HTML & CSS

Ben Gavan

July 18, 2020

Contents

1 Terminal Essentials **5**
1.1 Basic Commands . . . 5
1.2 Git . . . 6

2 Key Board Shortcuts **6**

3 HTML Essentials **6**
3.1 File Naming conventions . . . 6
3.2 Folder Naming Conventions . . . 6
3.3 Emmet.io . . . 7
3.4 Tag Attributes . . . 7
3.5 Absolute vs Relative URLs . . . 7

4 Block vs Inline **7**
4.1 Block . . . 7
4.2 Inline . . . 7

5 HTML Tags **8**
5.1 meta . . . 8
5.1.1 meta tag viewport . . . 8
5.2 div . . . 8
5.3 span . . . 8
5.4 a . . . 8
5.5 *q* - Short quote . . . 8
5.6 *blockquote* - long quote . . . 9
5.7 *cite* - citing an author . . . 9
5.8 *pre* . . . 9
5.9 *samp* . . . 9
5.10 *code* . . . 9
5.11 *kbd* . . . 9

6 CSS Attributes **9**
6.0.1 background-size . . . 9
6.0.2 background-repeat . . . 9
6.0.3 display . . . 9
6.0.4 Dimensions . . . 10
6.0.5 text-align . . . 10
6.0.6 Content - Padding - Border - Margin . . . 10
6.0.7 border . . . 10
6.0.8 margin . . . 10
6.0.9 box-sizing . . . 11
6.1 CSS Reset . . . 11

6.1.1 Motivation . . . 11
6.1.2 Aim . . . 11
6.1.3 Requirements . . . 11
6.1.4 Solution . . . 11
6.1.5 Alternatives . . . 11
6.2 *box-shadow* . . . 11
6.3 *vertical-align* . . . 11

7 CSS Colors (Data type) 11
7.1 Picking Good Color Combinations - Adobe Color (Kuler) . . . 11

8 Image Types 12
8.1 Vector . . . 12

9 CSS Selectors 12
9.1 element . . . 12
9.2 class . . . 12
9.3 id . . . 12
9.4 Attributes . . . 12
9.5 Pseudo Class . . . 12
9.5.1 Link . . . 13
9.5.2 Focus . . . 13
9.5.3 n^{th} child . . . 13
9.6 Pseudo Element . . . 14
9.6.1 First-letter . . . 14
9.6.2 First-line . . . 14
9.7 Nested Selectors . . . 15
9.7.1 all-children . . . 15
9.7.2 immediate . . . 15
9.7.3 all-siblings . . . 15
9.7.4 Immediate Siblings . . . 16
9.8 Compound Selectors . . . 16
9.9 CSS Specificity . . . 16
9.9.1 Example . . . 16

10 Formatting Text 17
10.1 Font property . . . 17
10.1.1 Font-family . . . 17
10.1.2 font-size . . . 17
10.1.3 font-weight . . . 18
10.1.4 font-style . . . 18
10.1.5 font-variant . . . 18
10.1.6 line-height . . . 18
10.1.7 letting-spacing . . . 18
10.1.8 word-spacing . . . 19
10.1.9 color . . . 19
10.1.10 Google Fonts . . . 19
10.2 text . . . 19
10.2.1 text-transform . . . 19
10.2.2 text-align . . . 19
10.2.3 text-shadow . . . 19
10.2.4 text-decoration . . . 19
10.2.5 text-indentation . . . 20
10.2.6 max-width . . . 20

11 Font Awesome **20**
11.1 Performance . . . 20
11.2 Downloading as a font to my computer . . . 20

12 SVG **20**
12.1 Path element . . . 20
12.2 viewBox . . . 20
12.3 ‹symbol›& ‹use› . . . 21
12.3.1 Compatibility issues with IE & Edge - JS Solution . . . 21
12.3.2 Compatibility issues with IE & Edge - Messy Efficient HTML Solution . . . 21
12.3.3 Styling using CSS . . . 21
12.4 Stacking SVGs (Symbol with multiple paths) . . . 22
12.5 Stroke % . . . 22

13 Document Structure **22**
13.1 HTML5 Outline Algorithm . . . 22
13.2 Semantic HTML . . . 23
13.3 article . . . 23
13.4 Section . . . 23
13.5 header . . . 24
13.6 nav . . . 24
13.7 main . . . 24
13.8 aside . . . 25
13.9 footer . . . 25
13.10figure and figcaption . . . 25
13.10.1 figure . . . 25
13.10.2 figcaption . . . 26
13.11Address . . . 26

14 Flexbox **26**
14.1 The Thought behind Flexbox . . . 26
14.2 Container Properties . . . 26
14.2.1 display . . . 26
14.2.2 flex-wrap . . . 26
14.2.3 flex-direction . . . 26
14.2.4 flex-flow . . . 27
14.2.5 justify-content . . . 27
14.2.6 align-content . . . 27
14.2.7 align-items . . . 27
14.3 Item Properties . . . 28
14.3.1 align-self . . . 28
14.3.2 order . . . 28
14.3.3 flex-grow . . . 28
14.3.4 flex-shrink . . . 28
14.3.5 flex-basis . . . 28
14.3.6 flex . . . 29

15 Media Queries **29**
15.0.1 Considerations . . . 29
15.0.2 Consists of . . . 29
15.0.3 Use . . . 30
15.0.4 DON'T Use . . . 30
15.0.5 min-width . . . 30
15.0.6 Example . . . 30
15.1 Buzz-words . . . 30

16 Google's Flexbox Design Patterns **31**
16.1 Main take-aways . . . 31
16.2 Useful links . . . 31
16.2.1 Recommended layout . . . 31

17 Position **31**
17.1 Overview . . . 31
17.2 Fixed . . . 31
17.3 Relative . . . 32
17.4 Absolute . . . 32
17.5 Static . . . 32
17.6 Accepted properties . . . 32

18 Float **32**
18.1 Overflow . . . 32
18.2 *clear* . . . 33

19 Background **33**
19.1 background-color . . . 33
19.2 background-image . . . 33
19.3 background-repeat . . . 33
19.4 background-size . . . 33
19.5 background-position . . . 34
19.6 background-origin . . . 34
19.7 background-attachment . . . 34
19.8 background-clip . . . 34

20 HTML Entities **34**
20.1 Non-breaking space ( ) . . . 34

21 CSS Transitions **34**
21.1 Properties . . . 35
21.2 Example . . . 35

22 CSS Animations **35**
22.1 Defining an animation's key frames . . . 36
22.2 animation-fill-mode . . . 36
22.3 Define . . . 36

