CSS:
Cascading Style Sheets

What is CSS?

It is a "style sheet language" to describe the layout and presentation of a markup language, most commonly HTML.

It is text written to alter how other text (markup) is displayed.

But first, some history!

Timeline

- 1969-1990 ARPANET
  - "This has made a lot of people very angry and been widely regarded as a bad move." - Douglas Adams
- 1989 Tim Berners-Lee writes RFC for the World Wide Web
- 1990 Tim Berners-Lee writes the first version of HTML
  - Nov 24, 1995 HTML 2 RFC 1866
  - Nov 24, 1995 HTML with form-based file uploads RFC 1867
  - Jan 14, 1997 HTML 3 W3C Recommendation
  - Dec 18, 1997 HTML 4 W3C Recommendation
  - Oct 28, 2014 HTML 5 W3C Recommendation

Sidebar: Request for Comments (RFCs)

- A way to standardize a syntax or protocol with respect to the "internet"
  - Anyone can make a proposal
  - Anyone can comment
- Published to the IETF (Internet Engineering Task Force) of the ISOC (Internet Society, a non-profit) aka a different group of people with vested interests that this whole "internet thing" stays working
- RFCs either get revised, go away, or become the standardRFCs can be in one of the following states: Informational, Experimental, Best Current Practice, Standards Track, or Historic

Sidebar: Some popular RFCs

- Why is my email address formatted this way? https://datatracker.ietf.org/doc/html/rfc5322#section-3.4
- What are valid characters I can have in a URL and how can I add data via query parameters? https://datatracker.ietf.org/doc/html/rfc3986/#section-3.4
- What are we gonna do about running out of IPv4 addresses? https://datatracker.ietf.org/doc/html/rfc8200
- What is this IRC thing the "engineers" are so fond of? https://datatracker.ietf.org/doc/html/rfc2810

Sidebar: RFCs versus W3C "Recommendations"

- What is the W3C?
    - World Wide Web Consortium, founded 1994, international community that any organization can join
- What is a W3C Recommendation?
    - A way to standardize a syntax or protocol with respect to the "world wide web"
- What is a W3C Standard?
    - A recommendation that has been edited, agreed upon, and finalized
- Very similar processes with different names/groups involved
    - W3C cares about the "web" and less so the "internet" so things like TCP/IP are RFCs and things like HTML/CSS/Web Sockets are W3C Recommendations

Sidebar fin.

There are many "standards" at various stages of **adoption**. (obligatory xkcd 927)

Everything that is good and bad about the internet is because it is a collaborative and evolving project.

Anyone can make a RFC or make a comment

RFCs and W3Cs are generally very dry and for people that are *implementing* the spec not using the spec so they contain information about how things work not how to use them.

The rest of this talk will be about how to use them. Just know that the theoretical "truth" is contained somewhere between the specs and the source code of the implementations.

Timeline Continued

- Dec 4, 1995 Netscape released and with it the first version of JavaScript
- Dec 17, 1996 CSS1 (aha! finally! Håkon Wium Lie makes the initial proposal)
    - https://www.w3.org/TR/CSS1/
    - "The first commercial browser to support CSS was Microsoft's Internet Explorer 3, which was released in August 1996." https://www.w3.org/Style/CSS20/history.html (That was prior to the finalized spec which means that IE had to change later.)
- May 12, 1998 CSS2
    - https://www.w3.org/TR/2008/REC-CSS2-20080411/
- Jun 7, 2011 CSS2.1
    - Primarily for backwards and forwards compatibility and to lay groundwork for CSS3
    - https://www.w3.org/TR/CSS2/

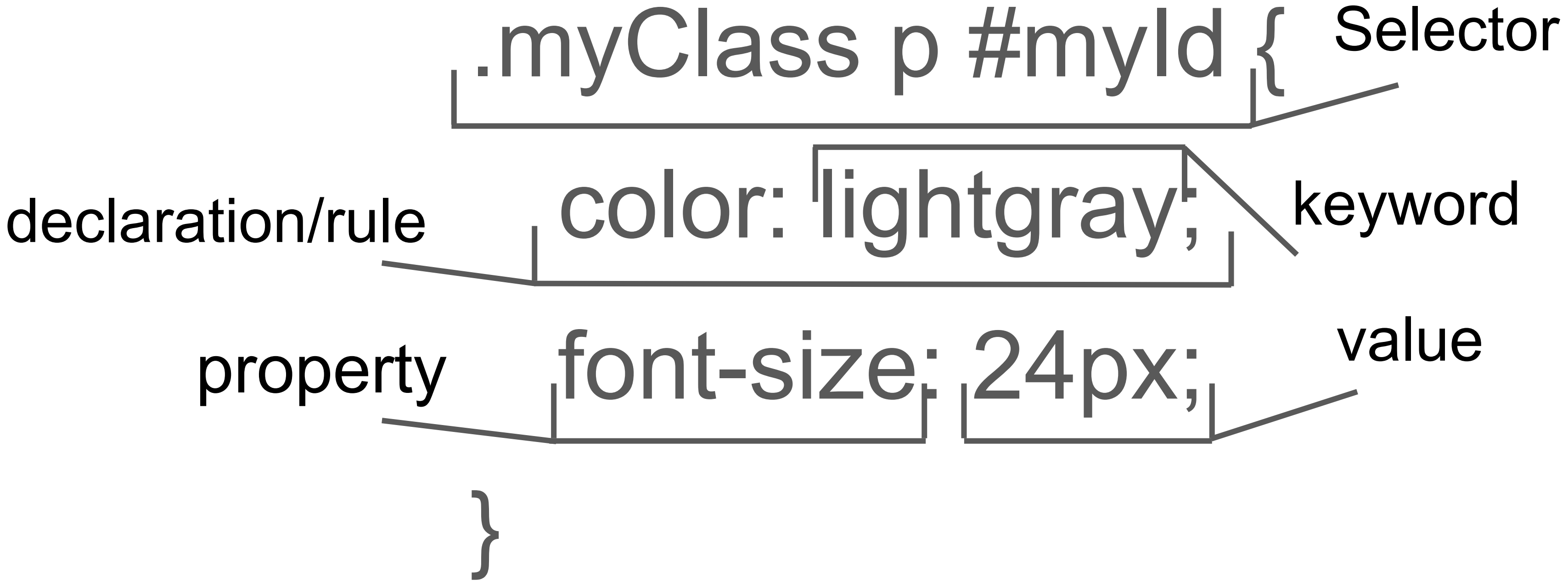
CSS3+

- Began draft work after CSS2 was finalized in 1999
- After CSS2.1 CSS3 became "modularized" now each specific thing in CSS is a module and they can be updated independently of each other. CSS 2.1 is the base and modules override portions as needed
- There are now "Snapshots" of the list of all of the standards that make up the current CSS standard. Each module has its own level/version
  - https://www.w3.org/TR/css-2020/
  - Slight lag between release snapshot and implementations

Okay wait so what is CSS?

- Text you write to describe how other text you write looks
    - Where in both cases it may be surrounded by text you *didn't* write
        - Examples of things that can produce additional CSS that your CSS interacts with explicitly or implicitly:
            - Other developers
            - Frameworks
                - CSS
                - JavaScript
            - User Generated Content

CSS Syntax Declaration Blocks



```
.myClass p #myId {
  color: lightgray;
  font-size: 24px;
}
```

CSS Syntax

- A declaration can have the "!important" modifier as follows
  - property: value !important;

- All declarations in the block apply to all the elements that match the selector
  - If an element matches multiple selectors all the rules for all the selectors are applied and conflicts are resolved via the cascade

Properties

- Not all properties are applicable/actually do things for all elements
- An incomplete list of properties that can be set:

background-attachment
background-color
background-image
background-position
background-repeat
border
border-bottom
border-bottom-color
border-bottom-style
border-bottom-width
border-color
border-left
border-left-color
border-left-style
border-left-width
border-right
border-right-color
border-right-style
border-right-width
border-style
border-top
border-top-color
border-top-style
border-top-width
border-width
clear
clip
color
cursor
display
filter
float
font
font-family
font-size
font-variant
font-weight
height
left
letter-spacing
line-height
list-style
list-style-image
list-style-position
list-style-type
margin
margin-bottom
margin-left
margin-right
margin-top
overflow
padding
padding-bottom
padding-left
padding-right
padding-top
page-break-after
page-break-before
position
stroke-dasharray
stroke-dashoffset
text-align
text-decoration
text-indent
text-transform
top
vertical-align
visibility
width
z-index

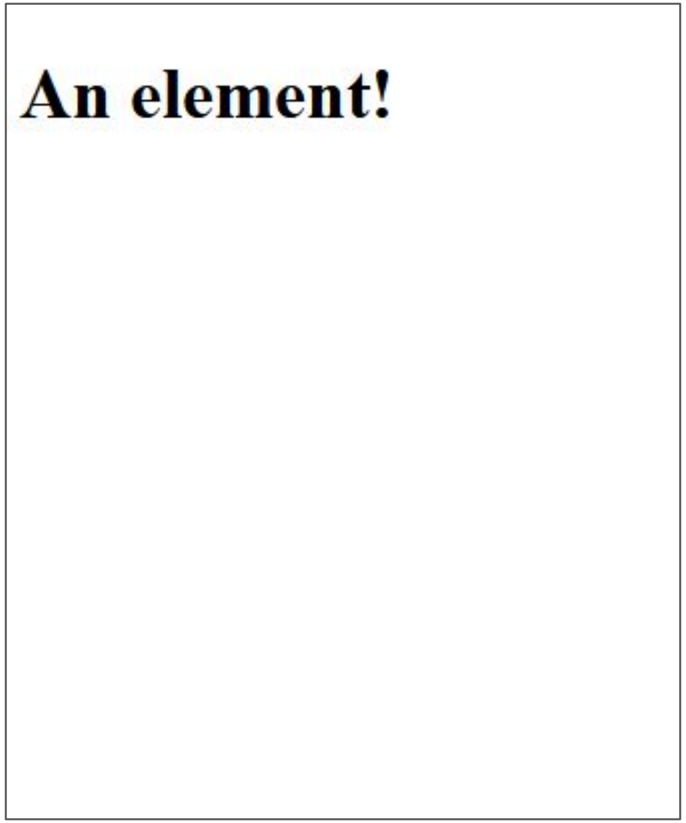
```
/* This slide intentionally left blank. */
.empty-slide {
    display: none;
}
```

An example

Input

```
<!DOCTYPE html>
<html>
        <head>
                <title>An example</title>
        </head>
        <body>
                <h1>An element!</h1>
        </body>
</html>
```

Output



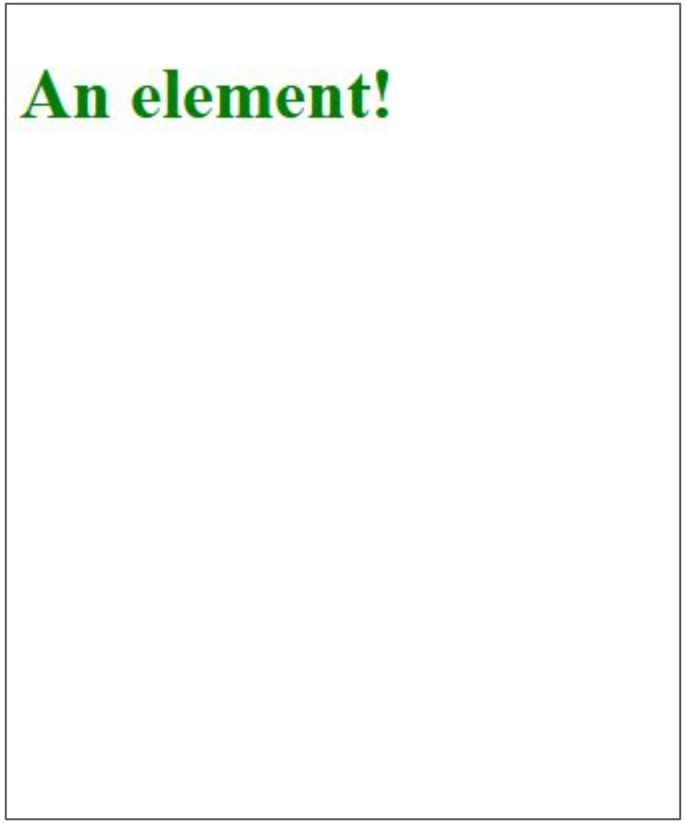
An element!

