Basics of CSS and Rendering

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8"/>
  <title>Example</title>
</head>
<body>
  This is a page heading
  This is a section heading
  This is a paragraph. With two sentences.
</body>
</html>
```

Viewing the Web Page

- Soon we'll learn how to run a simple webserver
- For now, we'll just open HTML file in Chrome
- Mac: File->View or `Cmd+0`
- Windows: `Ctrl-O`

This will have problems with absolute/relative links!

- But we don't have those yet

That looks awful

- "Whitespace", incl. newlines, became 1 space
- Everything is just unstyled text
- Let's add one element

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Example</title>
</head>
<body>
  This is a page heading
  This is a section heading
  <p>This is a paragraph. With two sentences.</p>
</body>
</html>
```

Marginally Better

- Still bad, but "better"
- This is the design principle of **Proximity**
    - Distance communicates relationship
- Another principle: **Whitespace**
    - Space eases comprehension
- But how is this space created?
- How does the browser decide how much space?

The CSS Box Model

- Document is a series of elements
- Each element renders as a "box":
    - The space the element takes up
    - This "box" may not itself be visible
- Document is a series of boxes
- Boxes inside boxes
- Boxes alongside boxes

Width and Height

Each element has a base **width** and **height**

- Both may default to fit the content
- Width may default to fill the parent element
    - The "container" of this element box
- Both can be set to different values

Text in a sentence is considered `inline`

- Has `width` equal to the content
- If content too long to fit in containing element
- Then the content wraps to new lines
- Has `height` based on the wrapping content needs

`inline` content ignores `width`/`height` CSS properties

Elements can be `inline`

- `inline` elements render like text in a sentence
- Being `inline` is defined by the `display` property
- Let's add an `inline` element, like a link:

```
<body>
  This is a page heading
  This is a section heading
  This is a paragraph.
  <p>
    This is a paragraph. With two sentences.
    A third with <a href="/">a link</a> in the middle.
  </p>
</body>
```

`block` Elements

Sentence-like text is `inline`

Paragraphs are `block` elements

- `width` defaults to container width
- `height` defaults to height needed for content
- `width` and `height` can be changed in CSS
- Will always start a newline before and after
    - Breaking "text flow"

Being `inline` or `block` are **default styles**

- Can set `display` to `inline` or `block`

Seeing our Box

- We'll soon learn better tools for this
- For now, we'll use CSS
- Which means we need to add CSS

```
<head>
  <meta charset="UTF-8"/>
  <title>Example</title>
  <link rel="stylesheet" href="/styles.css"/>
</head>
<body>
  This is a page heading
  This is a section heading
  This is a paragraph.
  <p>
    This is a paragraph. With two sentences.
    A third with <a href="/">a link</a> in the middle.
  </p>
</body>
```

Starting our CSS

Let's verify it works with a simple start

`styles.css`:

```
body {
    background-color: papayawhip;
}
```

`papayawhip`? What is with these colors?

CSS color values

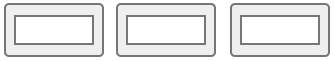
- CSS has many ways to give a color value
- Used in many places, most commonly:
  - `color` (element text color)
  - `background-color` (what it says)
- Options
  - Named Colors
  - RGB values
  - and more!

Named Color Values

```
color: white;
color: lime;
color: aqua;
color: bleachedalmond;
color: palevioletred;
color: rebeccapurple;
```

https://drafts.csswg.org/css-color/#named-colors

Original colors came from different sources

- A few weird results
- `dimgray` darker than `gray`
- `gray` darker than `darkgray`
- 

A recent named color

List of named colors not expanding

- Too many colors, names get complex
- One notable exception

Eric Meyer

- Early and influential web developer
- His daughter Becca passed away
- Community honored them with her favorite color

Was to be `beccapurple`, but Eric set one condition

Not `beccapurple`

...that if the proposal is accepted, the official name be `rebeccapurple`*. A couple of weeks before she died, Rebecca informed us that she was about to be a big girl of six years old, and Becca was a baby name. Once she turned six, she wanted everyone (not just me) to call her Rebecca, not Becca.*

She made it to six. For almost twelve hours, she was six. So Rebecca it is and must be.