23 Linear Gradient **36**

24 Audio & Video **36**
24.1 Audio . . . 36
24.2 Video . . . 37

25 Favicons **37**
25.1 Random URL . . . 37

26 Forms **37**
26.1 ‹input› . . . 37
26.2 ‹label› . . . 38

27 Markdown **38**
27.1 Headers . . . 38
27.2 Italics . . . 38
27.3 Bold . . . 38
27.4 Links . . . 38
27.5 Lists . . . 38
27.6 Code Block . . . 38

28 Random Bits **38**
28.1 Avoid random overrides in CSS . . . 39
28.2 Design Tips . . . 39
28.3 Prefixes . . . 39
28.4 Gulp . . . 39
28.5 Grunt . . . 39

Abstract

Acronyms:

- DOM = Document Object Model

Useful reminders:

- Content → Padding → Border → Margin

1 Terminal Essentials

g

1.1 Basic Commands

- pwd: Print working directory
- ls: list
- ls -la : list list all - shows more detail
- clear: clear current terminal
- cat: Prints out the contents of a file into the termial
- mkdir dir-name: makes a new directory with the name as the string supplied after 'mkdir' within the current dirrectory
- rm file-name: remove directory
- rm -rf dir-name: remove directory recursively (until complete)
- touch txt-file-name: creates new blank txt file with with file name specified
- open file-name: opens the file specified

1.2 Git

- git init: creates an emtpty git repo within current dir
- git status: Gives info on what files haven't been added
- git add
  - dir-name: adds only that dir to the staging index
  - . : add everything below current dir that's changed
  - - -all : adds all files below and above the current dir
- git commit
  - -m "commit-message": commits everything in the staging index along with a commit message.
- git remote add origin : connects the local repo to the cloud
- git push -u origin master : : pushes to the master
- git push: pushes to remote git 'origin/master' (on github)

2 Key Board Shortcuts

- 'cmnd + / ' comment current line
- 'p{TEXT_HERE} + tab' creates p tag with text here as text
- 'lorem' creates some lorem text
- 'control + space' when in element "" to bring up options of possible selections
- 'option + click' to select curser on multiple lines

3 HTML Essentials

3.1 File Naming conventions

- all html files end with the extension '.html'
- all css style sheets end with the extension '.css'
- home/default/main page should be called 'index.html'
- style sheet for main page should be called 'main.css'
- only use lower case alphanumeric characters (a-z, 0-9)
- no spaces

3.2 Folder Naming Conventions

- only use lower case alphanumeric characters (a-z, 0-9)
- no spaces
- follow seperation of cencerns
- for folder holding css files, use 'css'

3.3 Emmet.io

- 'cmnd + / ' comment current line
- 'p{TEXT_HERE} + tab' creates p tag with text here as text
- 'lorem' creates some lorem text

3.4 Tag Attributes

To link a CSS Style sheet to html page:

```
<link rel="stylesheet" href="main.css">
```

Add a picture:

```
<img src="puppy.jpg" alt="An image of cute puppy">
```

Link to other page

```
<a href="http://www.google.com" target="_blank">go to google</a>
```

3.5 Absolute vs Relative URLs

Absolute

- Full URL
- Examples:

  '<img src="https://pbs.twimg.com/media/DLUDtXlX0AEiCnG.jpg">'

Relative

- Within one domain, sshows where one resource is relative to another
- Examples:

  pic/anatomy-of-an-html-element.png

  ../pic/anatomy-of-an-html-element.png (goes up one level first by using '../'))

4 Block vs Inline

4.1 Block

- Will take up the width of the parrent (so will stack verticlly)

y

4.2 Inline

- Will take up only the size/space required - so will stack horizontally; hence inline.

5 HTML Tags

5.1 meta

5.1.1 meta tag viewport

Convention rule is to always include this line (in the head) in all HTML document (this is correct)

```
1 |     <meta name="viewport" content="width=device-width, initial-scale=1">
```

History of mobile and Font Boosting:
- In the beginning of mobile devices, websites were rendered the same as the desktop version so you had to zoom in manually to view the content. Someone then said to just make the font size larger for mobile devices - aka Font Boosting - with still the web page being displayed as 980px wide.

The solution sets the view port width equal to the device/screen width. Since there is normally less pixels, everything appears larger.
The *initial-scale=1* sets the mapping from CSS pixels to device independent pixels to 1 to 1.
If set to 2, each CSS pixels will be mapped onto 1 device independent pixels.

TL;DR:
- meta viewport tag controlls the width and scaling of the browser's viewport. - *width=device-width* matches the screen's width in device-independent pixels. - *initial-scale=1* establishes a 1:1 relationship between CSS pixels and device-independent pixels. - Ensure the page is accessible by NOT disabling user scaling.

5.2 div

- "block" level element = takes up the width of the parrent (so will stack verticlly)
- Generic Element
- Non-semantic

5.3 span

- "inline" element (stacked horizontally next to each other)
- Generic Element
- Non-semantic

5.4 a

Attributes

- 'href' urlm to open when clicked

  href="tel:+447471199741" to dial that phone number when clicked.

- 'target' how to do it when clicked

  '_blank' opens url in new tab

5.5 *q* - Short quote

Short quotations that don't require paragraph breaks
If has the cite attribute, it only appears in the source code - user can not see it.

5.6 *blockquote* - long quote

Used for long quotations.
Usually rendered with the block of text indented. has the cite element.

5.7 *cite* - citing an author

represents a reference to a creative work
Must include:

- Title of work
- or a url reference which may be in abbreviated form.

5.8 *pre*

represents preformatted text - usually displayed in mono-space font as it is in the file.
Need to escape ‹ and › characters with &lt and &gt

5.9 *samp*

= output
Identifies sample output from a computer program.

5.10 *code*

= fragment of computer code

5.11 *kbd*

= Input
Represents user input
propduces an inline element displayed in the browsers default mono-space font
kbd is short for keyboard

6 CSS Attributes

6.0.1 background-size

- 'cover': covers all of the background - avoids repeates

6.0.2 background-repeat

- 'no-repeat': prevents repeates

6.0.3 display

- 'inline' - Will take up only the size/space required - so will stack horizontally; hence inline.