An example

Input

```
<!DOCTYPE html>
<html>
        <head>
                <title>An example</title>
                <style>
                        h1 {
                                color: green;
                        }
                </style>
        </head>
        <body>
                <h1>An element!</h1>
        </body>
</html>
```

Output



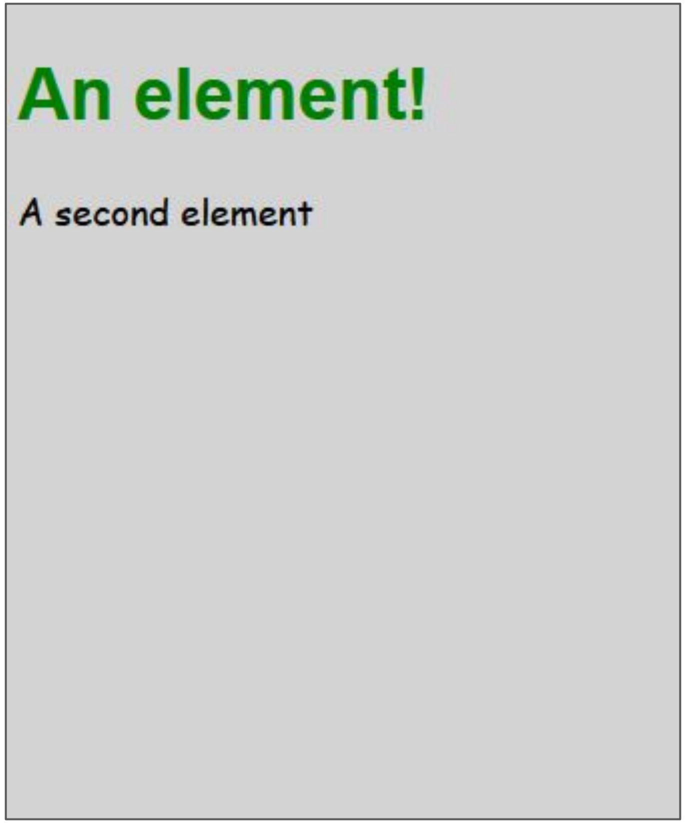
An element!

An example

Input

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            h1 {
                color: green;
                font-family: Arial, sans-serif;
            }
            body {
                background: lightgrey;
                font-family: "Comic Sans MS",
                cursive;
            }
        </style>
    </head>
    <body>
        <h1>An element!</h1>
        <p>A second element</p>
    </body>
</html>
```

Output



An element!

A second element

```
/* This slide intentionally left blank. */
.empty-slide {
    display: none;
}
```

CSS Selectors

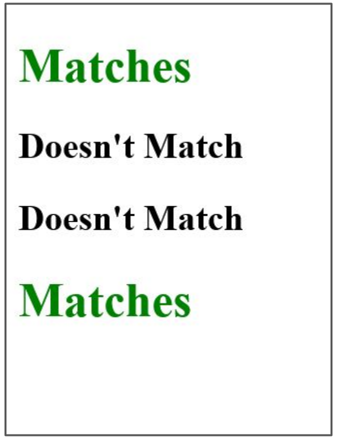
- HTML elements have:
  - tags/a type: h1, p, div
  - relationships: they are children or siblings or other elements
  - an Id (optional) <div id="customIdThatWeGetToMakeUp"></div>
  - classes (optional) <p class="someOtherNameWeMake aSecondClassEven"></p>
    - The order of classes on the element doesn't matter both of the above classes are applied based on their ordering in the style sheets
  - attributes
  - many other things
- CSS Selectors can select things by:
  - tags/type: h1, p, div
  - relationships: div > p
  - ID: #customIdThatWeGetToMakeUp
  - class: .aSecondClassEven
  - attributes
  - pseudo-attributes

CSS selectors examples

- h1
  - Read: all of the HTML elements that have a type of "h1"

```
<h1>Matches</h1>
<h2>Doesn't Match</h2>
<h2 class="h1">Doesn't Match</h2>
<div>
    <h1>Matches</h1>
</div>
```

```
h1 {
    color: green;
}
```



Matches

Doesn't Match

Doesn't Match

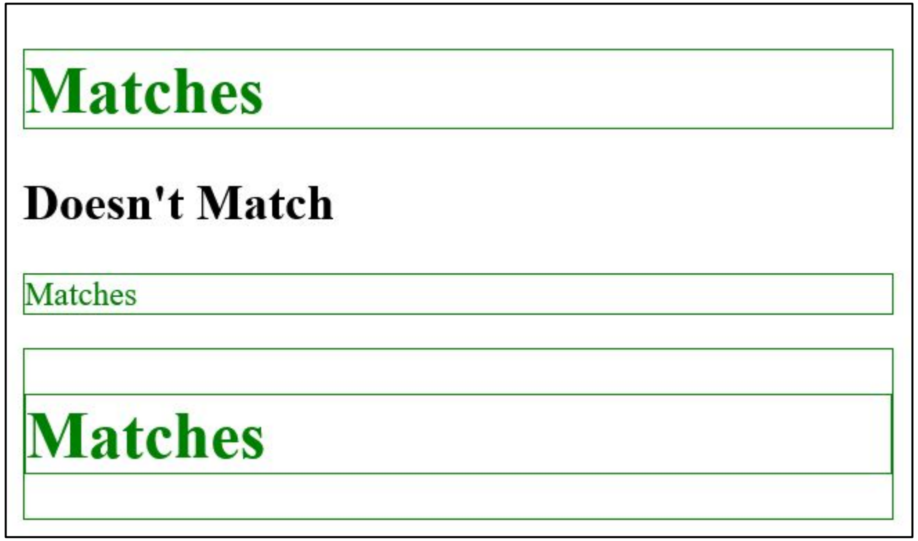
Matches

CSS selectors examples

- h1, p, div
  - Read: all of the HTML elements that have a type of "h1", "p", or "div"

```
<h1>Matches</h1>
<h2>Doesn't Match</h2>
<p>Matches</p>
<div>
    <h1>Matches</h1>
</div>
```

```
h1, p, div {
    color: green;
    border: 1px solid green;
}
```


Matches

Doesn't Match

Matches

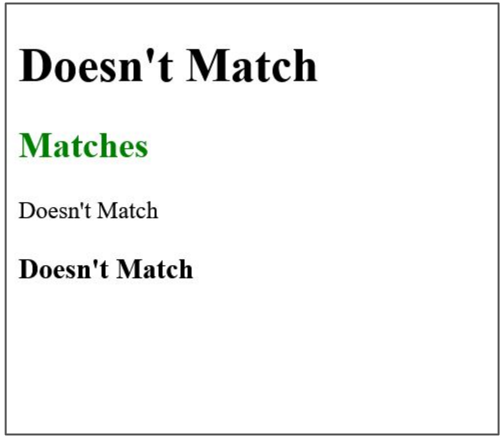
Matches

CSS selectors examples

- #someId
  - Read: the singular HTML element that has the id="someId"; it is invalid HTML to have two elements with the same id

```
<h1>Doesn't Match</h1>
<h2 id="someId">Matches</h2>
<p>Doesn't Match</p>
<h3>Doesn't Match</h3>
```

```
#someId {
    color: green;
}
```



Doesn't Match

Matches

Doesn't Match

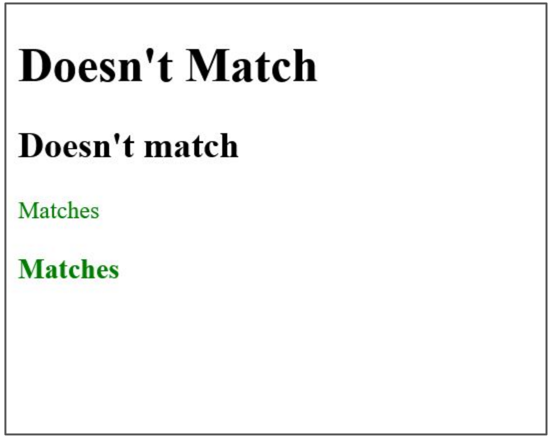
Doesn't Match

CSS selectors examples

- .someClass
  - Read: all of the HTML elements that have the class="someClass"

```html
<h1 class="someclass">Doesn't Match</h1>
<h2 id="someId">Doesn't match</h2>
<p class="someClass">Matches</p>
<h3 class="someClass">Matches</h3>
```

```css
.someClass {
    color: green;
}
```



Doesn't Match

Doesn't match

Matches

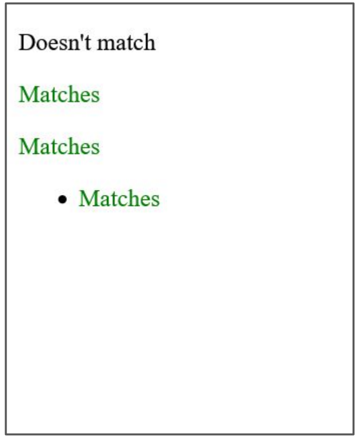
Matches

CSS selectors examples

- div p
  - Read: all of the paragraph tags that are inside of a div element

```
<p>Doesn't match</p>
<div>
    <p>Matches</p>
    <div><p>Matches</p></div>
    <ul>
        <li><p>Matches</p></li>
    </ul>
</div>
```

```
.div p {
    color: green;
}
```



Doesn't match

Matches

Matches

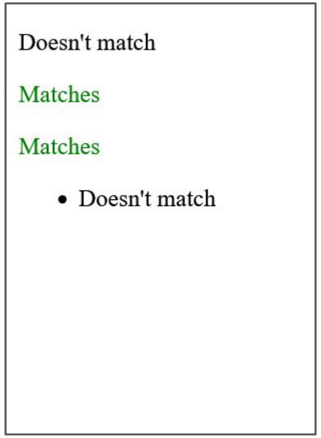
- Matches

CSS selectors examples

- div > p
  - Read: All HTML elements that have tag type p that are direct descendants of a div tag

```
<p>Doesn't match</p>
<div>
    <p>Matches</p>
    <div><p>Matches</p></div>
    <ul>
        <li><p>Doesn't match</p></li>
    </ul>
</div>
```

