

SSDP 1100 – Day 3

Course: Front-End Web Development Essentials
Instructor: Gabbie Bade

Agenda

- CSS Box Model
- Developer Tools
- Normalize.css
- Pseudo Classes
- Responsive Web Design
- Media Queries
- Assignment #2

CSS Box Model

CSS Box Model

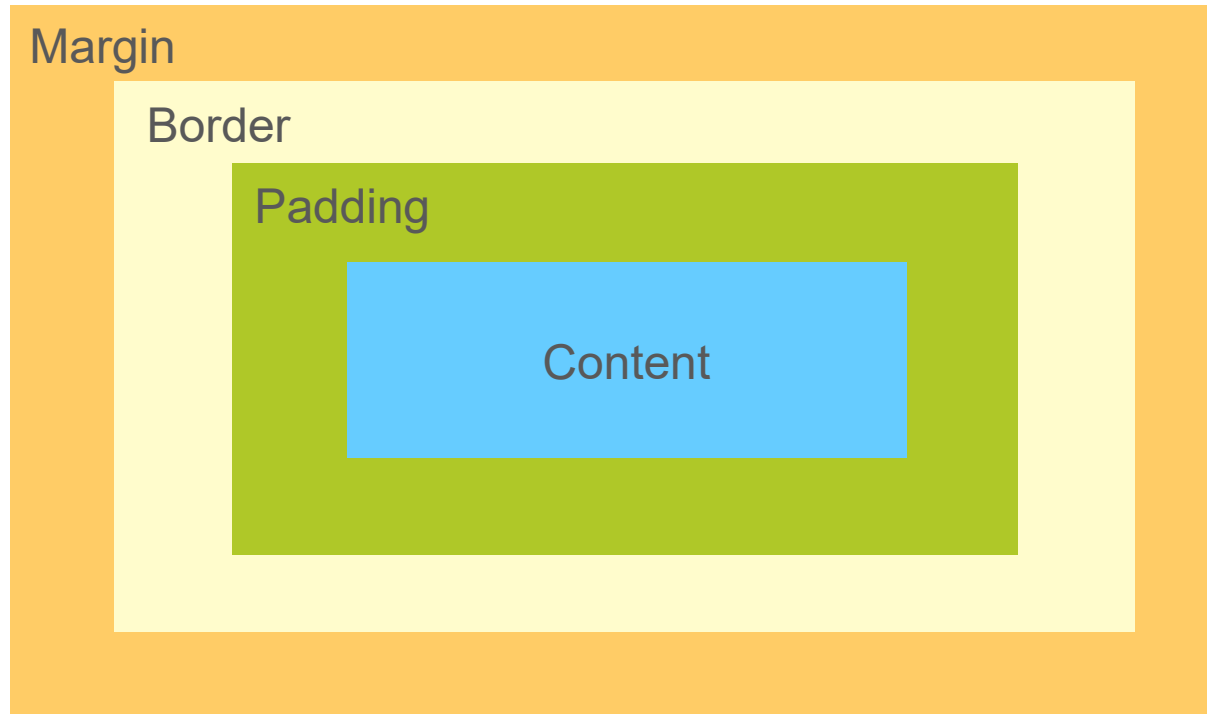
The CSS Box Model is the set of rules that CSS uses to determine the size of block-level elements.

The width and height of an element is determined by the addition of the following areas:

- Content + Padding + Border + Margin

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Introduction_to_the_CSS_box_model

CSS Box Model



Block-level Elements

Block-level elements...

- Have a default width of 100% of their parent's width. This width can be changed with CSS.
- Cause a line break.
- Height is determined by the content. This height can be changed with CSS.

Inline Elements

Inline elements...

- Do not cause a line break.
- Width is determined by the content inside their tags.
- No width or height can be set on them in CSS.

Changing Display Values

Block-level elements can be made inline and inline elements can be made block using CSS.

```
a.block {  
    display: block;  
}  
p.inline {  
    display: inline-block;  
}
```

← Any <a> tag with the class “block” will display like a block-level element.

← Any <p> tag with the class “inline” will display like an inline element.

<https://developer.mozilla.org/en-US/docs/Web/CSS/display>

Box Sizing Reset

The default box model in CSS is called “content-box” and is very confusing.

Example: You set an element’s width to 25%. Then add padding and a border to that element. It is now 25% **plus** the padding and border, potentially breaking your layout.

Because of this, we use a reset at the top of our CSS to switch to the “border-box” model.

Example: You set an element’s width to 25%. Then add padding and a border to that element. It is still 25% because the padding and border is added **within** that 25% width.

Box Sizing Reset Code

Add the following code at the top of your CSS files going forward...