  If width is set, it will have no effect (It will only take up the space required (for the content inside it))

  so since 'block' level elements take up the width of a parent, they cannot be nested inside 'inline' elements. (only can have children of 'inline')

- 'block' - Will take up the width of the parent (so will stack vertically)

- 'inline-block' - displays items inline, but leaves enough room for each element
  can set the width and height.
  Acts like a block as well in inline (acts like an inline block).
- 'none' - will not be displayed

6.0.4 Dimensions

- 'width: ....' - sets the width
- 'height: ....' - sets the height

```
div {
  width: 100px;
  height: 100px;
}
```

6.0.5 text-align

Sets how text should be aligned within the element.

6.0.6 Content - Padding - Border - Margin

- Content - the content itself
- Padding - The Amount of padding around the content before the border
- Border - Around the padding of the content
- Margin - The margin/padding around the border before another element can be displayed.

6.0.7 border

-
- "border-radius: ..." - sets the radius of border
  "...%" - sets the radius as a percentage of the width/height

```
border: 1px solid red;
border-width: 1px;
border-style: solid;
border-color: red;
```

6.0.8 margin

- TRBL
- TB RL
- T R B L
- "margin: 0 auto" - no padding on top or bottom, then auto set left/right so that element is on the centre.

6.0.9 box-sizing

- "box-sizing: border-box" - sets the max size of the box/div (up-to the outside of the border) to the set dimensions via width/height. So the border outside will stay as it is, but the content will shrink if padding is added. If the border is width increased - it will increase inwards, shrinking the content. The Element will not grow outwardly (grows inwards)

6.1 CSS Reset

Brings the CSS formatting down to a uniform base line.

6.1.1 Motivation

Each browser has its own base CSS ('user agent style sheet' for chrome). So if we want to have our website uniform across all browsers, we need to apply our own.

6.1.2 Aim

To provide uniformity across browsers.

6.1.3 Requirements

- Light weight
- is applied to all used elements

6.1.4 Solution

Create our own style sheet to override what the browser (agent) applies.

6.1.5 Alternatives

There are some standards, e.g. meyer css reset (oldest), normalize.css (newer), sanitize.css (newest). However, some of these don't bring the formatting down to zero. Instead they provide their own baseline formatting - overriding what the browser's baseline is.

6.2 *box-shadow*

```
box-shadow: <x off-set> <y off-set> <radius> <color>;
```

6.3 *vertical-align*

Sets the vertical alignment.
The container element grows to accommodate this shift.
Positive to up, Negative is down.

7 CSS Colors (Data type)

Use Hax or rgba

7.1 Picking Good Color Combinations - Adobe Color (Kuler)

- Explore for popular color schemes
- Color wheel for discovering new schemes
- Can import an image and gives colors that matches

Use iStock photo for copyright free images.

8 Image Types

Raster/Bitmap & Vector

8.1 Vector

File format: - SVG

9 CSS Selectors

Different ways to select HTML elements.

9.1 element

Formats that HTML element.

9.2 class

Adds the 'class' attribute' to a HTML element.
Can add them wherever and as many times as we want.
A period before the selector is used to denote that the selector is for a class. ".example {}"

9.3 id

- Can only be used once.
- Notation to denote id selector is "#example {}"

```
<p id="example"></p>
```

```
#example {
}
```

9.4 Attributes

- Only applies that selector to HTML tags which have that attribute

```
<p example="something"></p>
```

```
[example] {
}
```

- Only applies that selector to HTML tags that have the exact attribute and set to that specific value

```
<p example="something"></p>
```

```
[example=something] {
}
```

9.5 Pseudo Class

Denoted by ":" (single :)

9.5.1 Link

```
/\* order matters */
/\* LVHA */
/\* Link Visited Hover Active */
a:link {
  color: green;
}

a:visited {
  color: blue;
}

a:hover {
  color: red;
}

a:active {
  color: yellow;
}
```

```
div:hover {
  background-color: green;
  cursor: pointer;
}
```

Use 'active' for links and focus for other elements such as forms.

9.5.2 Focus

Used for other elements such as forms.

9.5.3 n^{th} child

Used to select and format the nth child of a parent in HTML.
First-Child:

```
li:first-child {
}
```

Last Child:

```
li:last-child {
}
```

To select every even child of the parent:

```
li:nth-child(even) {
}
```

To select every odd child of the parent:

```
li:nth-child(odd) {
}
```

Zebra-striping:

```
table, tr, td {
  width: 100%;
  border: 1px solid black;
}

tr:nth-child(even) {
  background-color: rgba(128, 128, 128, 0.49);
```

```
}

tr:nth-child(odd) {
  background-color: rgba(128, 128, 128, 0.19)
}
```

To select a specific child:

```
li:nth-child(3) {
}
```

To select the nth child from the bottom:

```
li:nth-last-child(10) {
}
```

To Select more complex pattern, use a linear line (mx+c)

```
li:nth-child(3n+2) {
  color: red;
}

/\*
Try changing the selector to select each of the following:
8, 18, 28, 38, ... 10n+8
9, 12, 15, ..., 39, ... 3n+9
3, 12, 21, 30, 39, ... 9n+3
\*/
```

To Select the a child of a parent where it is the only child:
(To Select the a HTML element of a parent where the element is the only child:)

```
article:only-child {
}
```

9.6 Pseudo Element

Denoted by "::" (Double :)
Typography:

9.6.1 First-letter

Selects the first letter

```
p::first-letter {
  font-family: cursive;
  font-size: 36px;
  line-height: .5;
}
```

9.6.2 First-line

Dynamically selects the first line (dynamically adapts/reforms depending on width of view port)

```
p::first-line {
  color: red;
  font-weight: 900;
}
```

9.7 Nested Selectors

9.7.1 all-children

Selects all children of a parent = "parent child {}". No matter how nested

```
parent child {
}
```

To select all children of a 'div' which are 'p' elements

```
div p {
}
```

will select all 'p' tags in the following case

```
<!--div>p*2+section>p^^p-->
<div>
  <p>first p</p>   <- selected
  <p>second p</p>  <- selected
  <section>
    <p>third p</p> <- selected
  </section>
</div>
<p>fourth p</p>
<p>fifth p</p>
<p>sixth p</p>
```