```
.div > p {
    color: green;
}
```


Doesn't match

Matches

Matches

- Doesn't match

CSS selectors examples

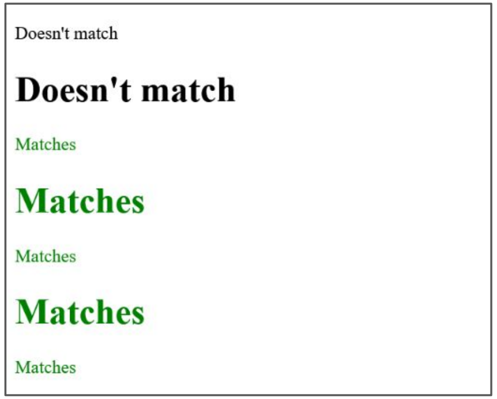
- a:visited
  - These are called "pseudo" selectors because they select things that are not actually part of the HTML
  - Read: select all the anchor tags (links) on the page that this user has "visited before"
  - It is the browser's job to remember which links have been visited
  - This is how when you search for something and you see the link is purple instead of blue you know you've been to that site before
    - At least until your browser history is deleted

CSS selectors examples

- div > div > h1.myClass, p.myOtherClass, .greenStuff
  - Read: all h1 tags that have "myClass" as a class that are direct descendents of a div tag that are a direct descendent of a div tag, all paragraph tags that have "myOtherClass" as a class, and anything with the class "greenStuff"

```
<p>Doesn't match</p>
<h1 class="myClass">Doesn't match</h1>
<p class="myOtherClass">Matches</p>
<div>
    <div>
    <h1 class="myClass">Matches</h1>
    <p class="myOtherClass">Matches</p>
    </div>
<h1 class="myClass
greenStuff">Matches</h1>
    <p class="myOtherClass">Matches</p>
<div>
```

```
div > div > h1.myClass,
p.myOtherClass, .greenStuff {
    color: green;
}
```



Doesn't match

Doesn't match

Matches

Matches

Matches

Matches

Matches

Sidebar Comments

- You can add comments to CSS (and JavaScript) like this:
  - Block comment/multi-line comment:
    - /**
    - * My comment's first line
    - * My comment's second line
    - **/
- You can add comments to HTML like this:
  - <!-- My comment -->
- Comments are useful to you later to remember what you did
- Comments are *publicly visible to anyone that views the site* so don't link to resources that you don't want other people to see. or cuss?

Ways to add CSS to a page

1. Inside of a <style></style>, only "valid" HTML if it is inside of the <head> tag
2. Linked to from a separate sheet <link rel="stylesheet" href="style.css">
    a. Most common and preferred
3. As "inline styles"
    a. <h1 **style="color:green;"**>A green HTML element</h1>
    b. This messes with the cascade order so should be used sparingly

```css
/* This slide intentionally left blank. */
.empty-slide {
    display: none;
}
```

Cascade Matters

1. It is called "Cascading" for a reason
    a. The rules themselves cascade down the HTML elements, e.g. if `<body>` font-family is Times New Roman all tags inside `<body>` will be Times New Roman unless a tag specifies its own font-family.
        i. If your parent element has a property you have that property unless you override or unset it.
    b. How rules are chosen cascades through the style sheets

What is the cascade order? (Abridged)

1. !important User declarations // For example if I am vision impaired and I set my browser's font to override a sites
2. !important Author declarations // What we are writing ie the CSS provided by the site
3. Normal Author declarations
4. Normal User declarations
5. "Encapsulation Context" (Safely Ignore - might affect iframes)
6. Specificity
7. Order of Appearance

Cascade - 6. Specificity

- The more *specific* a *selector* the more important the rule/declaration
- The more specific selector's rules get applied *last* and thus are the most likely to show up
- Style attributes directly on html elements have highest specificity (effectively author origin in the cascade)
- Selectors separated by commas are multiple different selectors and each have their own specificity

Cascade - 6. Specificity Calculation

> A selector’s specificity is calculated for a given element as follows:
>
> - count the number of ID selectors in the selector (= A)
> - count the number of class selectors, attributes selectors, and pseudo-classes in the selector (= B)
> - count the number of type selectors and pseudo-elements in the selector (= C)
> - ignore the universal selector

This gives each selector a specificity of three numbers (A, B, C). Sorted by A then B then C and if all numbers match then the selectors have the same specificity and the next step in the cascade, ordering, is used to determine what is applied.

Cascade - 6. Specificity Counting Example

p = (0, 0, 1)

div = (0, 0, 1)

div p = (0, 0, 2)

#one = (1, 0, 0)

div.someClass = (0, 1, 1)

a:visited = (0, 1, 1)

Ranked:

#one = (1, 0, 0)

div.someClass = (0, 1, 1) tied a:visited = (0, 1, 1)

div p = (0, 0, 2)

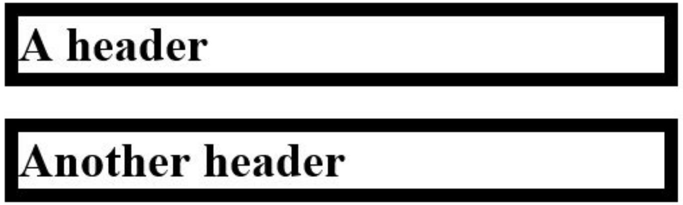
p = (0, 0, 1) tied div = (0, 0, 1)

Sidenote: (0, 1, 0) is higher than (0, 0, 3)

Cascade - 6. Specificity Example

```
<body>
    <div id="main" class="container">
      <h1 class="a-class">A header</h1>
    </div>
    <h1>Another header</h1>
</body>
```

```
body {
     width: 500px;
}

/* (0, 0, 1) */
h1 {
     border: 10px solid black;
}
```

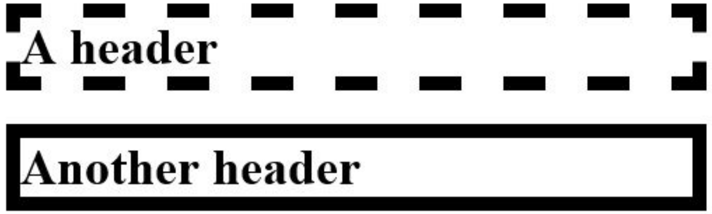

A header

Another header

Cascade - 6. Specificity Example

```
<body>
    <div id="main" class="container">
      <h1 class="a-class">A header</h1>
    </div>
    <h1>Another header</h1>
</body>
```

```
body {
     width: 500px;
}

/* (0, 0, 1) */
h1 {
     border: 10px solid black;
}

/* (0, 1, 0) */
.a-class {
     border: 10px dashed black;
}
```

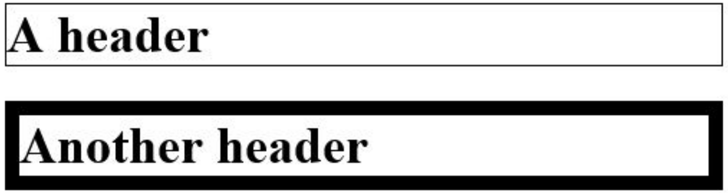


Cascade - 6. Specificity Example

```
<body>
    <div id="main" class="container">
      <h1 class="a-class">A header</h1>
    </div>
    <h1>Another header</h1>
</body>
```

```
body {
     width: 500px;
}

/* (1, 1, 2) */
div#main > h1.a-class {
     border: 1px solid black;
}

/* (0, 0, 1) */
h1 {
     border: 10px solid black;
}
```

```
/* (0, 1, 0) */
.a-class {
     border: 10px dashed black;
}
```


A header

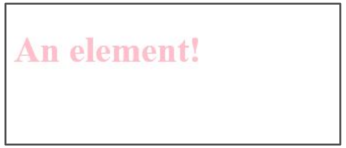
Another header

Cascade - 7. CSS Ordering on HTML Page

```html
<!DOCTYPE html>
<html>
    <head>
      <link rel="stylesheet" href="first.css">
      <link rel="stylesheet"
href="second.css">
     </head>
<body>
    <h1>An element!</h1>
    <link rel="stylesheet" href="third.css">
</body>
</html>
```

```css
/* third.css */
h1 {
    color: pink;
}
```

Output


An element!

```css
/* first.css */
h1 {
    color: red;
}
```

```css
/* second.css */
h1 {
    color: purple;
}
```

Things to remember in the cascade

- !important
- In-line style attributes on HTML elements
- JavaScript changing styles after CSS has been applied (not part of the cascade)

Keep these in mind if you are seeing behavior your don't expect. For example if a rule you wrote isn't being applied the rules might be in the wrong spot or one of these other factors could be causing your rule to not be applied.

```css
/* This slide intentionally left blank. */
.empty-slide {
    display: none;
}
```

Typography

Things you can edit:

- font-family (Times New Roman vs Arial)
  - Always choose a fallback as not all devices have all fonts
    - You can specify any font name with "quotes" but if the device visiting doesn't have the same font installed then it will display in a substituted font
- You can use @font-face to specify a font that is not "default"/web safe/load from a file or external resource
  - https://fonts.google.com/
  - https://www.fontsquirrel.com/
    - Licensing is weird/be advised just because you have a font doesn't mean you can put it in your website
  - Can also just specify serif or sans-serif if you don't care

Typography

- font-size (more on this later)
- font-weight (light vs normal vs bold but specified as a number that is a multiple of 100 up to 900)
  - If you are using a specific font-family there might be a bold font face that it would be better to use instead of setting the font-weight. CSS computes and alters the font itself to achieve this change uniformly (fonts are all vectors) and that may not be the font designer's intention
- font-style (normal vs italic vs oblique)
  - Italic will swap out to correct font face oblique will do math to make the font slanted
- line-height (how much space a single line of text takes up vertically)
  - This is different and often taller than the font size
  - The recommendation is to have this be a "unitless" value of at least 1.5 so that the line height is multiplied by the font-size and works in all cases. aka 1.5 line height on 24px font is a line height of 36px
- font-kerning (space between letters)
  - If you're using a reasonable font you probably don't need to mess with this
  - This is kind of an on/off and a property of the font you can use letter-spacing and word-spacing to force this but I think this is generally what people refer to as "bad kerning"
- text-transform (uppercase/lowercase)

CSS Values / Data Types

- Integer (1, 2), number (0.4, 0.7), dimension (## unit), percentage (50%)
- Lengths
  - Absolute values
    - cm, mm, Q (Quarter-millimeters 1/40 of 1cm), in (96px), pc (1/6 of 1in), pt (1/72 of 1in), px (1/96 of 1in)
  - Relative values
    - em 1em = font size or width of the parent 1.5 is 150% of that etc
    - rem 1rem = font size or width of the root element (<html>)
    - lh 1lh = the line height of the element
    - vw 1vw = 1% of the viewport's width
    - vh 1vh = 1% of the viewport's height
    - vmin 1% of the viewport's width or height whichever is smaller
    - vmax 1% of the viewport's width or height whichever is larger

CSS Values / Data Types

- Percentages are always relative, usually to the parent's value
- Numbers without units depend on the property (opacity 0.0 - 1.0)
- Colors
  - #0000FF
  - rgb(0, 0, 255) / rgba(0, 0, 255, 1)
  - hsl(240, 100%, 50%) / hsla(240, 100%, 50%, 1)
- Calculations
  - width: calc(30vw + 500px);
- Some properties accept multiple Data Types and some properties require specification of multiple values particularly if that property is a shorthand for multiple properties

```
/* This slide intentionally left blank. */
.empty-slide {
    display: none;
}
```

Box Model + margin gotcha: display types

- HTML elements have an outer and an inner display type; outer is how the element acts relative to other elements and inner is how elements that are children/nested inside are displayed.
- HTML elements can have an outer display type of "block", "inline", or "inline block"
- Block
    - Everything that isn't inline (but examples are <h1>, <p>, <div>)
- Inline
    - <a>, <span>, <strong>, <em>
    - <img>, but this is also a "replaced element"

Box Model + margin gotcha: display types

Block vs Inline

- block breaks onto a new line; inline doesn't
- block fills its parent container; inline doesn't
- block respects width and height properties; inline doesn't
- block vertical and horizontal padding and margins display; inline only horizontal padding and margins displays

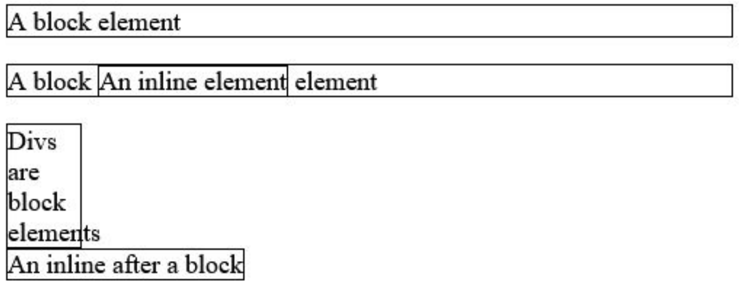
Inline-block

- doesn't start a new line
- width and height respected
- all padding/margins apply

Block vs Inline example