- 2014 - `rebeccapurple` accepted as a standard color
- 2024 - CSS logo created, white on rebeccapurple

RGB (Red Green Blue) values

- **hexadecimal**
  - (hex) is a base 16 number, shown as 0-F
- Hex pair (00-FF) is (0-255), or an 8-bit value
- CSS color can a "hexadecimal RGB value"
- A `#` followed by 3 hex pairs
  - not `id` related, not url hash related

Examples:

- `color: #BADA55;`
- `color: #336699;` (`rebeccapurple`!)
- `color: #C0FFEE;`

RGB Variations

- single characters (not pairs), treated as doubled
  - `#639` is `#663399`
- Programmers are lazy
  - A fourth character or pair is "alpha"
- Transparency
- `rgb()` or `rgba()` passing 3 RGB vals and an alpha
  - passed RGB values are decimal
  - alpha is 0-1 or 0%-100%
- `color: rgb(102, 51, 153);` (`rebeccapurple`)
- `color: rgb(192, 255, 238);` (`#C0FFEE`)

More Color Systems

- HSL (Hue, Saturation, Light)
- HWB (Hue White Black)
- LAB
- LCH
- Oklab
- Oklch

I don't even know how some of these work (yet!)

- Excited for when `color-mix()` is widely available!

Using Colors to Expose Our Element Boxes

```
body {
  background-color: papayawhip;
}

p {
  background-color: #C0FFEE;
}

a {
  background-color: burlywood;
}
```

- Our `inline` element is content-sized
- Our `block` element fills width
    - Exceeds width of content

Let's play with the `height`

Changing some `heights`

```
body {
  background-color: papayawhip;
}

p {
  height: 50px;
  background-color: #C0FFEE;
}

a {
  height: 50px;
  background-color: burlywood;
}
```

- `block` element changed height
- `inline` element did not
    - `inline` elements ignore `height`/`width`

Adding Some Lists

```
<body>
  This is a page heading
  This is a section heading
  This is a paragraph.
  <p>
    This is a paragraph. With two sentences.
    A third with <a href="/">a link</a> in the middle.
  </p>
  <ul>
    <li>Tiger</li>
    <li>Jaguar</li>
  </ul>
  <ul>
    <li>Jorts</li>
    <li>Maru</li>
  </ul>
</body>
```

Need Some Class

Distinguish our lists with different colors

- Add different `class` names to use in selectors

```html
<ul class="big-cats">
  <li>Tiger</li>
  <li>Jaguar</li>
</ul>
<ul class="chonky-cats">
  <li>Jorts</li>
  <li>Maru</li>
</ul>
```

```css
.big-cats {
  background-color: lavender;
}

.chonky-cats {
  background-color: plum;
}
```

Conclusions and Questions

- `ul` is a `block` element
- But what is the space between the elements?
- And the dot/left side?
- Why don't the colors quite fill the body?
- Can we put the lists NEXT to each other?

Narrowing the lists

- `width` alone won't align the lists

```
.big-cats {
  width: 100px;
  background-color: lavender;
}

.chonky-cats {
  width: 100px;
  background-color: plum;
}
```

Block elements force a break

`display: inline-block;`

- Can change content width/height, like `block`
- Doesn't break flow, like `inline`

```
.big-cats {
  display: inline-block;
  width: 100px;
  background-color: lavender;
}

.chonky-cats {
  display: inline-block;
  width: 100px;
  background-color: plum;
}
```

Almost Perfect

Notice the small space between the lists?

- That's the whitespace between the elements
- Rendered as a single space

Yuck! We don't want to remove that from our HTML

- More Next Class!

Padding

`padding` is the space around the content

- between content and edge of `background-color`

```
.chonky-cats {
  display: inline-block;
  width: 100px;
  padding: 16px;
  background-color: plum;
}
```

Why are items CLOSER to the left side?