9.7.2 immediate

To select all the immediate children of a 'div' which are 'p' elements

```
div > p {
}
```

will select all 'p' tags except the ones that are not immediate descendants of a 'div' so only 'first p' and 'second p' will be selected in the following case

```
<!--div>p*2+section>p^^p-->
<div>
  <p>first p</p>      <- selected
  <p>second p</p>     <- selected
  <section>
    <p>third p</p>
  </section>
</div>
<p>fourth p</p>
<p>fifth p</p>
<p>sixth p</p>
```

9.7.3 all-siblings

All 'p' tags that are siblings of a 'div'

```
div ~ p {
}
```

will select ..... 'p' tags in the following case

```
<!--div>p*2+section>p^^p-->
<div>
  <p>first p</p>
  <p>second p</p>
```

```
  <section>
    <p>third p</p>
  </section>
</div>
<p>fourth p</p> <- selected
<p>fifth p</p>  <- selected
<p>sixth p</p>  <- selected
```

9.7.4 Immediate Siblings

will select the tag that is immediately after the tag - only the immediate sibling.

```
div + p {
}
```

will select only the 'fourth p' 'p' tag in the following case

```
<!--div>p*2+section>p^^p-->
<div>
  <p>first p</p>
  <p>second p</p>
  <section>
    <p>third p</p>
  </section>
</div>
<p>fourth p</p> <- selected
<p>fifth p</p>
<p>sixth p</p>
```

9.8 Compound Selectors

```
ul#some-id li.some-class {
}
```

Will select an 'li' with the class 'some-class' which is a child of a 'ul' with an id 'some-id'.

9.9 CSS Specificity

- Element selector = 1
- Class = 10
- ID = 100
- Inline = 1000

If same specificity, go to the order that rule sets are declared in the CSS. Last = More powerful.

Selector Type	Value	Place
Element	1	0-0-1
Class/attribute	10	0-1-0
ID	100	1-0-0
Inline	1000	1-0-0-0

9.9.1 Example

```
ul#some-id li.some-class {
}
```

The first section

```
ul#some-id
```

has a specificity of 101 (1 from ul (element selector) 100 from '#some-id' (id selector))
The Second section

```
li.some-class
```

has a specificity of 11
So the total specificity is 112.

10 Formatting Text

10.1 Font property

A short-hand property.
Best not to use the short hand property version when starting off. Ensure you understand the individual properties, then use the short hand property.

10.1.1 Font-family

- set specific fonts first, but may fail if they are not installed or cannot be installed. So best to state a generic font family, such as sans-serif, at the end to select any font that the person has of that family. This is called fallback.

Sans-serif is as the best font to use on the web. (the de-facto standard). It's the most readable on the web.

10.1.2 font-size

Default is usually set to 16px.
Font size values:

- xx-small, x-small, small, medium, large, x-large, xx-large

  relative to default font size (root font size).

  default font size is usually medium.

- larger, smaller

  relative to the parent's font size.

- rem

  relative to default font size (root font size).

  will scale fonts proportionally to the default font size (usually 16px but normally adjusted for accessibility issues).

  originates from how wide the letter 'm' is on a printing press.

- vw - viewport width

  1/100th of the width of the viewport.

  Useful if you want to have a banner across the page

  Will scale dynamically as the width of the page changes.

- vh - viewport height

  1/100th of the height of the viewport.

10.1.3 font-weight

Can take the values:

- normal
  same as 400
- bold
  same as 700
- lighter
- bolder
- 100, 200, 300, 400, 500, 600, 700, 800, 900

10.1.4 font-style

10.1.5 font-variant

Is a sort hand for five longhand font related properties:

- font-variant-caps
- font-variant-numeric
- font-variant-alternates
- font-variant-ligatures
- font-variant-east-asian

Values:

- 'small-caps' - kind of looks good

10.1.6 line-height

Letting
Adjusts the vertical distance between each line box. Each line of text is contained within a box. This property adjusts the vertical distance between each of these line boxes.

Can be used to centre things, however, it is more of a hack method. This is completed by setting the line-height to the same value of the height of the div that the text is a child of.

10.1.7 letting-spacing

Kerning
Can take the values:

- normal
- <length> CSS Data type
  Should really just stick to 'rem'. Can use 'px' if you really want to.

10.1.8 word-spacing

Can take the values:

- normal
- <length> CSS Data type

  Should really just stick to 'rem'. Can use 'px' if you really want to.
- percentage (not a CSS length data type)

10.1.9 color

Is a CSS Data type. Can either be a keyword with a hexadecimal number associated with it. Or a pure hexadecimal number can be used - a RGB value (RGBA). Each of the red, green, and blue (and alpha) values are represented by two digits (up to 255).

10.1.10 Google Fonts

Recommended to choose/sort by popularity so that there will be a high chance that that font is already been downloaded onto the user's/client's machine. This will reduce the data needed to be transferred, so will speed up the web page load time.
Should monitor the weight (the size of the files that you are sending/using)

10.2 text

Now we have got the font, let us now form the text. Such as indenting, underlining, and applying shadow effects.

10.2.1 text-transform

- 'uppercase' - sets all selected text to uppercase.
- 'capitalise' - sets the first letter of each word to a capital/uppercase.

10.2.2 text-align

Describes how content like text is aligned in its parent block element.
Does not control the alignment of block elements, only their inline content.

10.2.3 text-shadow

- accepts a comma-separated list: x, y, blur radius, color

  offset-x | offset-y | blur-radius | color

  text-shadow: 1px 1px 2px black;

10.2.4 text-decoration

text-decoration-line Adds/removes underline, overline, and strike through.
Therefore, is commonly used to remove the underline from hyperlinks (anchor tags).
Takes the values which any number of them can be used at once:

- none
- underline
- overline

- line-through

But use text-decoration instead - does not work otherwise??????????

Do NOT use '<s>' tag in HTML, use the text-decoration-line element in CSS to achieve the same effect.

Do NOT use '<del>' tag in HTML. Used to be used to denote that that text had been deleted. If you want something deleted, just delete it from the page. Or if you still want to have the same effect use the CSS property, 'text-decoration-line'.

10.2.5 text-indentation

Sets how indented e.g. the first line is of a 'p' tag.
recommended to use 'rem' length units.

- <length> CSS data type
- percentage

The text-indent property specifies the amount of indentation (empty space) should be left before lines of text in a block. By default, this controls the indentation of only the first formatted line of the block, but the hanging and each-line keywords can be used to change this behaviour

10.2.6 max-width

Sets the maximum width an element can be. Enables it to be smaller than that, so can be useful for adaptations for smaller screens.