```
<html>
<body>
    <p>A block element</p>
    <p>A block <span>An inline element</span> element</p>
    <div>Divs are block elements</div>
    <span>An inline after a block</span>
</body>
</html>
```

```
* {
     border: 1px solid black;
}

html, body {
     border: initial;
     width: 500px;
}

div {
     width: 50px;
}
```


A block element

A block An inline element element

Divs
are
block
elements
An inline after a block

Images

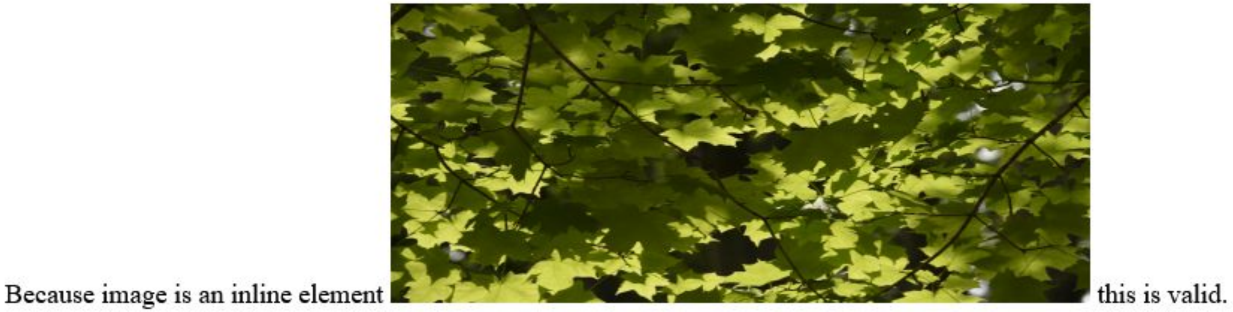
- Is an inline element
- Is a replaced element
    - Means that its width and height are dependent on the source
- Aspect ratio is important if you define both width and height in a way that is different than the aspect ratio of the source you'll get squishing of the image
- alt attribute is "required" for "valid" HTML
    - It is a great idea for fallback when the image doesn't load or for accessibility but browsers will display a lot of HTML that isn't "valid"
- You can specify the url/source of the image within CSS using the function content: url("example.jpg") as well assuming that example.jpg is in the same folder as the CSS file

Image Squishing Example

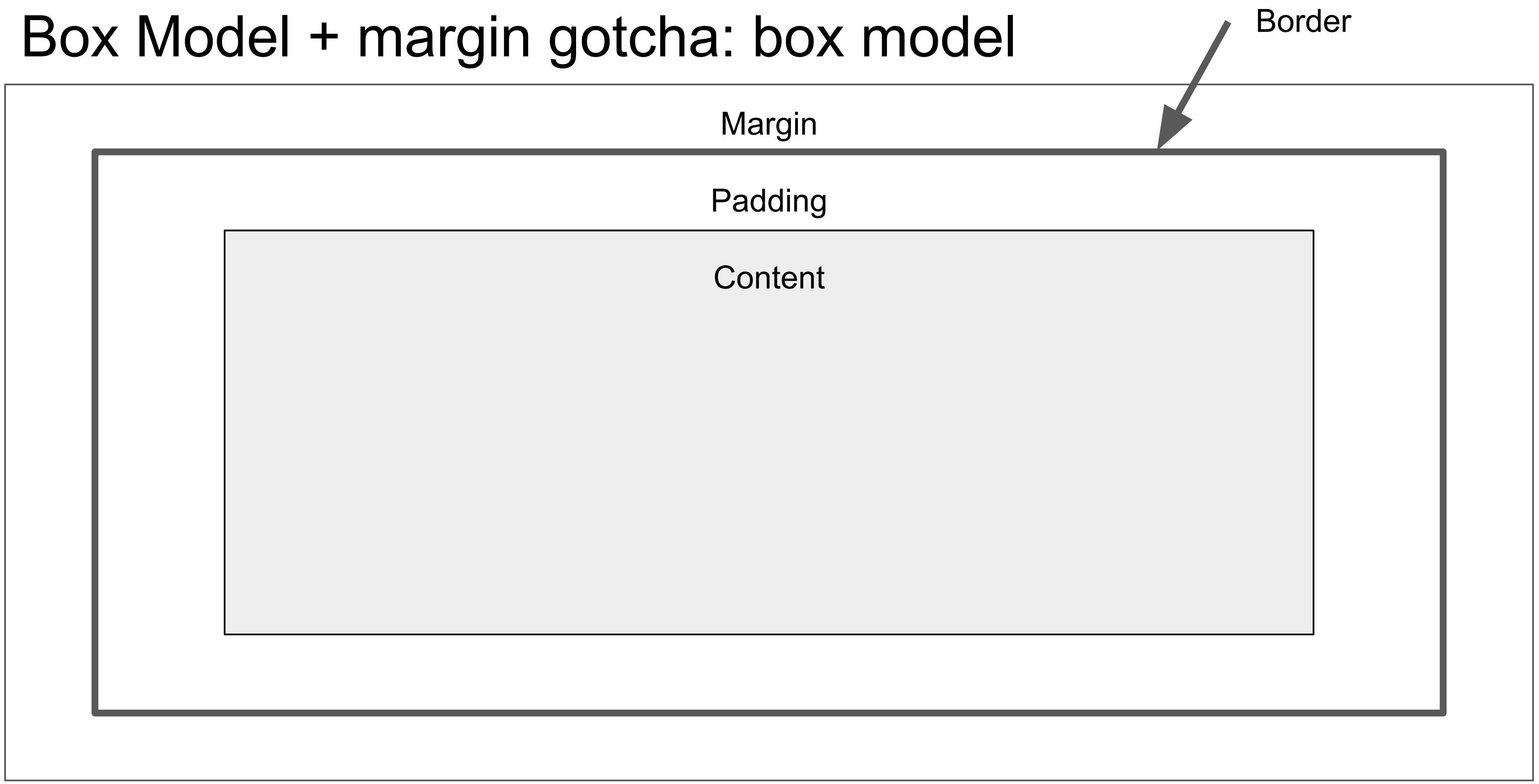
```
<body>
    <p>Because image is an inline element <img
alt="Looking up at green leaves of a tree."
src="example.jpg"> this is valid.</p>
</body>
```

```
img {
     width: 40vw;
     height: 200px;
}
```

*original aspect ratio ->


Because image is an inline element this is valid.

Box Model + margin gotcha: box model



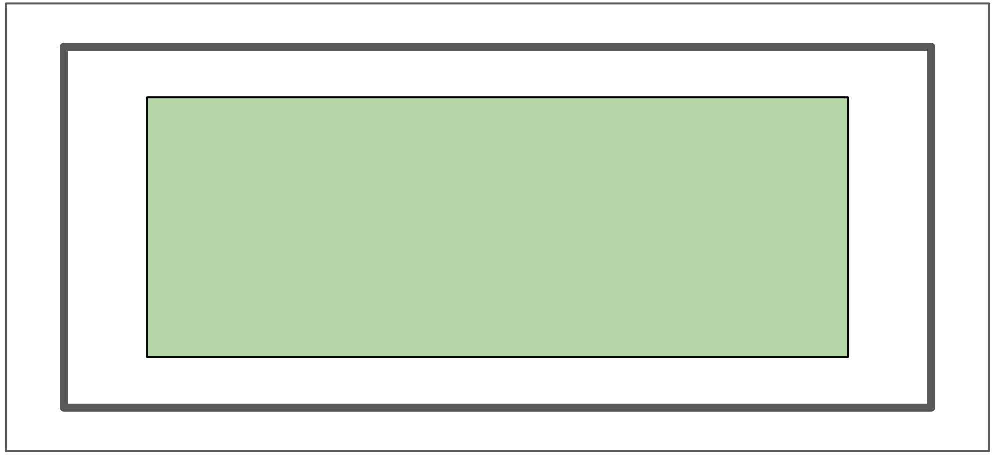
Box Model + margin gotcha: box model

- You are specifying the width and height of the *content box* when you put width and height in your CSS
  - This means that if you add padding or a border of 1px or more then the actual size the entire element takes up will be greater than the width and height you specified
  - Margin technically doesn't "count" in the final height/width of the element but it does take up space on the page

Sidebar: Box Model + margin gotcha box model alternative

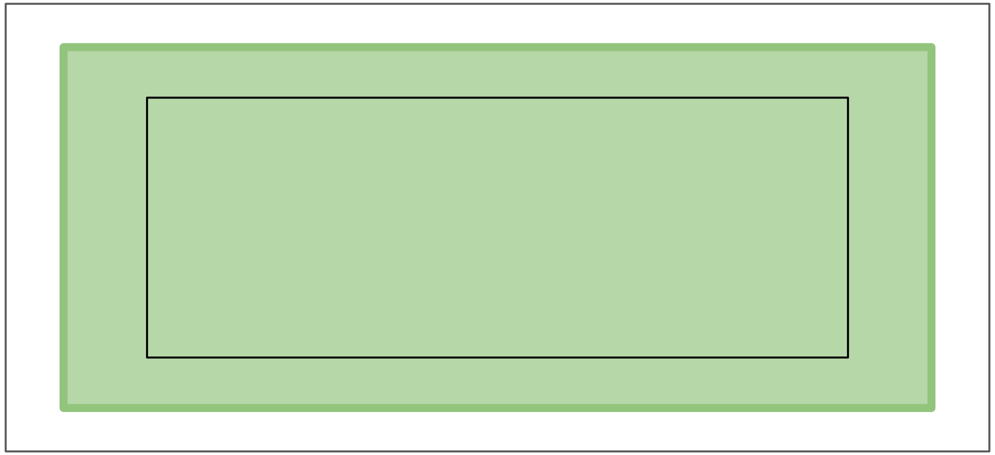
- You can specify "box-sizing: border-box;" and have the height and width you specify include the padding and the border
- Sometimes people make **all** of the elements border-box.
  - Don't make this decision lightly and it is likely a decision you'd want to make at the beginning of a project not in the middle
  - While possibly "clever" and helpful be sure to document this choice
- border-box is the default setting for the <table>, <select>, <button>, radio buttons, checkboxes, and a few other inputs

Width and height of the default box-sizing: content-box



vs

Width and height of the box-sizing: border-box

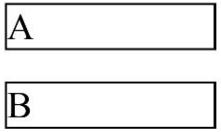


Box Model + margin gotcha: margin gotcha

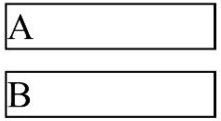
When you have two elements that are both display: block; (box-sizing either way) next to each other and they both have margins then those margins *collapse.* For example if you have two divs and html that looked like this

```
<div id="div1">A</div>

<div id="div2">B</div>

#div1 {
     margin-bottom: 10px;
     border: 1px solid black;
     width: 100px;
}

#div2 {
     margin-top: 5px;
     border: 1px solid black;
     width: 100px;
}
```

Expectation:



Reality:



Then they would display with 10px of margin in between them. NOT 15px as you might reasonably assume.