- Unlike `width`/`height` not just one `padding`
- One for each of the four sides
    - TRBL (Trouble): Top, Right, Bottom, Left
- `padding: 16px;` is a **shorthand property**
    - Sets multiple values
    - Here: `padding-top`, `padding-right`, etc.

Using DevTools to View Information

Let's use DevTools to learn more of what is going on

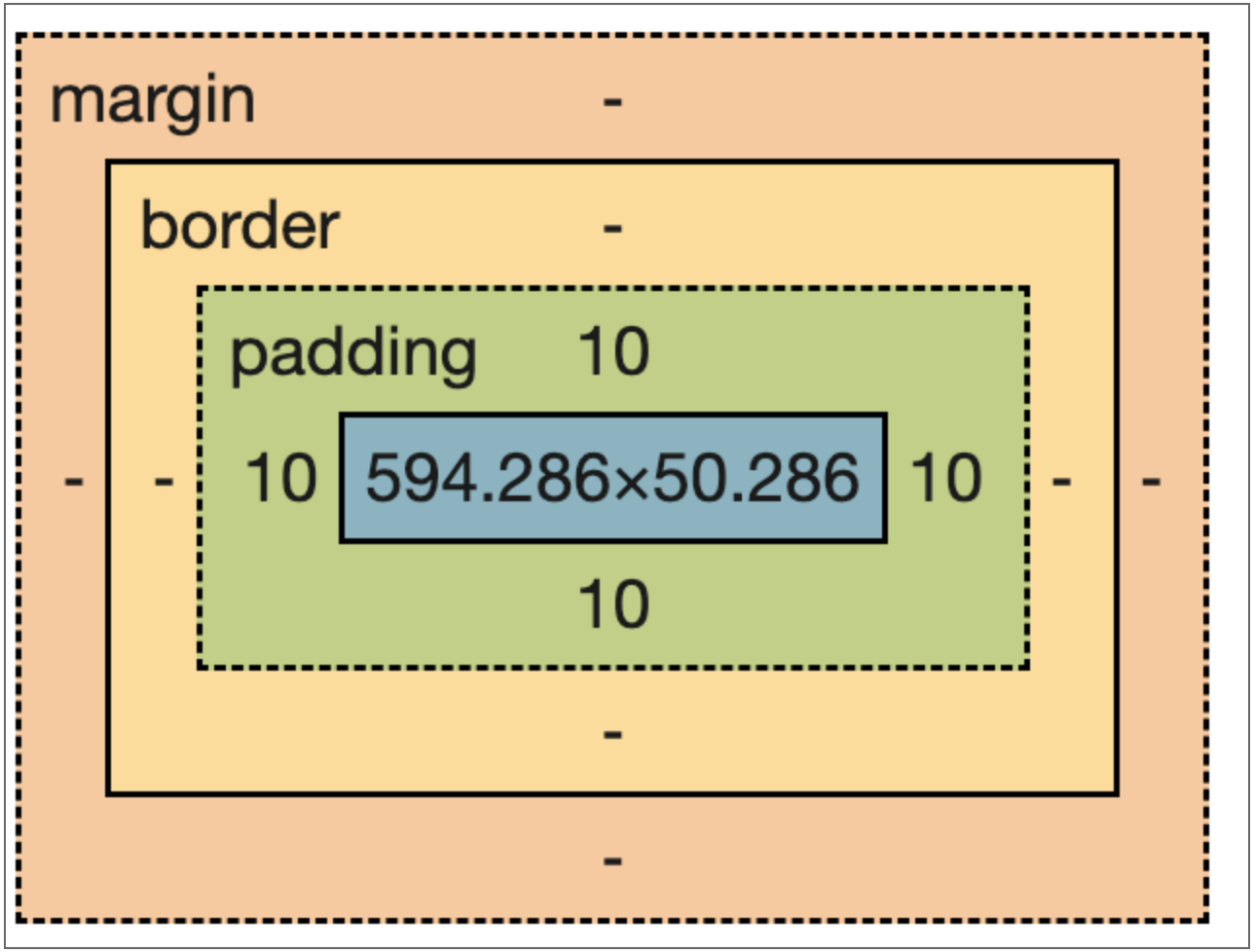
- Core Job skill, practice it now!
    - "Where is this space coming from?"
    - "What styles are on this element"?
    - Browser has some DEFAULT styles
        - Styles you didn't set!
        - Checkout `<ul>`!
- Coding should minimize "guessing"
- Use DevTools to *know* what is happening
- My advice to improve: "Use DevTools more"