11 Font Awesome

https://fontawesome.com/cheatsheet

Full open source under MIT license.

11.1 Performance

To only send the icons you are needing, use the SVG of them.

11.2 Downloading as a font to my computer

12 SVG

Ideally start with uniform dimensions, such as 100px by 100px.
Can use InkScape.
Leave space around the element on the art board/canvas so say if the stroke is increased, the SVG will not be clipped.

12.1 Path element

```
<path d="M10 10 l90 140 V30 h20 m20 10 l90 140" stroke="black" stroke-width=
    "4" fill="transparent"/>
```

12.2 viewBox

For example,

```
    <symbol id="play-circle" viewBox="0 0 100 100">
```

It is analogous to the canvas size (in pixels) when the SVG is first created. So by defining the viewBox on in XML it allows the following coordinated used in the definition of that element to be keyed relative to that view box.

```
viewBox="<x-zero> <y-zero> <x-dimension> <y-dimension>"
```

It also can act as a scaling mechanism. For example, if width=200 and height=250 with a view box set as:

```
<svg version="1.1"
  xmlns="http://www.w3.org/2000/svg"
  width="200"
  height="250"
  viewBox="0 0 400 500">
```

each 2 pixels will be mapped onto 1 display-independent pixel. So will be half the dimensions (quarter of size 2^2)

12.3 ‹symbol› & ‹use›

Used to define graphical template objects which can be instantiated by a use element.

By using the decleration of

```
<svg version="1.1"
  xmlns="http://www.w3.org/2000/svg">

  <symbol id="play-circle" viewBox="0 0 100 100">
    <path d="M83.4,50.9c0,18.5-15,33.4-33.4,33.4s-33.4-15-33.4-33.4c0-18.5
        ,15-33.4,33.4-33.4S83.4,32.4,83.4,50.9z M68.1,50.9
    c0-1-0.5-1.9-1.4-2.4L43,34.5c-0.8-0.5-1.9-0.5-2.8,0c-0.9,0.5-1.4,1.4-1.4
        ,2.4v27.9c0,1,0.5,1.9,1.4,2.4c0.4,0.2,0.9,0.3,1.4,0.3
    s1-0.1,1.4-0.4l23.7-13.9C67.6,52.8,68.1,51.9,68.1,50.9z"/>
  </symbol>
</svg>
```

which can then be referenced in any number of html files by using

```
<svg><use xlink:href="icons.svg#play-circle"></use></svg>
```

12.3.1 Compatibility issues with IE & Edge - JS Solution

To make this work on Edge (and possibly other browsers) use the js file on git hub (/044_svg/10_symbol/svg4everybody.js) and then run at the bottom of the HTML page using

```
<script src="svg4everybody.js"></script>
<script>svg4everybody();</script>
```

12.3.2 Compatibility issues with IE & Edge - Messy Efficient HTML Solution

Just declare the SVG at the top of the HTML page with *display="none"* and then reference as normal.

Could set the server up so that it injects the SVG declaration at the top of the page when it builds the page so you don't need to look at it.

12.3.3 Styling using CSS

Cannot be done if you put it onto the page using an image tag.

12.4 Stacking SVGs (Symbol with multiple paths)

```
<svg version="1.1"
  xmlns="http://www.w3.org/2000/svg">

  <symbol id="check" viewBox="0 0 100 100">
    <path d="M72.2,43.2L50.7,63.5l-4,3.8c-0.5,0.5-1.3,0.8-2,0.8c-0.7,0
        -1.5-0.3-2-0.8l-4-3.8L27.8,53.4c-0.5-0.5-0.8-1.2-0.8-1.9
    s0.3-1.4,0.8-1.9l4-3.8c0.5-0.5,1.3-0.8,2-0.8c0.7,0,1.5,0.3,2,0.8l8.7,8.3
        l19.5-18.5c0.5-0.5,1.3-0.8,2-0.8s1.5,0.3,2,0.8l4,3.8
    c0.5,0.5,0.8,1.2,0.8,1.9C73,42,72.7,42.7,72.2,43.2z"/>
  </symbol>


  <symbol id="circle-thin" viewBox="0 0 100 100">
    <path d="M50,84.4c-18.5,0-33.5-15-33.5-33.5s15-33.5,33.5-33.5c18.5
        ,0,33.5,15,33.5,33.5S68.5,84.4,50,84.4z M22.1,50.9
    c0,15.4,12.5,27.9,27.9,27.9c15.4,0,27.9-12.5,27.9-27.9S65.4,22.9,50,22.9
        C34.6,22.9,22.1,35.5,22.1,50.9z"/>
  </symbol>

  <symbol id="checked" viewBox="0 0 100 100">
    <!-- the check -->
    <path fill="currentcolor" d="M72.2,43.2L50.7,63.5l-4,3.8c-0.5,0.5-1.3
        ,0.8-2,0.8c-0.7,0-1.5-0.3-2-0.8l-4-3.8L27.8,53.4
        c-0.5-0.5-0.8-1.2-0.8-1.9
    s0.3-1.4,0.8-1.9l4-3.8c0.5-0.5,1.3-0.8,2-0.8c0.7,0,1.5,0.3,2,0.8l8.7,8.3
        l19.5-18.5c0.5-0.5,1.3-0.8,2-0.8s1.5,0.3,2,0.8l4,3.8
    c0.5,0.5,0.8,1.2,0.8,1.9C73,42,72.7,42.7,72.2,43.2z"/>
    <!-- the circle -->
    <path d="M50,84.4c-18.5,0-33.5-15-33.5-33.5s15-33.5,33.5-33.5c18.5
        ,0,33.5,15,33.5,33.5S68.5,84.4,50,84.4z M22.1,50.9
    c0,15.4,12.5,27.9,27.9,27.9c15.4,0,27.9-12.5,27.9-27.9S65.4,22.9,50,22.9
        C34.6,22.9,22.1,35.5,22.1,50.9z"/>
  </symbol>

</svg>
```

12.5 Stroke %

13 Document Structure

Importance of Good document structure:

- maintainability

Important terms:

- Document Object Model (DOM)
- Document Flow

  Since every element is a box, this is how these boxes naturally stack/flow.

  So Block level elements will stack taking up the entire width/row from top to bottom.

  So inline elements will stack next to each other.

13.1 HTML5 Outline Algorithm

DO NOT use
Even though it is in the specification, no one has implemented it since it was too complicated.
However, do outline your documents using the heading tags (h1 through h6).