```css
/* This slide intentionally left blank. */
.empty-slide {
    display: none;
}
```

Layouts through the ages

- HTML tables
- CSS divs + floating
- Responsiveness and Media Queries
- **Flexbox**
- **CSS Grid**

HTML Tables as layout

- Pre-CSS/Initial CSS1
- Not much to say here other than this is what people did HTML tables are supposed to be for things that are "tabular data" but there weren't other options for layout other than one after the other left center or right

CSS divs and floating elements as layout

- Make containers and "float" them to the left or right so they would line up and other content can fill space surrounding them
- This is alternative to each "block" element taking up an entire line/each element breaking onto new lines
- Could make a div take 80% and then another take 20% float them both left and then you have a right sidebar
- Alternative to "floats" is "absolute positioning" you can put any element at a pixel perfect spot by saying to absolute position it and then give coordinates

Responsive Design

- Mainly a question of how do I get this layout to look good on different sized screens
- @Media queries allow you to target screens of specific sizes in CSS
- The viewport is the visible portion of the content. This can be different than the device screen size. Initially smaller devices would have the same viewport and then zoom in/out and pan to view things but now you can specify the viewport size with a meta tag in HTML:
    - <meta name="viewport" content="width=device-width, initial-scale=1">
- Testing and validation is more complicated as there are more screens sizes and edge cases to test

Media Queries Examples

The following would apply the rules inside the brackets only if the screen is larger than 576px wide.

```
@media (min-width: 576px) {

    some-selector { some-property: some-value; }

}
```

Sizes can be chosen by you and combined with other qualifications:

```
@media (max-width: 768px) and (orientation: landscape) { }

@media (min-height: 680px) { }
```

You can also query for devices that are printed, read by screen readers, or displays that are monochrome.

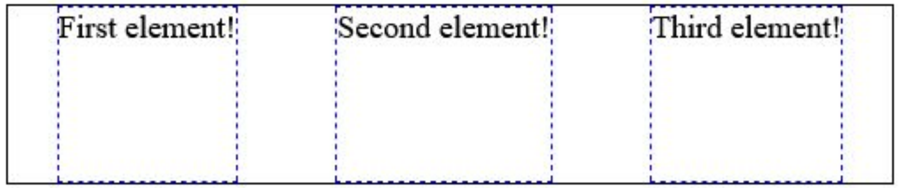
Flexbox

- Useful for vertically or horizontally spacing items so that each item takes up the same amount of space especially when the amount of space available isn't known in advance (responsive)
- Make multiple columns the same height
- Recommended for components and not the layout of the entire page
- You can now order elements with CSS that are not the same as the DOM order!!
- Flex is an "inner display" so it affects an element's children when applied
    - specifically *all descendants* are affected making it not great for entire pages

Flexbox Example

```
<body>
    <div class="flex-parent">
      <div class="flex-item">First element!</div>
      <div class="flex-item">Second element!</div>
      <div class="flex-item">Third element!</div>
    </div>
</body>
```

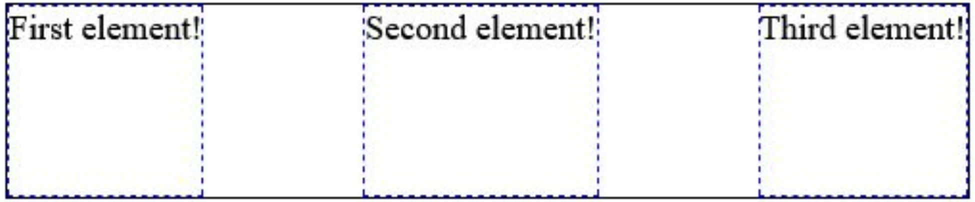
```
.flex-parent {
     display: flex;
     flex-direction: row;
     border: 1px solid black;
     width: 500px;
     height: 100px;
     justify-content: space-around;
}

.flex-item {
     border: 1px dashed blue;
}
```



Flexbox Example

```
<body>
    <div class="flex-parent">
      <div class="flex-item">First element!</div>
      <div class="flex-item">Second element!</div>
      <div class="flex-item">Third element!</div>
    </div>
</body>
```

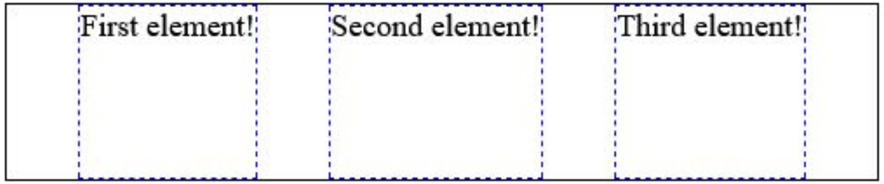
```
.flex-parent {
     display: flex;
     flex-direction: row;
     border: 1px solid black;
     width: 500px;
     height: 100px;
     justify-content: space-between;
}

.flex-item {
     border: 1px dashed blue;
}
```



Flexbox Example

```
<body>
    <div class="flex-parent">
      <div class="flex-item">First element!</div>
      <div class="flex-item">Second element!</div>
      <div class="flex-item">Third element!</div>
    </div>
</body>
```

```
.flex-parent {
     display: flex;
     flex-direction: row;
     border: 1px solid black;
     width: 500px;
     height: 100px;
     justify-content: space-evenly;
}

.flex-item {
     border: 1px dashed blue;
}
```

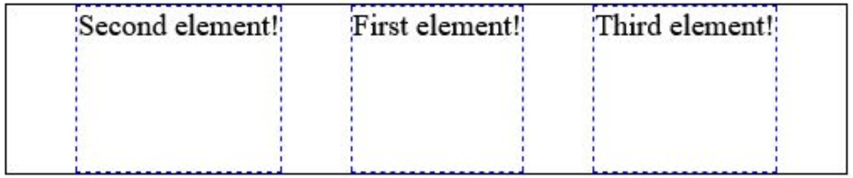

First element! Second element! Third element!

Flexbox Example

```
<body>
    <div class="flex-parent">
      <div class="flex-item">First element!</div>
      <div class="flex-item most-important">Second element!</div>
      <div class="flex-item">Third element!</div>
    </div>
</body>
```

```
.flex-parent {
     display: flex;
     flex-direction: row;
     border: 1px solid black;
     width: 500px;
     height: 100px;
     justify-content: space-evenly;
}

.flex-item {
     border: 1px dashed blue;
     order: 2;
}
```

```
.most-important {
     order: 1;
}
```

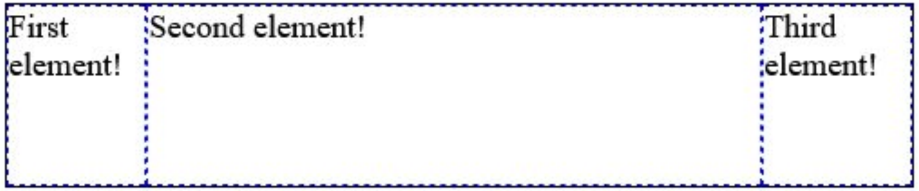


Flexbox Example

```
<body>
    <div class="flex-parent">
      <div class="flex-item">First element!</div>
      <div class="flex-item most-important">Second element!</div>
      <div class="flex-item">Third element!</div>
    </div>
</body>
```

```
.flex-parent {
     display: flex;
     flex-direction: row;
     border: 1px solid black;
     width: 500px;
     height: 100px;
     justify-content: space-evenly;
}

.flex-item {
     border: 1px dashed blue;
}
```

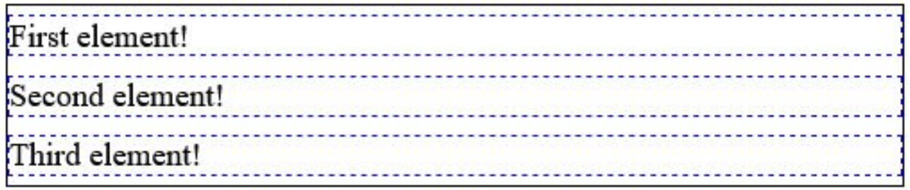
```
.most-important {
     flex-basis: 90%;
}
```



Flexbox Example

```
<body>
    <div class="flex-parent">
      <div class="flex-item">First element!</div>
      <div class="flex-item">Second element!</div>
      <div class="flex-item">Third element!</div>
    </div>
</body>
```

```
.flex-parent {
     display: flex;
     flex-direction: column;
     border: 1px solid black;
     width: 500px;
     height: 100px;
     justify-content: space-around;
}

.flex-item {
     border: 1px dashed blue;
}
```


First element!
Second element!
Third element!

CSS Grid

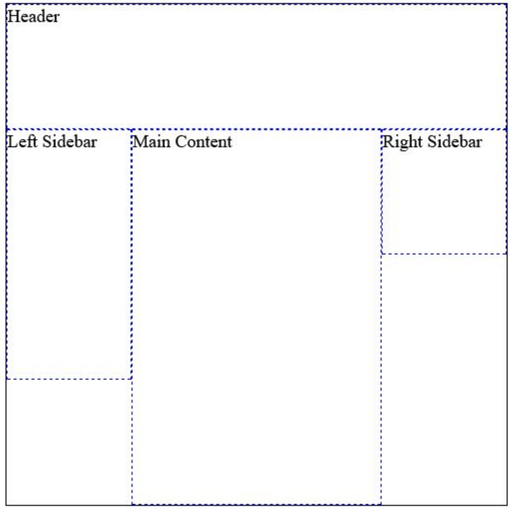
- Is recommended for full page layout
- You define the number of rows, columns, and their sizes of the grid
- Then each child defines which of the rows/columns to take up and how to align itself within that area
- Grid is an "inner display" so it affects an element's children when applied
- Made responsive via media queries

CSS Grid Example

```
<body>
    <div class="grid-parent">
       <div class="grid-item grid-item-header">Header</div>
       <div class="grid-item grid-item-left">Left Sidebar</div>
       <div class="grid-item grid-item-right">Right Sidebar</div>
       <div class="grid-item grid-item-main">Main Content</div>
    </div>
</body>
```

```
.grid-parent {
      display: grid;
      grid-template-columns: 25% 50% 25%;
      grid-template-rows: 25% 25% 25% 25%;
      border: 1px solid black;
      width: 500px;
      height: 500px;
}

.grid-item {
      border: 1px dashed blue;
}

.grid-item-header {
      grid-column: 1 / span 3;
      grid-row: 1 / 1;
}
```

```
.grid-item-left {
      grid-column: 1 / 1;
      grid-row: 2 / span 2;
}

.grid-item-main {
      grid-column: 2 / span 1;
      grid-row: 2 / span 3;
}

.grid-item-right {
      grid-column: 3 / span 1;
      grid-row: 2 / span 1;
}
```

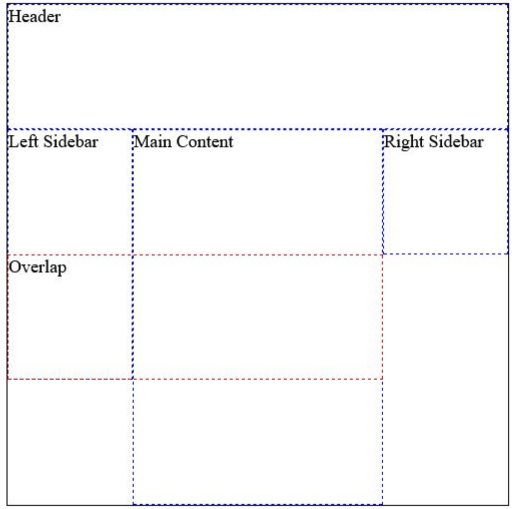


CSS Grid Example