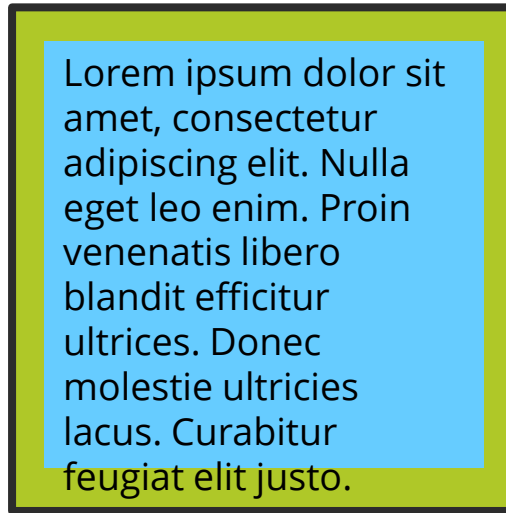
```
html {  
    box-sizing: border-box;  
}  
*, *:before, *:after {  
    box-sizing: inherit;  
}
```

<https://css-tricks.com/inheriting-box-sizing-probably-slightly-better-best-practice/>

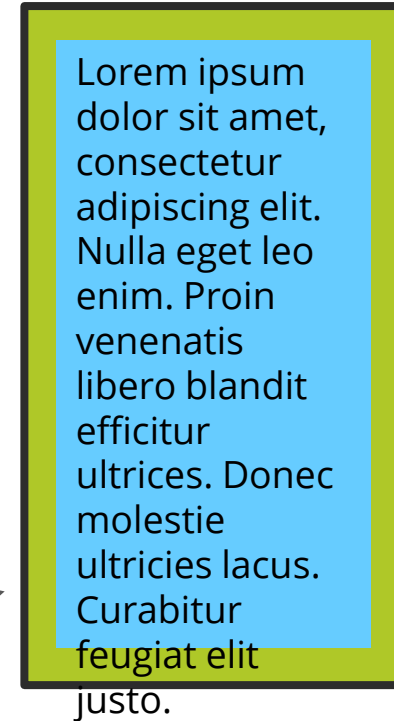
Border-box vs Content-box

If we set the width in CSS to 200px using the default “content-box” and then add padding and border, the width is actually larger than 200px because...

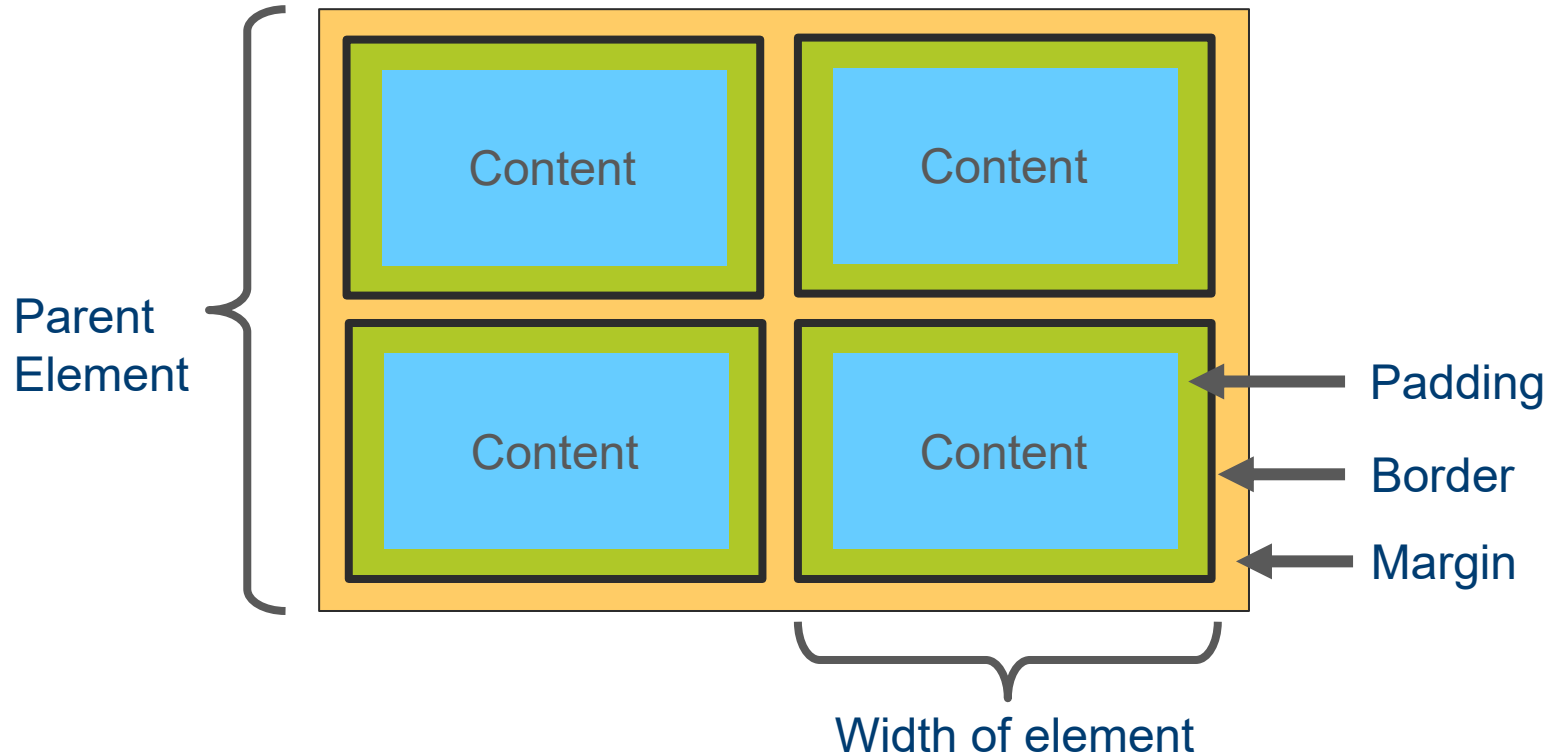
width + padding + border



Using the same CSS but switching to use “border-box”, the width stays at 200px.



