARN MORE?

ONLINE RESOURCES:

- MDN
- CSS Tricks
- Codecademy
- Fred's Backpack
- Dash (General Assembly)
- Shay Howe's Learn to Code HTML & CSS
- CodePen

IN-PERSON CLASSES & MORE:

- General Assembly
- ChiHTML5 Meetup
- CodePen Chicago Meetup
- Refresh Chicago Meetup

UPCOMING GA CLASSES

WEB DEVELOPMENT IMMERSIVE

- Full-time, 12 weeks, career-prep
- Starts October 26th

This intensive full-stack program is designed for people looking to completely change careers into being web developers, and we provide rigorous job prep and networking support. Over 90% of grads from this program get employed in web dev within 3 months of graduating.

FRONT-END WEB DEVELOPMENT

- Part-time, 10 weeks
- Starts August 17th

Learn to create beautiful, interactive websites with HTML, CSS, and Javascript!

FIND OUT MORE AT HTTPS://GENERALASSEMB.LY/CHICAGO

Q&A TIME

THANKS!

MATT SORIA

- matt.m.soria@gmail.com
- mattsoria.com
- @poopsplat
- codepen.io/poopsplat
- github.com/poopsplat