```
<body>
    <div class="grid-parent">
       <div class="grid-item grid-item-header">Header</div>
       <div class="grid-item grid-item-left">Left Sidebar</div>
       <div class="grid-item grid-item-right">Right Sidebar</div>
       <div class="grid-item grid-item-main">Main Content</div>
       <div class="grid-item grid-item-overlap">Overlap</div>
    </div>
</body>
```

```
.grid-parent {
      display: grid;
      grid-template-columns: 25% 50% 25%;
      grid-template-rows: 25% 25% 25% 25%;
      border: 1px solid black;
      width: 500px;
      height: 500px;
}

.grid-item {
      border: 1px dashed blue;
}

.grid-item-header {
      grid-column: 1 / span 3;
      grid-row: 1 / 1;
}
```

```
.grid-item-left {
      grid-column: 1 / 1;
      grid-row: 2 / span 2;
}

.grid-item-main {
      grid-column: 2 / span 1;
      grid-row: 2 / span 3;
}

.grid-item-right {
      grid-column: 3 / span 1;
      grid-row: 2 / span 1;
}

.grid-item-overlap {
      grid-column: 1 / span 2;
      grid-row: 3 / span 1;
      border-color: red;
}
```



*the columns and rows can also have names specified if you don't like the numbers

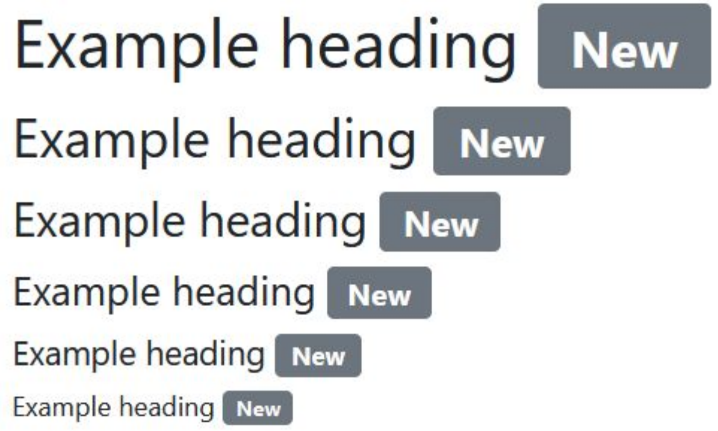
```
/* This slide intentionally left blank. */
.empty-slide {
    display: none;
}
```

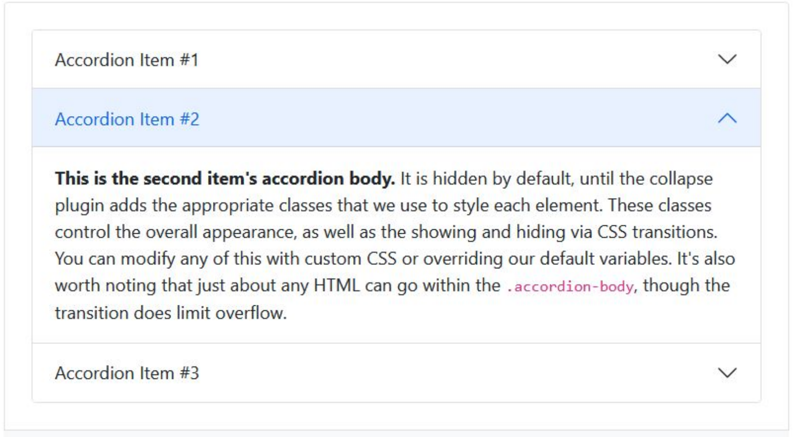
CSS "Frameworks"

1. Bootstrap
2. Foundations
   a. Tailwind CSS
3. Other people's CSS surrounding yours
4. JavaScript Plugins/Libraries

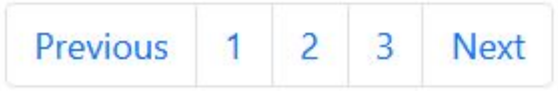
Bootstrap

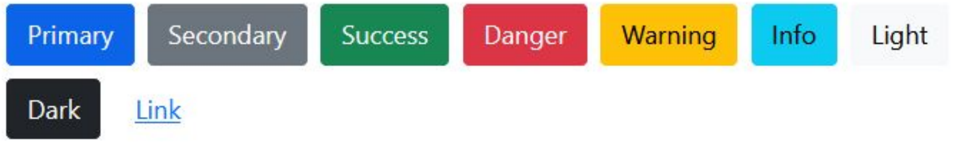
- Provides utilities, components, and a grid



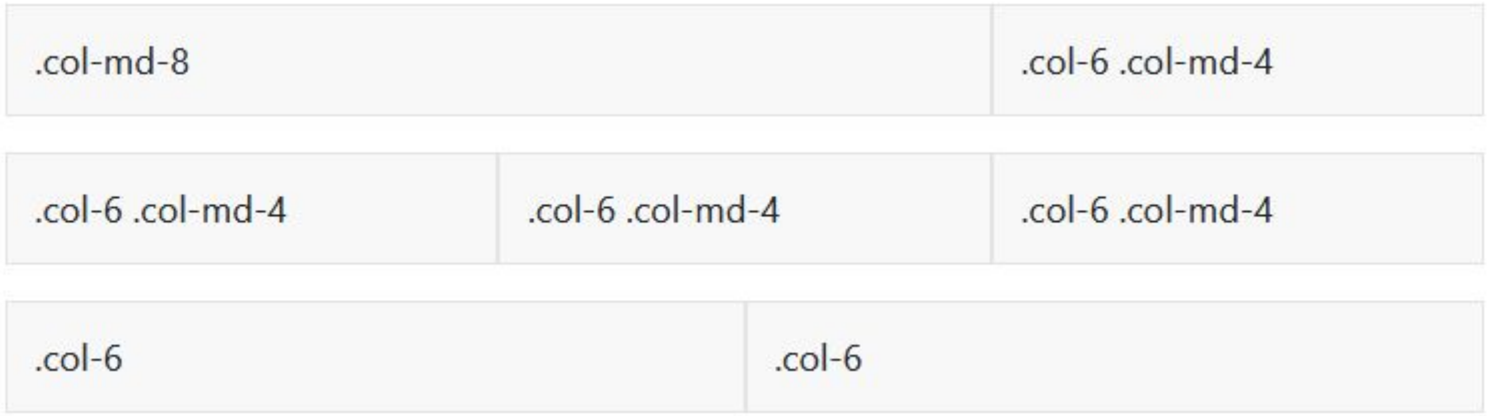


This is the second item's accordion body. It is hidden by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the `.accordion-body`, though the transition does limit overflow.





Bootstrap Grid

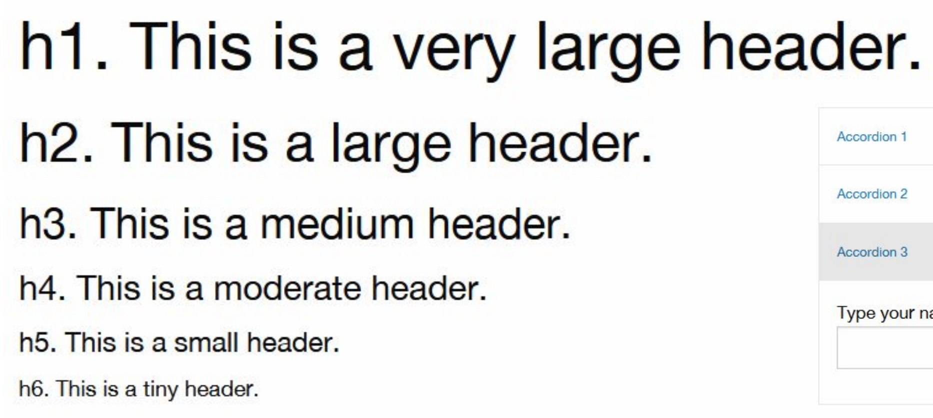


Bootstrap Grid

```
<div class="container">
  <!-- Stack the columns on mobile by making one full-width and the other half-width -->
  <div class="row">
      <div class="col-md-8">.col-md-8</div>
      <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>

  <!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop -->
  <div class="row">
      <div class="col-6 col-md-4">.col-6 .col-md-4</div>
      <div class="col-6 col-md-4">.col-6 .col-md-4</div>
      <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>