CSS Margin, Padding, Border



CSS Margin, Padding, Border

Border is the line around the HTML element.

Padding is the space between the content and the border.

Margin is the space **around** the HTML element. It is not included in the width of the element.

CSS Margin & Padding

When setting margin or padding on block-level elements, you can write your CSS one of two ways.

```
.item {  
    margin-top: 1rem;  
    margin-right: 1.5rem;  
    margin-bottom: 1.25rem;  
    margin-left: .75rem;  
}
```

```
.item {  
    margin: 1rem 1.5rem 1.25rem .75rem;  
}
```

CSS Shorthand

The CSS shorthand for margin and padding can handle the following scenarios:

- All sides have the same value
- Left and right have the same value, top and bottom have the same value
- Left and right have the same value, top and bottom have different values
- Top, right, bottom, and left have different values

<https://developer.mozilla.org/en-US/docs/Web/CSS/Margin>

<https://developer.mozilla.org/en-US/docs/Web/CSS/Padding>

CSS Shorthand – Example 1

Example 1: All sides have the same value.

```
.item {  
    margin: 1.25rem;  
}
```

The top, right, bottom, and left will all have a margin of 1.25rem, or 20px for most browsers.

CSS Shorthand – Example 2

Example 2: Top and bottom have the same value, left and right have the same value.

```
.item {  
  margin: 1rem .5rem;  
}
```

The first value represents the top and bottom margins.

The second value represents the left and right margins.

CSS Shorthand – Example 3

Example 3: Top and bottom have different values, left and right have the same value.

```
.item {  
  margin: 1rem .5rem 2rem;  
}
```

The first value
represents the
top margin.

The second value
represents the left
and right margins.

The third value
represents the
bottom margin.

CSS Shorthand – Example 4

Example 4: Top, right, bottom, left have different values.
Notice it moves clockwise in this scenario (see next slide).

```
.item {  
  margin: 1rem .5rem 2rem .75rem;  
}
```

The first value
represents the
top margin.

The second value
represents the
right margin.

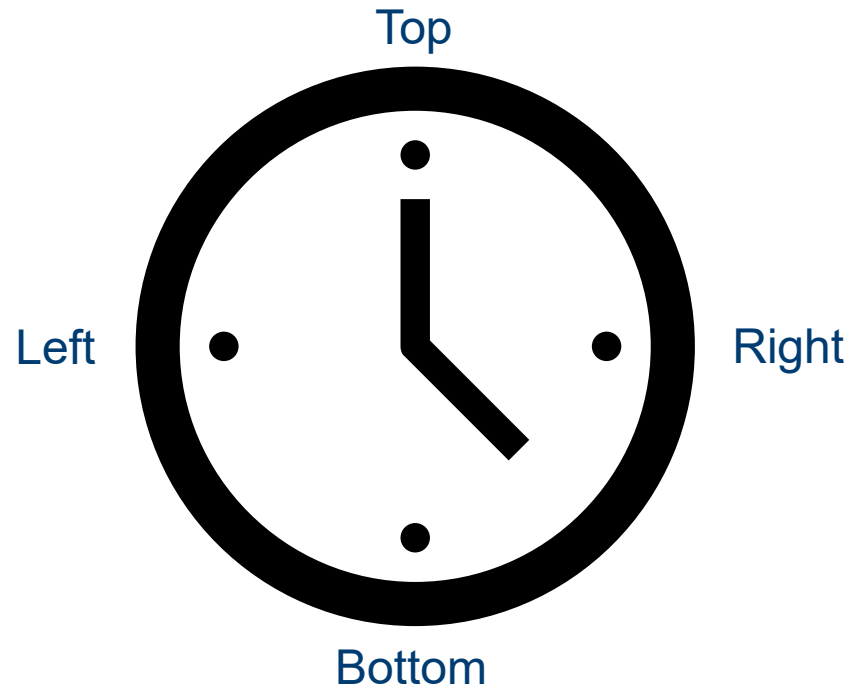
The third value
represents the
bottom margin.

The fourth value
represents the
left margin.

CSS Shorthand

```
.item {  
  margin: 1rem .5rem 2rem .75rem;  
}
```

When all sides have different values, you can think of the CSS shorthand as a clock moving from 12 to 3 to 6 to 9.