13.2 Semantic HTML

Non-semantic:

- div
- span

Semantic:

- header
- nav
- main
- article
- section
- aside
- footer
- h1 - h6
- figure
- figcaption
- address

13.3 article

Needs to be self contained.
Should be able to cut it out and it would still make sense.
On the other hand, each part????? of the article will not stand on its own - needs the rest of the article to make sense.
An article has sections.

13.4 Section

Like an article but does not need to be able to stand on its own.
See article for good example on MDN

```
<article class="film_review">
  <header>
    <h2>Jurassic Park</h2>
  </header>
  <section class="main_review">
    <p>Dinos were great!</p>
  </section>
  <section class="user_reviews">
    <article class="user_review">
      <p>Way too scary for me.</p>
      <footer>
        <p>
          Posted on
          <time datetime="2015-05-16 19:00">May 16</time>
          by Lisa.
        </p>
      </footer>
```

```
    </article>
    <article class="user_review">
      <p>I agree, dinos are my favorite.</p>
      <footer>
        <p>
          Posted on
          <time datetime="2015-05-17 19:00">May 17</time>
          by Tom.
        </p>
      </footer>
    </article>
  </section>
  <footer>
    <p>
      Posted on
      <time datetime="2015-05-15 19:00">May 15</time>
      by Staff.
    </p>
  </footer>
</article>
```

13.5 header

- a group of "introductory content" or "navigational aids"

  may contain some heading elements but also other elements like a logo

- commonly has a heading (hn tag)
- must not be a descendent of

  header

  footer

  address

13.6 nav

Is a section with navigation links

- not all links within a document must be in a nav element
- It is intended for the major block of links

  such as the main navigation of the page/website

  Helps to be categorized by search engines

- "it is common for footers to have a list of links to various key parts of a site, but the footer element is more appropriate in such cases, and no nav element is necessary for those links." (src: W3)
- a document may have several nav elements, for example, one for site navigation and one for intra-page navigation.

13.7 main

the main content of the <body> of a document or application.
Should be unique to that page.
There should only be one main tag per page.

13.8 aside

Content related to something.

Description:
The HTML <aside> element represents a section of the page with content connected tangentially to the rest, which could be considered separate from that content. These sections are often represented as sidebars or inserts. They often contain the definitions on the sidebars, such as definitions from the glossary; there may also be other types of information, such as related advertisements; the biography of the author; web applications; profile information or related links on the blog. (MDN)

Examples:

- sidebars
- inserts
- advertisements
- biography of the author
- related links on the blog

13.9 footer

Description:
The HTML <footer> element represents a footer for its nearest sectioning content or sectioning root element. A footer typically contains information about the author of the section, copyright data or links to related documents.

Usage notes:

- Enclose information about the author in an <address> element that can be included into the <footer> element

Examples:

- information about the author of the section
- copyright data
- links to related documents

13.10 figure and figcaption

13.10.1 figure

Self-contained content, frequently with a <figcaption>
While it is related to the main flow, its position is independent of the main flow; it could be moved to another page or an appendix without affecting the main flow.
Typically referenced as a single unit
Examples:

- image
- illustration
- diagram
- code snippet

13.10.2 figcaption

Used as the caption within a figure.
It can only be the first or last element in the <figure> block.
It is optional. A <figure> does not a caption. However, a <figcaption> requires a <figure>.

13.11 Address

Wraps the contact information of that element.
Includes at the end of a body

14 Flexbox

Search CSS tricks flexbox for the best resource - https://css-tricks.com/snippets/css/a-guide-to-flexbox/
The standard currently, with 94% compatibility.
There is a flex container. The immediate children of this element, are called flex items.

Check out: https://flexboxfroggy.com/

14.1 The Thought behind Flexbox

You should think in terms of containers to layout the page. Once the page has been separated into containers, you can start to make containers 'flex containers' as necessary, and arrange the children as 'flex items'.
This opens another whole world of layout opportunities.

Flex items can only be immediate children of the flex container.

14.2 Container Properties

14.2.1 display

- flex
- inline-flex

14.2.2 flex-wrap

- Nowrap (default)
- Wrap
- Wrap-reverse

14.2.3 flex-direction

Allows us to dictate which direction the flex items are laid out.
Possible values:

- Row (default)
- Row-reverse
- Column

- Column-reverse

Primary Axis:
- The axis which is parallel to flex-direction's value.
- Whatever your flex-direction is set to, that is your primary axis.
Cross axis:
- The axis which is perpendicular to the primary axis

14.2.4 flex-flow

Is a short-hand property for 'flex-direction' and 'flex-wrap'.

14.2.5 justify-content

'justify-content' moves/aligns items along the main/primary axis (as discussed in 'flex-direction').
Possible values:

- Flex-start
- Flex-end
- Center
- Space-between
- Space-around

14.2.6 align-content

Aligns content in the cross axis when there is space to do so.
Shifts all of the content items together (not with respect to each row (along the primary/main axis))
Shifts the lines
e.g. flex-start = will shift all lines packed at the top of the container.

```
1 .container {
2   align-content: flex-start | flex-end | center | space-between |
        space-around | stretch;
3 }
```

14.2.7 align-items

Moves the items along the cross axis according to the possible values:

- Flex-start
- Flex-end
- Center
- Stretch
- Baseline

Moves the items relative to their row (remains inside their row) whereas align-content aligns all of the content/items - also moves the lines/columns.
align-content moves the entire group/line.

```
1 .container {
2   align-items: stretch | flex-start | flex-end | center | baseline;
3 }
```

14.3 Item Properties

Flex properties that can be applied to individual items inside of the flex container.
CSS properties of the actual item.

14.3.1 align-self

Applies align properties solely on individual flex items, irrelevant to the alignment set by/on the flex container.

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```

14.3.2 order

States what order that item will be displayed.
If it does not have an order value, it will be displayed after the items that do.
Items with the 'order' property set, will be displayed first.
Can use negatives.
Items can have the same order value. But will be ordered along with the other ordered items, but ones with the same will be displayed in the order that they are declared in. (But will be ordered in the order that they are declared in the HTML source code).

```
.item {
  order: <integer>; (default is 0)
}
```

14.3.3 flex-grow

A value given to a flex item that decides what fraction of space it should occupy (Similar to how potential difference is split according resistance).
Default value 0 - should will not grow to fill the space.
Negative values are not valid.

```
.item {
  flex-grow: <number>;  (default 0)
}
```

14.3.4 flex-shrink

A value which dictates/enables how the item shrinks when required.
Value is the fraction of the shrinkage that that item will experience.
So 3 items, 2 with value 1 and 1 with value 2 - The value 2 will receive $\frac{1}{2}$ of the shrinkage and the other 2 will receive $\frac{1}{4}$. So value 2 will shrink twice as fast as the other two.