  <!-- Columns are always 50% wide, on mobile and desktop -->
  <div class="row">
      <div class="col-6">.col-6</div>
      <div class="col-6">.col-6</div>
  </div>
</div>
```

Foundations

- Provides utilities, components, and a grid


h1. This is a very large header.

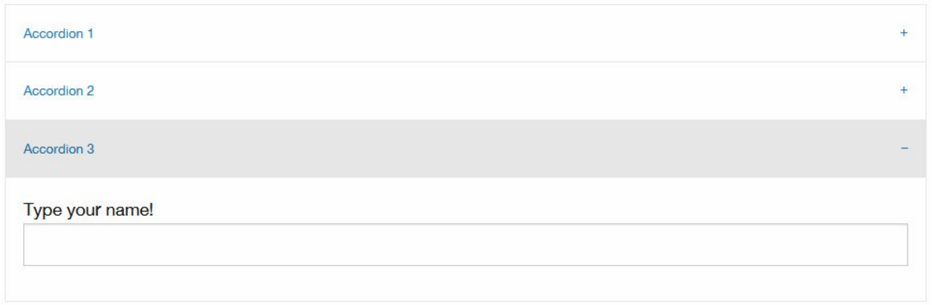
h2. This is a large header.

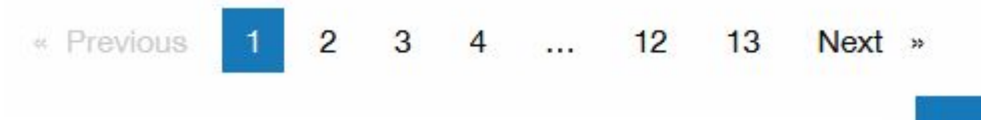
h3. This is a medium header.

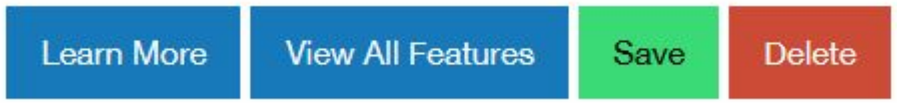
h4. This is a moderate header.

h5. This is a small header.

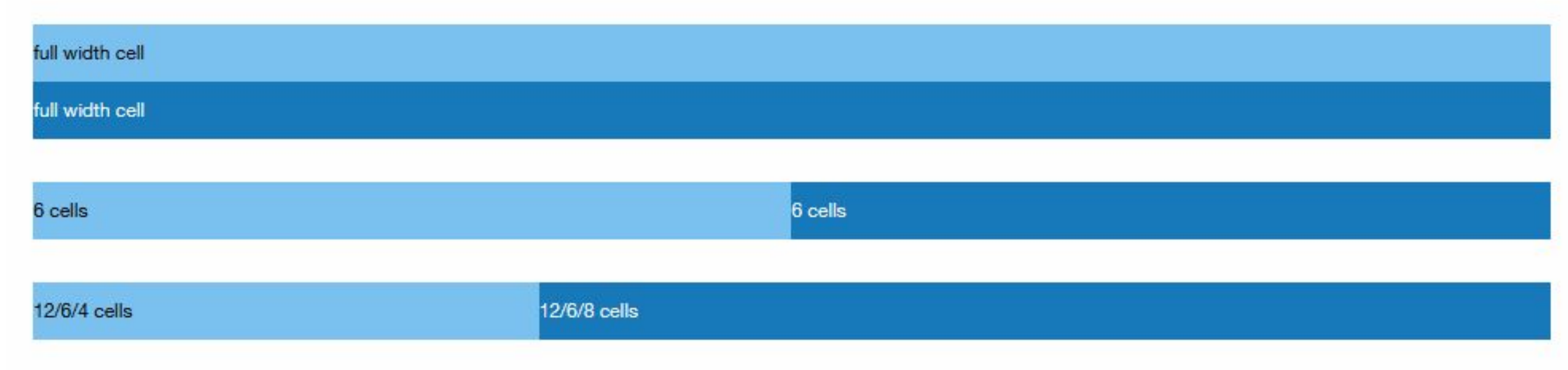
h6. This is a tiny header.







Foundations Grid



Foundations Grid

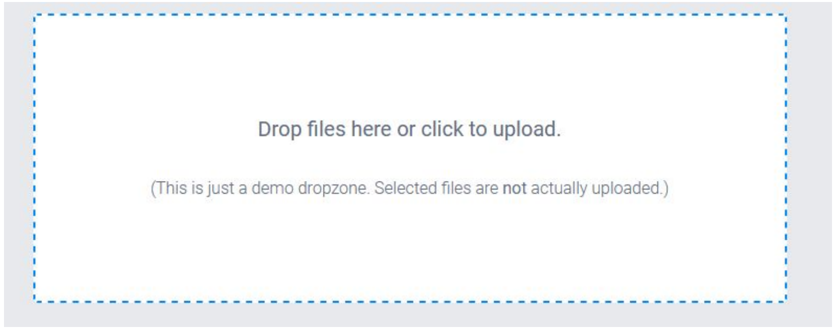
```
<div class="grid-x">
  <div class="cell">full width cell</div>
  <div class="cell">full width cell</div>
</div>
<div class="grid-x">
  <div class="cell small-6">6 cells</div>
  <div class="cell small-6">6 cells</div>
</div>
<div class="grid-x">
  <div class="cell medium-6 large-4">12/6/4 cells</div>
  <div class="cell medium-6 large-8">12/6/8 cells</div>
</div>
```

Framework Differences

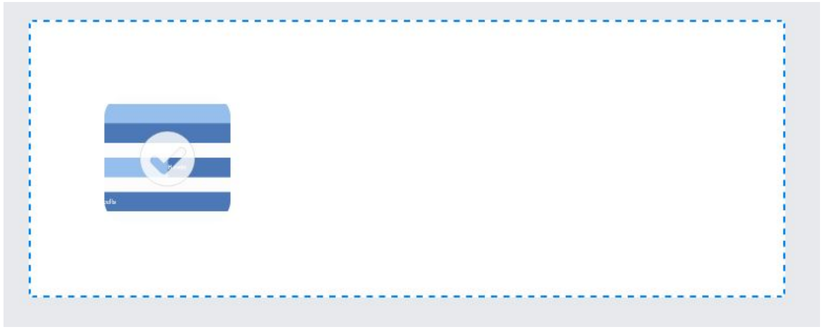
- Class names
- Number of class
- Which elements the class needs to be added on
- Default styling
- CSS Pre-processor/Compiler both use SASS
  - Bootstrap used to use Less
- Flexibility?
  - The internet informs me that foundations is more flexible with the tradeoff of being "more complicated."
- Popularity?
  - Bootstrap is supported by twitter and much much more popular/used
- Browser Support
  - Bootstrap IE 11+ (only with v4 and below) and Foundations IE9+ (as of July 2021)
- box-sizing
  - Bootstrap is border-box by default Foundations appears to use content-box with a utility to set border-box

JS/CSS Plugin Example: Dropzone.js

- It is a JS library for drag and drop file uploading
- You add it to your site by including some JS and some CSS
  - <form action="/file-upload" class="dropzone" id="my-awesome-dropzone"></form>
- Then it has some default styling for the place to upload files, the text, and the success styling


Drop files here or click to upload.

(This is just a demo dropzone. Selected files are not actually uploaded.)



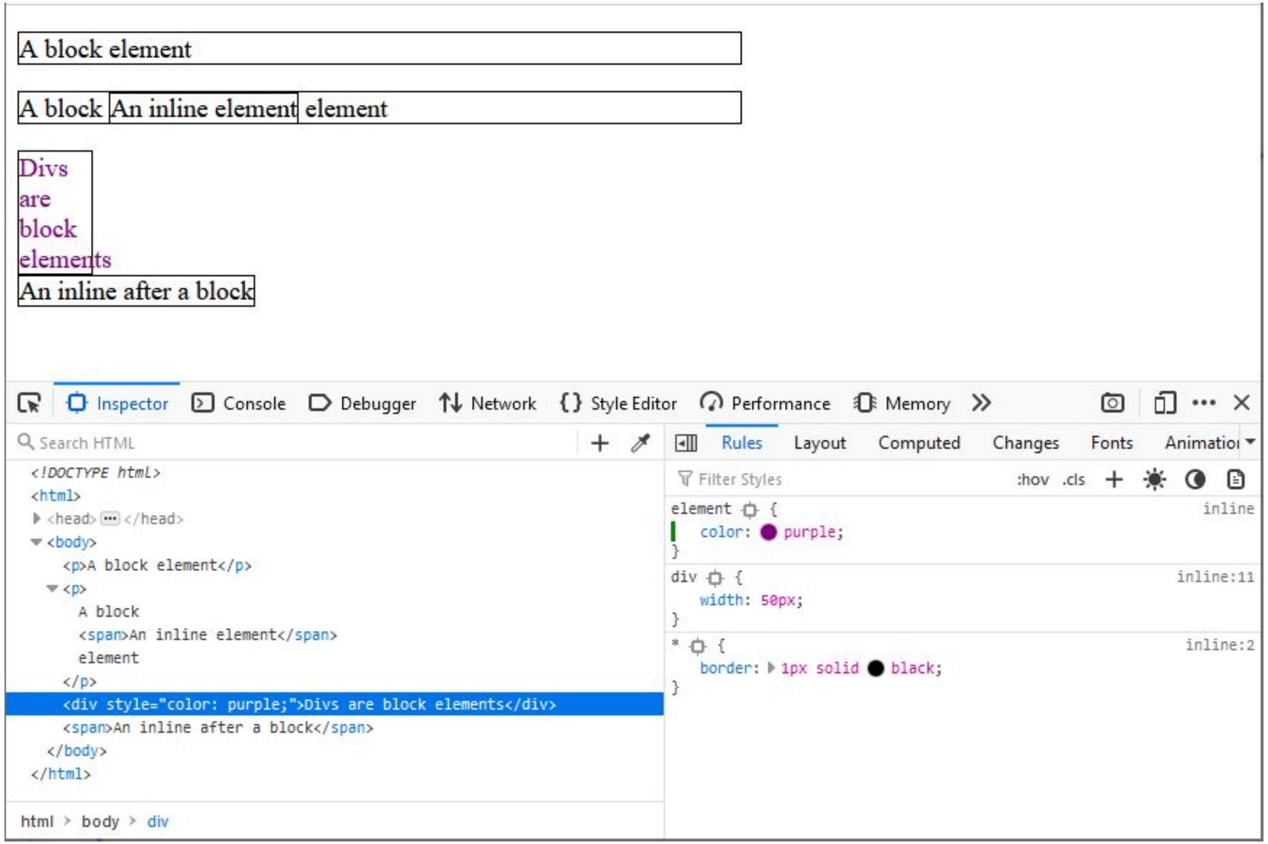
Opinions/prefixes/namespaces

- While scaling out CSS one of the main issues people run into is consistency in naming
- How to tell where a class can be used or what it is for by its name?
  - Reusable by its name?
  - Semantic vs descriptive/presentation (.promotion-pop-up vs .overlay or .product-listing vs .tile)
- In JavaScript properties are camel case so one common pattern is having CSS class names be hyphenated as this matches css properties and values
  - With debates on if the IDs should be similarly hyphenated if they are for CSS only
- Separate ids/classes for your JavaScript hooks and CSS might avoid breaking either
- There is the "BEM" or Block Element Modifiers for example `.nav__secondary--blue`
- Prefixing a class for .t- for theme related rules or .u- for utilities
- If you are using a framework for 99% of things and you write only a few classes maybe you prefix your classes with something like .project-name-class so people know it is custom
  - Plugins may prefix their classes with .plugin-name to avoid conflicts as well

Takeaways

- Important questions
  - Where will the CSS you are writing be loaded?
  - What other CSS surrounds what you are working on?
- Test your CSS in all browsers and screen sizes you are supporting

Browser Inspection Tools



A block element

A block An inline element element

Divs are block elements

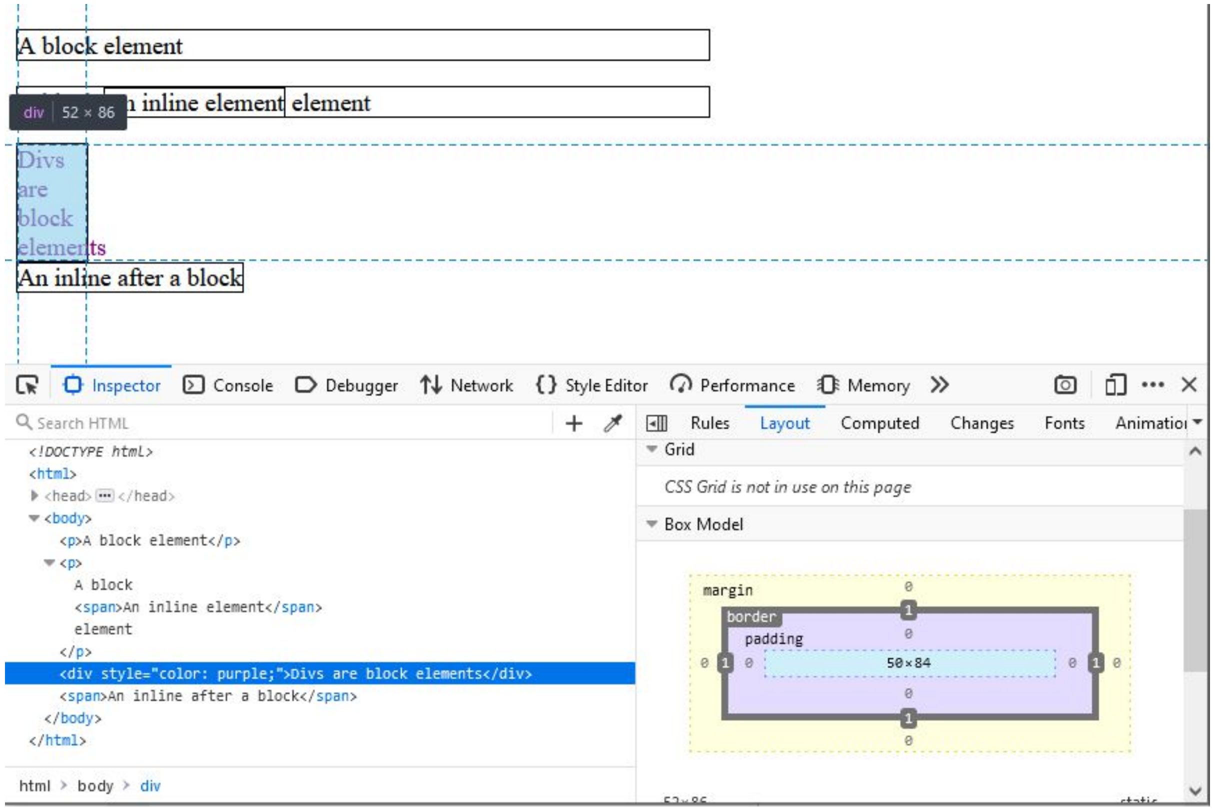
An inline after a block