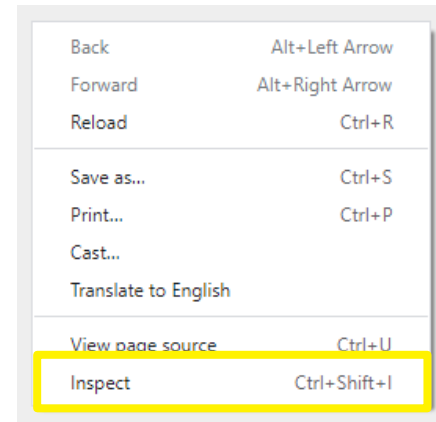
Developer Tools

Dev Tools in Browsers

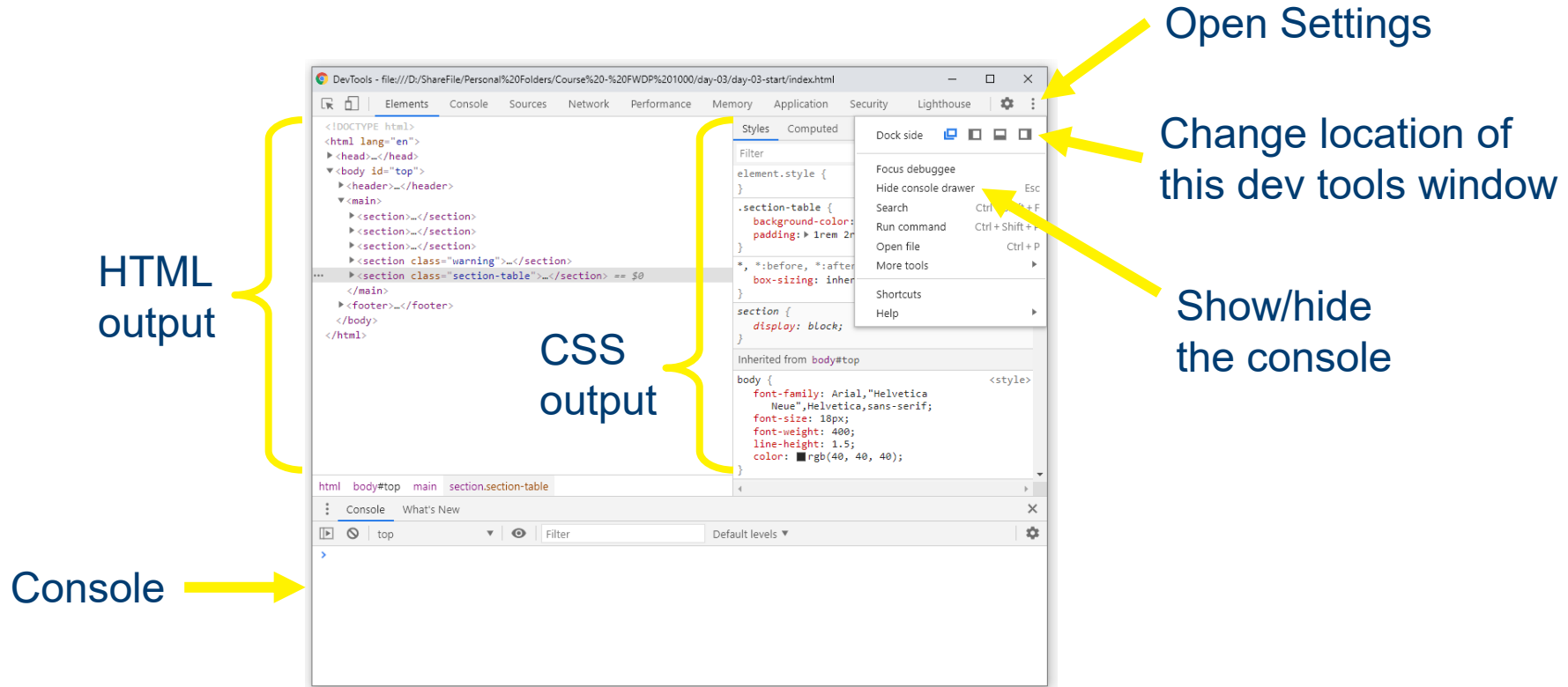
Every browser has a version of Developer Tools to help you troubleshoot and build websites.

It can be opened by pressing F12 or right-clicking the page and choosing Inspect.

They are similar across browsers but Chrome and Firefox have the best developer tools.



Chrome Dev Tools



Testing CSS in Chrome Dev Tools

Write inline CSS on whatever element you have targeted in the HTML

See the margin, border, padding and width/height of the selected element



Write new CSS properties with the + symbol

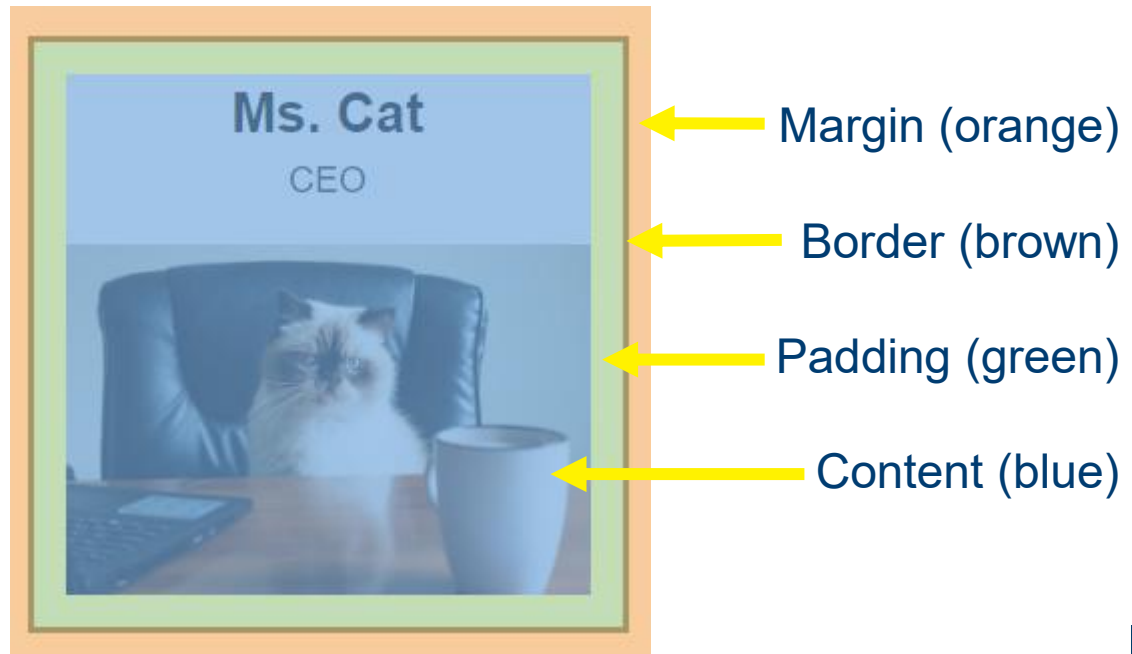
See all of the CSS applying to the selected element and the hierarchy

