# HTML & CSS

Ben Gavan

July 9, 2019

## Contents

# 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)

### 4.2 Inline

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

# 5 HTML Tags

## 5.1 div

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

## 5.2 span

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

## 5.3 a

**Attributes**

- 'href' urlm to open when clicked
- 'target' how to do it when clicked

  '_blank' opens url in new tab

# 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.

# 7 CSS Selectors

Different ways to select HTML elements.

## 7.1 element

Formats that HTML element.

## 7.2 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 {}"

## 7.3 id

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

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

```
#example {
}
```

## 7.4 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] {
}
```

## 7.5 Pseudo Class

Denoted by ":" (single :)

### 7.5.1 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.

### 7.5.2 Focus

Used for other elements such as forms.

### 7.5.3 $n^{th}$ 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 {
}
```

## 7.6 Pseudo Element

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

### 7.6.1 First-letter

Selects the first letter

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

### 7.6.2 First-line

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

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

## 7.7 Nested Selectors

### 7.7.1 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>
```

### 7.7.2 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>
```

### 7.7.3 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
```

### 7.7.4 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>
```

## 7.8 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'.

## 7.9 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 |

### 7.9.1 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.

# 8 Formatting Text

## 8.1 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.

### 8.1.1 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.

### 8.1.2 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.

#### 8.1.3 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

#### 8.1.4 font-style

#### 8.1.5 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

#### 8.1.6 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.

#### 8.1.7 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.

#### 8.1.8 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)

#### 8.1.9 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).

#### 8.1.10 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)

### 8.2 text

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

#### 8.2.1 text-transform

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

### 8.2.2 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.

### 8.2.3 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;

### 8.2.4 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'.

### 8.2.5 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

#### 8.2.6 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.

# 9 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.

## 9.1 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).

## 9.2 Semantic HTML

Non-semantic:

- div
- span

Semantic:

- header
- nav
- main
- article
- section
- aside

- footer
- h1 - h6
- figure
- figcaption
- address

## 9.3 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.

## 9.4 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>
```

## 9.5 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

## 9.6 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.

## 9.7 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.

## 9.8 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

## 9.9 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

## 9.10 figure and figcaption

### 9.10.1 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

#### 9.10.2 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>.

### 9.11 Address

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

## 10 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/

### 10.1 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.

## 10.2 Container Properties

### 10.2.1 display

- flex
- inline-flex

### 10.2.2 flex-wrap

- Nowrap (default)
- Wrap
- Wrap-reverse

### 10.2.3 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

### 10.2.4 flex-flow

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

### 10.2.5 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

### 10.2.6 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 }
```

### 10.2.7 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 }
```

## 10.3 Item Properties

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

### 10.3.1 align-self

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

```
1 .item {
2   align-self: auto | flex-start | flex-end | center |
        baseline | stretch;
3 }
```

### 10.3.2 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).

```
1 .item {
2   order: <integer>; (default is 0)
3 }
```

### 10.3.3 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.

```
1 .item {
2   flex-grow: <number>;  (default 0)
3 }
```

### 10.3.4 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)
}
```

### 10.3.5 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.

### 10.3.6 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.

# 11 Media Queries

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

### 11.0.1 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.

### 11.0.2 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">
```

### 11.0.3 Use

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

### 11.0.4 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

### 11.0.5 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.

### 11.0.6 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.

## 11.1 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

# 12 Google's Flexbox Design Patterns

## 12.1 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

## 12.2 Useful links

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

### 12.2.1 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.

# 13 Forms

```
1 <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.

# 14 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