```
<!DOCTYPE html>
<html>
  <head>…</head>
  <body>
    <p>A block element</p>
    <p>
      A block
      <span>An inline element</span>
      element
    </p>
    <div style="color: purple;">Divs are block elements</div>
    <span>An inline after a block</span>
  </body>
</html>
```

```
element {                                   inline
  color: ● purple;
}
div {                                       inline:11
  width: 50px;
}
* {                                         inline:2
  border: 1px solid ● black;
}
```

html > body > div

Browser Inspection Tools



A block element

A block An inline element element

Divs are block elements

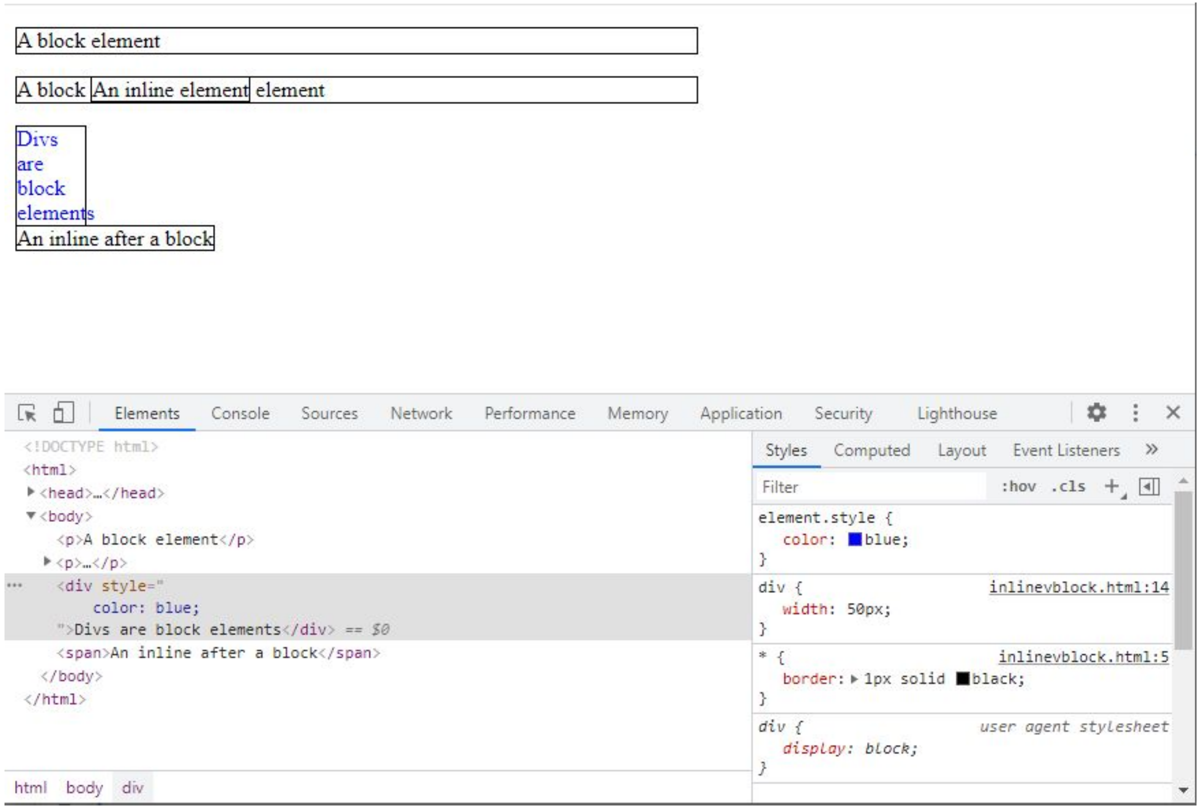
An inline after a block

```
<!DOCTYPE html>
<html>
 ▸ <head>…</head>
 ▾ <body>
    <p>A block element</p>
  ▾ <p>
      A block
      <span>An inline element</span>
      element
    </p>
    <div style="color: purple;">Divs are block elements</div>
    <span>An inline after a block</span>
  </body>
</html>
```

CSS Grid is not in use on this page

```
margin            0
  border          1
    padding       0
0 1 0     50×84     0 1 0
                  0
                  1
                  0
```

Browser Inspection Tools



A block element

A block An inline element element

Divs
are
block
elements

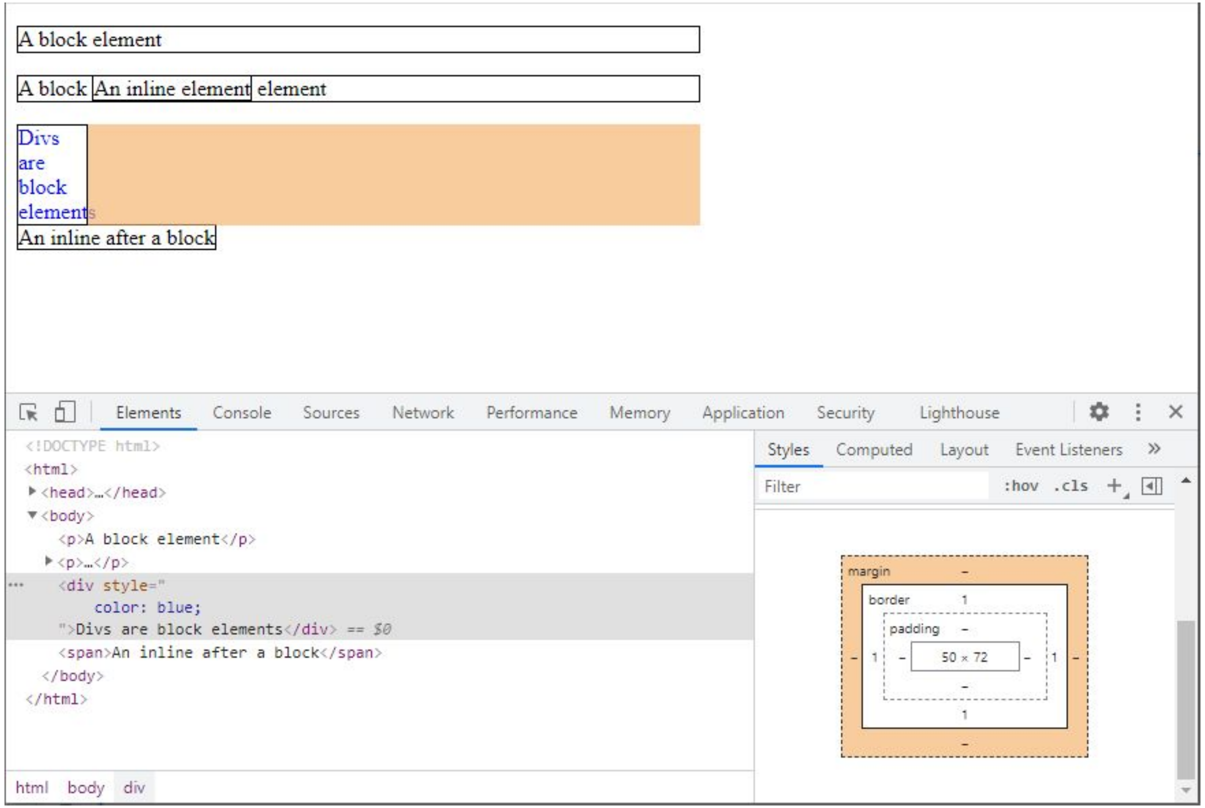
An inline after a block

```
<!DOCTYPE html>
<html>
  ▶<head>…</head>
  ▼<body>
      <p>A block element</p>
    ▶<p>…</p>
      <div style="
          color: blue;
      ">Divs are block elements</div> == $0
      <span>An inline after a block</span>
    </body>
</html>
```

```
element.style {
  color: blue;
}
div {                          inlinevblock.html:14
  width: 50px;
}
* {                            inlinevblock.html:5
  border: ▶ 1px solid black;
}
div {                          user agent stylesheet
  display: block;
}
```

html body div

Browser Inspection Tools



A block element

A block An inline element element

Divs are block elements

An inline after a block

```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body>
    <p>A block element</p>
   ▶<p>…</p>
    <div style="
        color: blue;
    ">Divs are block elements</div> == $0
    <span>An inline after a block</span>
  </body>
</html>
```

Code "smells"

- You have !important
  - Don't have !important
- You have CSS that is "append only"
- The less CSS you have the generally better off you are
  - If you find yourself making new CSS classes every time you do something maybe you can try to write a few classes to reuse (applies slightly less in the beginning)
- It is easy to be too specific; for example using IDs for everything is similar to the problem above where CSS is only for one thing

Things to keep in mind

- JavaScript is changing the styles after they load
- Hard coded heights and widths ruin your responsive efforts
- Unsupported properties or values
    - You might be using CSS that isn't supported yet in some browsers
    - Some properties or new features have browser-prefixes where you specify the property multiple times to ensure cross browser similarity
    - If you want to use a specific CSS selector or property go to https://caniuse.com/ and compare the browser compatibility chart with your needs
- Look-up properties and their allowed values paying attention to units. I recommend: https://developer.mozilla.org/en-US/docs/Web/CSS3
- Accessibility
    - For example some of my examples only make distinctions on color this is not recommended

Other things of possible interest

- There are two major "compiled" CSS options (more like pre-processors)
    - You'll note that since CSS is just a file you serve it up and it just works you don't have to "do" anything to it
    - However if you want things like variables or other fancy features you can write CSS in SASS/SCSS or Less
- You can "minify" CSS to make it smaller on page loads
- You can animate things with CSS
    - Its near the top of the cascade above normal author declarations
- Throw a border around everything can sometimes be a debugging tool
- CSS matters in i18n. Directions and layouts often swap between languages
- Many ways to handle custom themes but a common one is to separate out theme relevant selectors and make multiple style sheets only loading the relevant one via JavaScript or server side decisions

Conclusion

- You can do this
- Thank you

General Resources

- https://css-tricks.com/
  - Generally if you have an idea of what you want something to look like this is a cookbook
- https://developer.mozilla.org/en-US/docs/Learn/CSS
- https://codepen.io/pen/
  - Quick testing in-browser of your elements without having to save out files to your desktop etc
- A more complete list of CSS Selectors: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors
- http://www.csszengarden.com/
  - It is the same markup styled by different people
- https://www.awwwards.com/
- https://wattenberger.com/blog/css-percents
- https://www.w3.org/Style/CSS20/history.html
- https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties

References

- CSS Syntax Really Cool Example: https://codepen.io/marcobiedermann/full/osurh
- Property List
  - https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Properties_Reference
- The cascade 2020 Snapshot: https://www.w3.org/TR/css-cascade-4/#cascade-sort
- Specificity: https://www.w3.org/TR/selectors/#specificity
- Box Model: https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model
- Bootstrap Screenshots/Code: https://getbootstrap.com/
  - https://getbootstrap.com/docs/5.0/layout/grid/

References

- Foundation Screenshots/Code: https://get.foundation/index.html
    - https://get.foundation/sites/docs/xy-grid.html
- Flexbox: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- CSS Grid: https://css-tricks.com/snippets/css/complete-guide-grid/

Author and Contact Info

Slides produced July 2021

Sarah Whelan

hello@sarahwhelan.com

Please reach out with any corrections, questions, or comments. They will be greatly appreciated.