Inspecting Elements

What it looks like
in the browser...

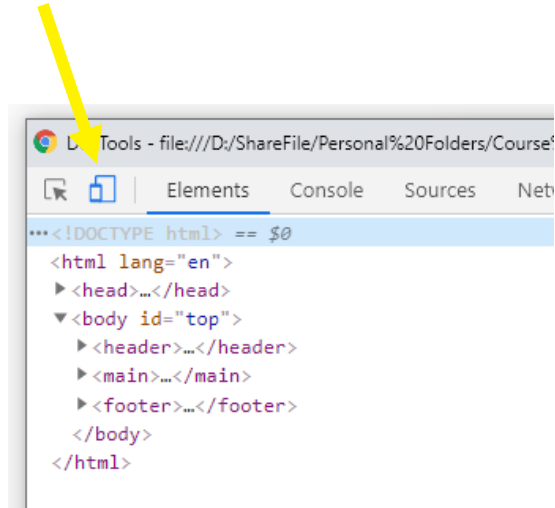


What it looks like when
inspecting in dev tools...



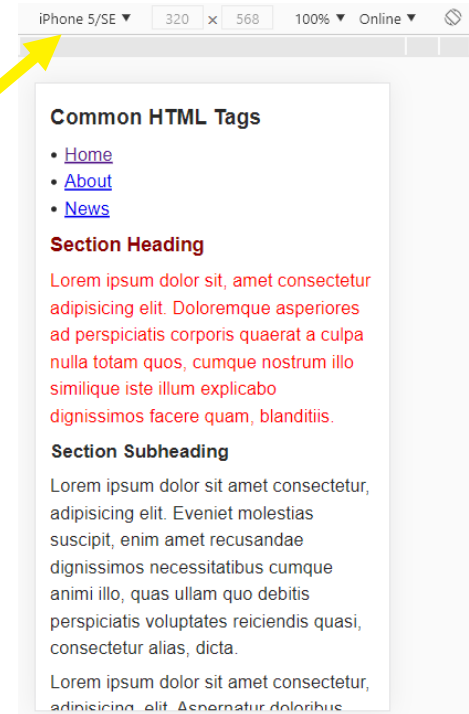
Responsive Layouts Tool

Click here to open the device toolbar



You can select devices to see what the webpage will look like on them.

Or choose “Responsive” to adjust to any size by dragging the side or bottom.



Developer Tools

Nothing you do in the developer tools will save anywhere.

As soon as you refresh your page, everything will be lost.

It is an **extremely** useful tool for testing, troubleshooting and understanding.

Responsive Design

Mobile First CSS

Over 55% of all page views on the web come from mobile devices.

Another ~2% come from tablets.

<https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/>

Mobile First CSS

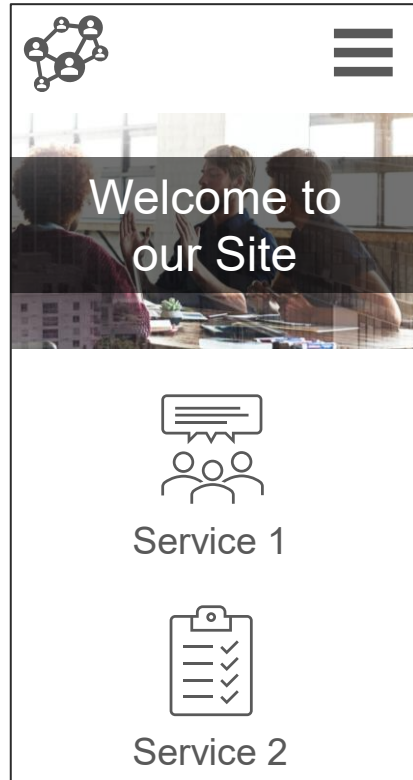
When we write our CSS, we write it “mobile first”.

This means we write our styles according to how the webpage will look on smaller devices, **then** we write styles for how it will look on laptops and desktops.