Tips for Inspecting Styles

- Right-Click:Inspect selects THAT element
    - Saves time/effort, esp. when many elements
- Hover Elements to see Box Model on rendered
- Select Element to see styles of that element
    - Overridden styles have strike-through
    - It tells us `height` isn't applied to `inline`
- Styles lists source of styling
    - "user agent" here means the browser
- CSS Properties list results of shorthand
- Styles Shows Box Model

DevTools shows Element Box Model

- Content **height**, **width**, and **padding**
- What are **border** and **margin**?


margin
-
border
-
padding
10
-
-
10
594.286×50.286
10
-
-
10
-
-

Border

```
.chonky-cats {
  display: inline-block;
  width: 100px;
  padding: 16px;
  border-style: solid;
  background-color: plum;
}
```

- Creates a line around `padding`
- Shorthand property (the four TRBL sides)

More About Border

- Box Model shows `border-width`
  - Also a TRBL shorthand
- `border-color` controls color
  - Also a TRBL shorthand
- All can be set with `border` shorthand

```
.chonky-cats {
  display: inline-block;
  width: 100px;
  padding: 16px;
  border: 1px solid black;
  background-color: plum;
}
```

Box Sizing

How much width does `.chonky-cats` use up?

```
.chonky-cats {
  display: inline-block;
  width: 100px;
  padding: 16px;
  border: 1px solid black;
  background-color: plum;
}
```

- 100+16+16+1+1 = 134px
- What `width` to set to take up 150px?
- Often inconvenient to do that math

Box Sizing Values

- With `box-sizing: content-box;` (default)
  - `width`, `height` sets content dimensions
- With `box-sizing: border-box;`
  - Set total of content+padding(s)+border(s)
  - Content dimensions set by automatic math

"Universal selector" can match all elements:

- Not uncommon to see

```
* {
  box-sizing: border-box;
}
```

Margin is space outside element

Think of it as the elements "personal space"

- Not _in_ the element
  - Does not show elements background color
  - Not inside border
- But element doesn't want other elements there
- `margin` is a TRBL shorthand

```
.chonky-cats {
  display: inline-block;
  width: 100px;
  padding: 16px;
  border: 1px solid black;
  margin-left: 40px;
  background-color: plum;
}
```

Many Elements have default margins

- paragraphs, lists, section headings, etc
- Our block elements don't fill page width?
    - DevTools can show us a margin involved
- You often want different margins than default!

Proximity and **Whitespace**

- Communicate relationships
- **Hierarchy** Design Principle

Backgrounds or Borders

- Use **Contrast** Principle

Margin Collapse - A common source of confusion

Imagine the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>
  <link rel="stylesheet" href="styles.css"/>
</head>
<body>
  <header><h1>This is a top heading</h1></header>
  <main>
    <p>Paragraph 1</p>
    <p>Paragragh 2</p>
  </main>
  <footer>This is a footer</footer>
</body>
</html>
```

Sample CSS for Margin Collapse demo