0 if it doesn't shrink,
not 0 if it does shrink and by how much compared to the other items.

```
.item {
  flex-shrink: <number>; (default 1)
}
```

14.3.5 flex-basis

The value that 'flex-shrink' and 'flex-grow' kicks in.
If set at a percentage, since the basis will always change along with the value of the parent, shrink and grow will never kick in.

14.3.6 flex

A short-hand property for:

- flex-basis
- flex-shrink
- flex-grow

with the second and third parameters optional.

```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

Recommended to use instead of individual properties.
Will set the other values, that are not set, intelligently.

15 Media Queries

Allows us to apply different CSS depending on the viewport size. e.g. for phone, tablet, desktop...

15.0.1 Considerations

- Best to use separate files for different media queries.
- Start by developing for mobile size first to keep it as light weight as possible. Then we can scale up to desktop and add more to the page (will take up more data) aka - Mobile First Design.
The majority of users are on mobile.

15.0.2 Consists of

Media Type:

- optional
- can only have one value / can't try to have multiple media declarations = although can use 'all'
- can be:
    All
    screen
    print - how things appear when it is printed out.
    Speech

  Media Feature / Expression:

- can have as many as required.
- Default value = 'screen'
- Values excepted
    min-width
    max-width
    and more...

```
<!-- this doesn't work - you can't have multiple media types -->
<link rel="stylesheet" href="multiple.css" media="screen and print" >

<!-- this does work -->
<link rel="stylesheet" href="multiple.css" media="all">
```

15.0.3 Use

- min-width
- max-width
- min-height
- max-height

15.0.4 DON'T Use

The following can have some issues - DON'T use them! Everything can be achieved with the parameters above.

- min-device-width
- max-device-width
- min-device-height
- max-device-height

15.0.5 min-width

Apply this CSS from this width onwards.

```
<link rel="stylesheet" href="mq_900-plus.css" media="(min-width: 900px)">
```

surround parameters with parenthesis.

15.0.6 Example

If we want the navigation horizontally along the top of the page, set that element to a width of 100% with the flex container being 'row wrap'.
When the page width increases and we want the navigation to be on the left-hand side, use a min-width media query so that above a certain fresh-hold, this width of the items will change. For example, 15% and 85% so they will appear in a row next to each other, occupying one full row across the page.

15.1 Buzz-words

Responsive Design
- Your web page responds to the size of the viewport

Mobile First
- Build your pages to look good on mobile first
- Then add in break-points so that your pages work at larger resolutions

Breakpoints based upon content
- Create your breakpoints based upon your content
- Have your content look good, and be readable, at different sizes

70 - 80 characters of text per line

- Classic readability theory suggests that an ideal column should contain 70 to 80 characters per line (about 8 to 10 words in English)
- When the width of a text block grows past about 10 words, a breakpoint should be considered

16 Google's Flexbox Design Patterns

16.1 Main take-aways

- Build mobile first
- Use:

  display: flex;

  flex-flow: row wrap;
- Each row:

  What you want on each row adds up to 100

16.2 Useful links

- https://developers.google.com/web/fundamentals/design-and-ux/responsive/patterns

16.2.1 Recommended layout

A few 'div's with the central one being a flex container. Divs will span the full width and can adjust the width ratios of the elements inside the flex container at break points.

17 Position

17.1 Overview

The position CSS property is good for positioning elements. A positioned element is an element whose computed position property is either relative, absolute, fixed or sticky.

17.2 Fixed

No space reserved for the element - it is taken out of the document flow (i.e. not block level where it takes up the entire row, nor it is an inline element where it takes up as little horizontal space as possible and stack with other elements along the same row)

A fixed position element is positioned relative to the view-port (at a fixed position relative to the view-port), or the browser window itself.

Since the element is positioned relative to the view-port, and the view-port does not change when the page is scrolled, the element will remain in the same position/stay where it is.

This is useful/can be used for:

- Menus
- Navigation
- Cookie alerts...

17.3 Relative

Positioned **relative** to itself.

The position is computed relative to where it should be (to itself). Where the element should be is still blocked out in the DOM.

17.4 Absolute

An *absolute* positioned element is removed from the DOM and has it's position computed from relative to the next element up in the DOM (tree) which also has the position property set. If there are non, the element is positioned in the normal document flow.

17.5 Static

Normal behavior - positioned within the document flow as per usual.

17.6 Accepted properties

- Top
  - ‹length›
  - percentage
- Right
  - ‹length›
  - percentage
- Bottom
  - ‹length›
  - percentage
- Left
  - ‹length›
  - percentage

18 Float

Old approach - use Flexbox instead. (Its a very hack-y way of positioning things on the page)
The float CSS property takes the element out of the normal DOM and aligns it to the left or right relative to its **container**.
Other text or inline elements then position and wrap around this object.
(Needs to be added in HTML before the elements that wrap around it are in the HTML)

18.1 Overflow

With overflow not set, the element's container does not take into consideration that element when it comes to sizing - it will only use the other elements which does not have the *float* property set for calculating it's size.
So if the element with the float set is too large, it will not fit inside it's container (since it has been taken out of the DOM).

The solution to this it to set the *overflow* CSS property on the container element.

- auto
  - will set it's size to accommodate/fit all of the child elements, with float set, inside it.

18.2 *clear*

When the *clear* CSS property is set to an element, that element will ensure that it respects where elements with *float* set are - i.e. the element will not be positioned under an element with *float*.
clear has the following possible values:

- both

  will not be positioned under elements with *float set*

19 Background

background is a CSS shorthand property that is not recommended to be used - just causes confusion (little uniformity of implementation). (Be clear, not clever).
It also sets other default values to other unset properties (relating to background) which could alter the expected outcome.

19.1 background-color

Sets the background color of the element.

19.2 background-image

Sets an image as the background of an element.

19.3 background-repeat

Refers to background-image property to define how the *background-image* is repeated.
Has the possible values:

- Row

  Repeats the background image along the x-axis

- Column

  Repeats the background image along the column axis only

- Both

  Repeats the background image along the x **and** y axis.

19.4 background-size

The *background-size* specifies the size of the *background-image* if set.
Does **NOT** set the size of the background since this size is determined by the size of the element.
Can take the values:

- length
- percentage
- auto
- contain

  sized so this it is *contained* within the element it is the background of

  will scale with the source ratio maintained.

- cover

  The background image will be sized so that *covers* the element it is the background of.

  Will scale so that the source ratio maintained.

19.5 background-position

Refers to the background image.
background-position is a CSS property that sets the position of the background image relative to the element it is the background of.

19.6 background-origin

Refers to the background image.
background-origin is a CSS property that determines where the background image starts and ends.
Can take the values of:

- *content-box*

  Sets the start and end to align with the content-box of the element.

- *padding-box*
- *border-box*

19.7 background-attachment

Refers to the background-image.
Can take the values of:

- scroll (default)

  the background image scrolls with the page/element

- fixed

  Stays in the same place in the viewport

  the background image does not scroll so stays fixed in the background as the remaining page elements scrolls over it.

19.8 background-clip

Extends to both background-image and background-color.

20 HTML Entities

To use stuff like <, use HTML entities:

```
1 &#60
2 &lt
```

20.1 Non-breaking space ( )

Use

```
1  
```

to force the text not to be put onto more than one line at that space.

21 CSS Transitions

The quick & dirty version of *animations* - only need to specify the start and end point.

21.1 Properties

- *transition-property:*

  ‹property›; (e.g. color)

  all; (applies the transition to possibly all properties)

- *transition-duration:*

  500ms;

- *transition-timing-function:*

  ease-in-out;

- *transition-delay:*

  1s;

21.2 Example

A basic transition can be applied to any element to specify how CSS properties are to be animated when the state (e.g. *:hover*) changes.

```
1  div {
2    width: 30vh;
3    height: 30vh;
4
5    display: flex;
6    justify-content: center;
7    align-items: center;
8
9    background-color: blue;
10   border: 4px solid black;
11   font-size: 4vh;
12   color: yellow;
13 }
14
15 div:hover {
16   color: blue;
17   background-color: yellow;
18 }
19
20 .trans {
21   transition-property: all;
22   transition-duration: 0.5s;
23   transition-timing-function: ease;
24 }
```

22 CSS Animations

To create an animation, you need to specify:

- the CSS properties at "key frames".

  @keyframes <animation-name> {... 10% {...} ....}

- duration

  animation-duration: <seconds>s

- delay

  animation-delay: <seconds>s

- fill mode

  animation-fill-mode: <builtin>

- timing function

  animation-timing-function: <builtin>

- animation name

  animation-name: <name>

22.1 Defining an animation‘s key frames

```
@keyframes grow {
  0% {
    font-size: 0;
  }
  100% {
    font-size: 40px;
  }
}
```

22.2 animation-fill-mode

Defines what properties specified during the animation are permanently applied to the element being animated (after the animation has finished).
Has the possible values:

- *none*

  The animation will not permanently apply any styles after the animation has finished.

  Default value

- *forwards*

  The element will retain the styles set by the last keyframe of the animation

- *backwards*

  The element will retain the styles set by the first keyframe of the animation

- *both*

  Follows the rules for both forwards and backwards.

22.3 Define

23 Linear Gradient

```
.gradient {
  background-image: linear-gradient(#72c41f, #5c9e19);
  background-image: linear-gradient(hotpink, green, deepskyblue);
}
```

24 Audio & Video

24.1 Audio

```
<audio controls>
  <source src="../00_files/Newish_Disco.mp3" type="audio/mpeg">
</audio>
```

24.2 Video

The browser will try to play each source in order and only plays it if the format is supported.

```
<video controls>
  <source id="s3" src="../00_files/me.webm" type="video/webm">
  <source id="s2" src="../00_files/me.ogv" type="video/ogg">
  <source id="s1" src="../00_files/me.mp4" type="video/mp4">
</video>
```

25 Favicons

25.1 Random URL

Add a random string to the end of the URLto make the browser request the new resource (would just retrieve it from cache)

```
href="favicon/apple-touch-icon.png?v=eEavRGEXR9"
```

26 Forms

```
<input type="text" id="fname" name="first-name">
```

- 'id' is what is used on the website, e.g. for targeting that specific element with CSS or JavaScript.
- 'name' is the name that is given to the variable that contains the data from that element which is sent off to the server.

```
<form action="/where/its/beging/sent" method="post">
  <!--GET-->
  <!--URL-->

  <!--POST-->
  <!--BODY-->
</form>
```

26.1 ‹input›

```
<form action="/" method="post">
  <input type="text" id="fname" name="firstName" title="enter your name"
      placeholder="first name please" autofocus autocomplete="off">
</form>
```

- *id* = CSS (used to target the element)
- *name* = the name the input value is sent to the server with (name: data)
- *value* = preset input data (can be overridden)

26.2 ‹label›

```
<form action="/" method="post">

  <label for="fname">First Name:</label>
  <input type="text" id="fname" name="firstName" title="enter your name"
     placeholder="first name please" autofocus autocomplete="off">

</form>
```

- *for* = the *id* of the input element the label is for.

27 Markdown

Markdown is a markup language with the file extension *.md*.

27.1 Headers

```
# H1
## H2
### H3
#### H4
##### H5
###### H6
```

27.2 Italics

```
*This would be in italics*
```

27.3 Bold

```
**This would be in bold**
```

27.4 Links

```
[go to google](https://www.google.com)
```

27.5 Lists

```
1. first item
1. second item
1. third item
```

27.6 Code Block

```
Inline `code` has `back-ticks around` it.
```

Blocks of code are fenced by lines with three back-ticks “‘

28 Random Bits

At the start of the semester, ask students to ask two people their names. Then, start by doing a quiz and ask what those two people’s names were. Should get a funny response and more importantly demonstrate that the brain is bad at remembering things and effort is needed to succeed.

Average page size is 2.4mb

28.1 Avoid random overrides in CSS

Set an id to the body tag, then when referring to another element on the same page, can then use a descendant (that element is a descendant of body). So when you add additional things in, you don't get unintentional overrides.

28.2 Design Tips

- Things that are higher than center look better.

28.3 Prefixes

Use autoprefixer for deployment (can also set it up within the IDE).

28.4 Gulp

Automate workflow (can add prefixes)

28.5 Grunt

The Javascript Task Runner.