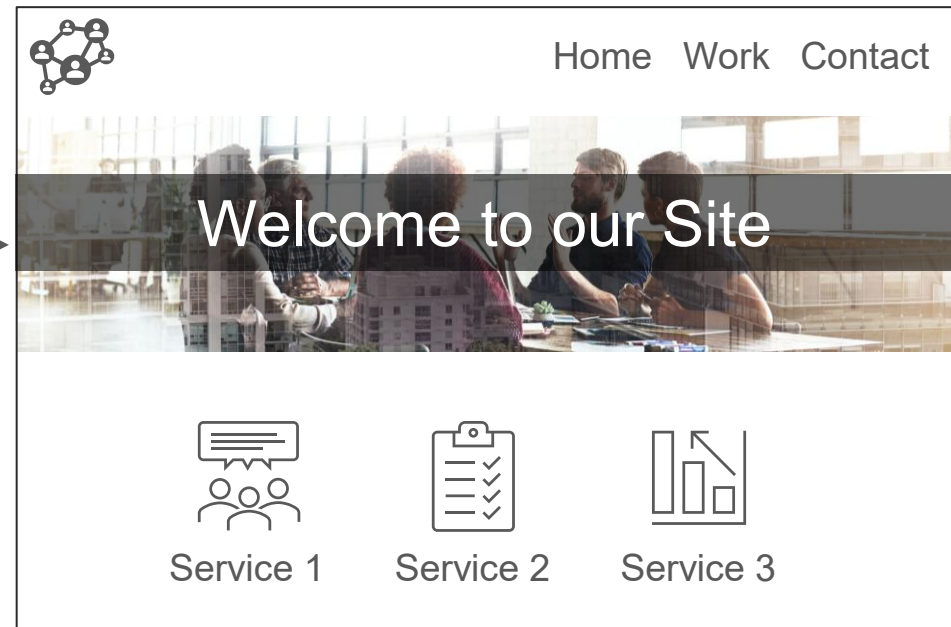
Larger screens involve more advanced layouts, we will cover how to do that with CSS later.

Desktop vs. Mobile Layout

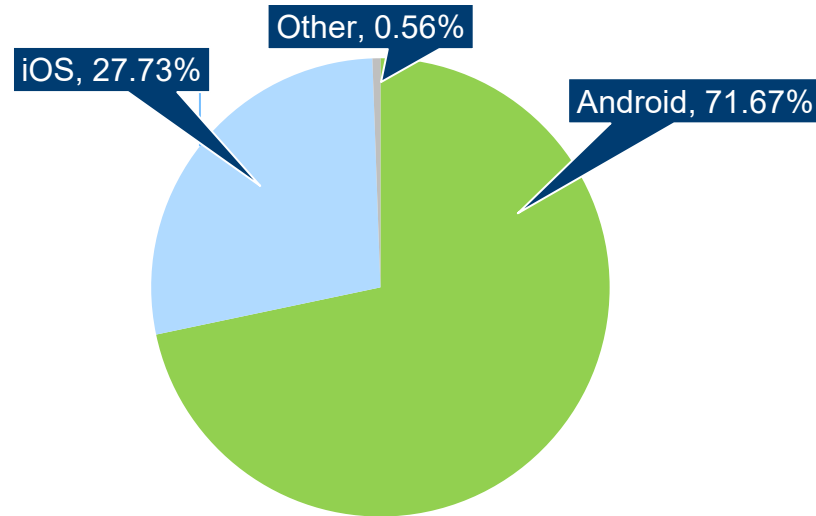
Mobile layouts are usually a single column because of the narrow screen width.



Desktop layouts are usually multiple columns because of the wider screen width.

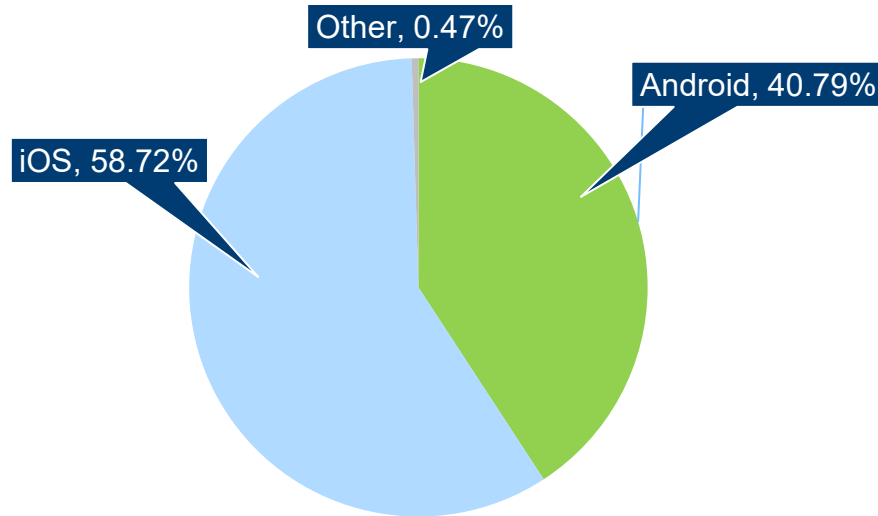


Smartphone Market Share (Global)



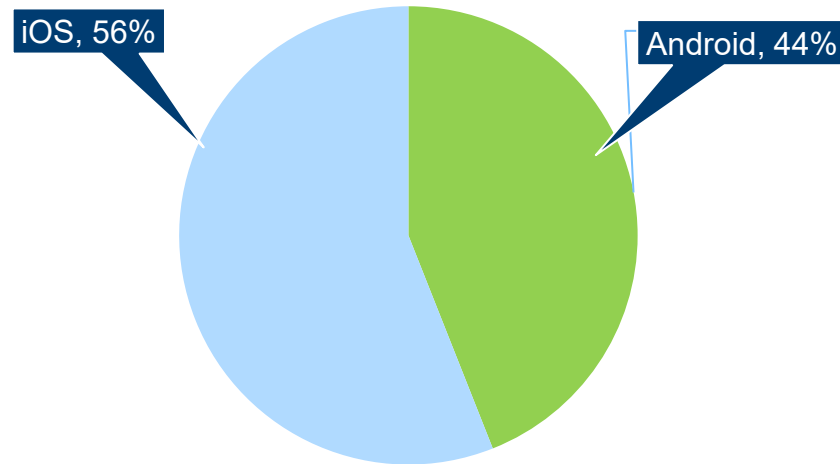
<https://gs.statcounter.com/os-market-share/mobile/worldwide>

Smartphone Market Share (Canada)



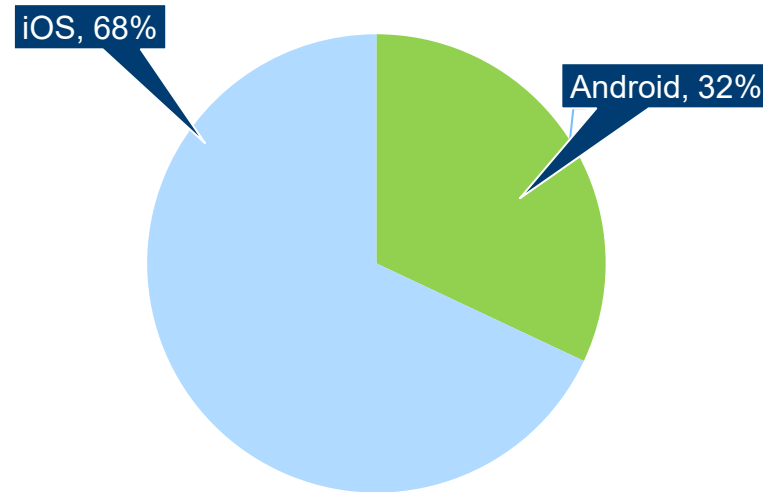
<https://gs.statcounter.com/os-market-share/mobile/canada>

Tablet Market Share (Global)



<https://gs.statcounter.com/os-market-share/tablet/worldwide>

Tablet Market Share (Canada)



<https://gs.statcounter.com/os-market-share/tablet/canada>

Responsive Design

Responsive Web Design (RWD) means creating a single website that adjusts according to the screen size, orientation and/or device.

Instead of making separate websites for mobile and desktop, we make one website that adjusts to the situation.

Some visual examples: <https://mediaqueri.es/>

Device Resolution vs Viewport Size

To understand pixel density, consider the iPhone 12...

- It has a device resolution of 1170px by 2532px.
- It has a viewport size of 390px by 844px.

Simply put... three pixels are being squished into one pixel on the screen to make the display crisper.

Viewport Size

For our CSS, we only care about the viewport size.

Using our iPhone 12 example, that means all of our content needs to fit in a space with a maximum width of 390px.

Device viewports and resolutions:

<https://experienceleague.adobe.com/docs/target/using/experiences/vec/mobile-viewports.html>

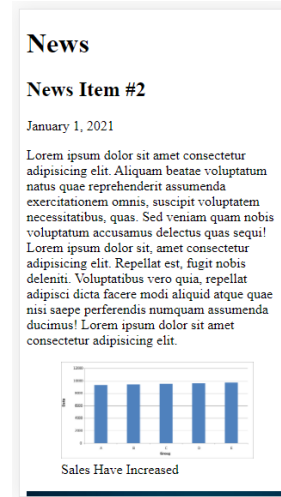
Viewport Size

There's just one problem... browsers don't use the device's viewport size by default when rendering our webpages.



This is the default way smartphones handle websites.

The same webpage with the proper HTML meta tag added.



Viewport Meta Tag

To use the device's viewport width, we add the following code to into the <head> element of all HTML files:

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

This tells the browser to use the device's viewport width when rendering the page.

This tells the browser to set the initial device zoom level to 1.

https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag

Viewport Meta Tag

Make sure that code is in the `<head>` section of every HTML file you create going forward.

Media Queries

Media Queries

Media queries are CSS conditional statements that run code only when the condition is true.

For example, a media query may say “run the following code if the viewport is 400px or wider”.

Media queries can be used to apply styles based on screen size, screen orientation, user preferences, print mode, and more.

Media Query Syntax

The most common way to write a media query is within your existing stylesheets.

```
@media (min-width: 400px) {  
  
    body {  
        background-color: red;  
    }  
  
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

Media Query Syntax

This tells the browser to only run the code within the curly brackets if the viewport width is 400px or higher.

```
@media (min-width: 400px) {  
  
    body {  
        background-color: red;  
    }  
  
}
```

These styles will only apply when the condition above is met.

Complex Media Queries

This media query has three conditions and they all must be true for the code inside to apply.

```
@media screen and (min-width: 25em) and (max-width: 3.125em) {  
    body { background-color: red; }  
}
```

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Media_queries

Using ems in Media Queries

We use the **em** unit in our media query conditions for accessibility.

If the user has changed the text size in their browser, then our media queries will use that new base font value.

In a media query, the value of **em** will be the same as the value of **rem**.

- This is only true for media queries! Anywhere else in your CSS and the **em** value inherits from its ancestors so *may* be different than **rem**.

Calculating the em Value

The default value of **rem** and **em** for most browsers is 16px.

If you want a breakpoint at 800 pixels...

$$800 / 16 = 50$$

So your media query would be...

```
@media (min-width: 50em) { }
```


Writing Mobile First CSS

When you start writing your CSS, follow this process:

1. Resize the browser down to mobile screen size manually or by using your browser's Developer Tools (see Day 3 slides).
2. Write your general styles (for all screen sizes) and your mobile styles.
3. Resize the browser up until the layout breaks or looks bad and note the screen width.
4. Write a media query with that screen width and put your tablet and/or desktop styles inside the media query.
5. Repeat steps 3 and 4 until styling is complete.

Testing Screen Sizes

In addition to resizing your browser or using Developer Tools to test for different screen sizes...

It's always ideal to test on a real device (your phone, your tablet, etc.)

There are also services like Browser Stack:

<https://www.browserstack.com/>

Container Queries

A new CSS featured called a **Container Query** works like a Media Query but is based on an element's width instead of the entire document's width.

It has [92.76% browser support](#) but you can practice it now to use it later.


<https://developer.mozilla.org/en-US/docs/Web/CSS/@container>

Normalize CSS

Cross Browser CSS

All browsers have default styles that they apply to some HTML elements. These styles are not always the same from one browser to the next though.

Example of the default styles in Chrome for an `<h1>` element



```
h1 {  
  display: block;  
  font-size: 2em;  
  margin-block-start: 0.67em;  
  margin-block-end: 0.67em;  
  margin-inline-start: 0px;  
  margin-inline-end: 0px;  
  font-weight: bold;  
}
```

user agent stylesheet

To address this, developers tend to use either Eric Meyer's CSS Reset or Normalize.css.

Normalize vs Reset

The CSS Reset removes the browser defaults entirely.

<https://meyerweb.com/eric/tools/css/reset/>

Normalize.css instead tries to make the defaults consistent so we can still take advantage of the defaults.

<http://necolas.github.io/normalize.css/>

Normalize.css gives us less work and provides nice fallbacks for any element we didn't style, so we will use that.

Normalize.css

To use **normalize.css** all you need to do is download it as a **.css** file and attach the CSS file to your HTML file.

Normalize.css should be attached **before** any other stylesheets so your styles can override these defaults.

```
<head>
  <meta charset="utf-8">
  <title>Page Title</title>
  <link rel="stylesheet" href="styles/normalize.css">
  <link rel="stylesheet" href="styles/styles.css">
</head>
```

Responsive Media

To make sure your images and a few other elements don't expand outside of their parent elements, add the following code to your normalize.css file...

```
embed,  
iframe,  
object {  
    max-width: 100%;  
}  
  
img,  
video {  
    max-width: 100%;  
    height: auto;  
}
```


Rename **normalize.css**

Since we have modified the original **normalize.css** file, we should rename it so we know it is different than the original.

I will rename mine to **normalize-ssd.css**.

Going forward...

Consider moving the box sizing reset code from earlier to the bottom of your `normalize.css` file.

Now, you can use this `normalize.css` file on all of your webpages for any project.

Pseudo-classes

CSS Pseudo-classes

A pseudo-class is a keyword added to a CSS selector that specifies a special state.

For instance, when someone hovers over an <a> element, we can change the color...

```
a:hover {  
    color: red;  
}
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes>

Common Pseudo-classes

- `:visited` – style an `<a>` element if the user has already visited the URL.
- `:hover` – style any HTML element when the user hovers the mouse over the element. Commonly used for links and buttons.
- `:active` – style any HTML element when the user is holding down the mouse button. Commonly used for links and buttons.
- `:focus` – style any focusable HTML element once it has been clicked into or selected with the keyboard Tab key. Commonly used for links, buttons and form fields but any HTML element with the “`tabindex`” attribute can accept focus.

Similar Pseudo-classes

There are two more “focus” pseudo-classes for specific situations:

- `:focus-visible` – lets the browser only apply focus styles if keyboard navigating, instead of using a mouse.
- `:focus-within` – style a parent element if any of its children have focus.

Anchor Link Order

If using these pseudo-classes on your <a> tags, you should put them in the following order:

```
a {}  
a:visited {}  
a:hover {}  
a:focus {}  
a:active {}
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/:active>

Assignment #2

Assignment #2

- Please refer to Assignment #2 in the Learning Hub.
- To submit the assignment, you can do **one** of these:
 - Have me check your assignment in class before 4pm.
 - Zip today's **folder** and upload it to the Learning Hub before next class.
- If you have questions or need guidance, just ask!

Resources – Media Queries

Media Queries (MDN -- Web)

https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

Media Queries (MDN -- Learn)

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Media_queries

Why use em for media queries? (Medium)

<https://medium.com/zoosk-engineering/to-em-or-not-to-em-that-is-the-media-query-question-22f4a65e9747>

Resources

Rems vs Pixels

<https://www.joshwcomeau.com/css/surprising-truth-about-pixels-and-accessibility/>

Box-sizing Reset

<https://css-tricks.com/inheriting-box-sizing-probably-slightly-better-best-practice/>

Normalize.css

<http://necolas.github.io/normalize.css/>

MDN Web Docs – CSS

<https://developer.mozilla.org/en-US/docs/Web/CSS>

QUESTIONS & ANSWERS