```
body {
  margin: 0;
  background-color: lime;
}

header, footer {
  background-color: #bada55;
}

main {
  background-color: #c0ffee;
}
```

Margin Collapse in action

- Paragraphs (`<p>`) are children of `<main>`
- Top and bottom of those paragraphs do NOT show `<main>` background color

Exploring with DevTools increases confusion

- `<header>` contains `<h1>`
    - But `<h1>` margin extends OUTSIDE `<header>`
- `<p>` are inside `<main>`
    - But `<p>` margins extend OUTSIDE `<main>`
- `<h1>` margin and top `<p>` margin OVERLAP

This is all due to **margin collapse**

What is Margin Collapse?

General rule of Box Model:

- The box contains the contents
- When `height` and `width` are `auto;` (the default)
    - Box will size to fit the contents

Margins with **margin collapse** can violate this

- Collapse **upwards** (top) and **outwards** (parent)
- Only when margin collapse happens!
    - Requires a **block formatting context**
    - Never with `display: flex;` or `display: grid;`

Why does Margin Collapse exist?

Remember the original context of the web

- Sharing big linking text documents
  - Like Wikipedia

Margin Collapse makes a lot of things more convenient

- Paragraphs have top/bottom margins
  - But 2 `<p>` in a row won't get double margin

Margin Collapse makes OTHER things LESS convenient

- Like teaching/learning the box model

What do we do with this knowledge?

When debugging with DevTools

- If margins aren't included in parent content box
    - Margin collapse is to blame
- This is a rare spot DevTools doesn't help you

You can avoid Margin Collapse

- Switching to `display` of `flex`/`grid`
- By having `padding`
- By having a border on parent

We Experimented by Changing CSS

- That's fine
- Great, even!
- But DevTools is sometimes useful for that

Tips for (temporarily) Changing the Page

- All changes reset on page load/reload
- Can delete/edit elements
- Checkboxes in Styles can remove/apply property
- Can alter values in Styles
    - Can see possible non-unit values
    - Up/down to change numeric values
- Can add to `element.style`
- `+` to add new rule, selector and all
- `.cls` to add/edit/remove classes

CSS Units

- `%` of container
- `vh` and `vw`
    - "viewport"
- `px` vs `rem` vs `em`
    - `px` is (mostly) fixed
        - fixed is often bad
    - `em` causes inheritance problem
    - Sizes based off of "root" font "em" width
        - "root" is `<html>` element
    - **https://css-tricks.com/html-vs-body-in-css/**
    - `rem` useful with browser text settings

So what units to use?

Users may have different text settings

- `px` for parts that don't change based on text size
- `rem` for parts that DO change based on text size

Do border sizes change based on text size?

- It Depends - you have to decide

CSS Custom Properties

Often we have values that we want to reuse

- Height/widths of elements interacted with (nav?)
- Colors (background, accent, highlight, etc)

Technically these are **custom properties**

- Sometimes called "CSS Variables"
- But they act like CSS properties
- Follow the normal cascading/precedence rules

Outside CSS

CSS took a long time to add "variables"

- Can't work everywhere even still

We will talk about SASS later in semester

- Has own solution for "variables"
- But SASS isn't actual CSS

This is the pure (but limited) CSS solution

Using a CSS Custom Property

Assign:

```
.some-selector {
  --my-var: black;
  --another: 5rem;
}
```

Use:

```
p {
  color: var(--my-var);
}
```

"Global" assign:

```
:root {  /* same as `html` */
  --main-bg-color: #BADA55;
}
```

Real World Example of CSS Custom Properties

Taken from **http://washingtonpost.com/**

```
a {
    color: var(--link-color);
    text-decoration: none
}

:root {
    --color-brand-blue-normal: #1955a5;
    --color-brand-blue-dark: #172a52;
    --color-ui-white: #fff;
    --color-ui-offwhite: #f7f7f7;
    --color-ui-gray-light: #d5d5d5;
    /* Cut ~100 lines */
    --primary-background: var(--color-ui-black);
    --secondary-background: var(--color-ui-gray-darkest);
    --primary-fill: var(--color-ui-white);
    --secondary-text: var(--color-ui-gray-light);
    --link-color: var(--color-brand-blue-normal)
}
```

Pseudo-classes

Added to a selector to indicate a state

- `:hover`
- `:focus` and `:focus-within`
- `:active`
- `:not()`
- `:first-child`
- `:nth-child()`

Pseudo-elements

Not elements, but allow you to style them like one

- `::selection`
- `::first-line` and `::first-letter`
- `::before` and `::after`
    - These require a `content` property

CSS Functions

- `calc()`
- `max()` and `min()`
- `clamp()`
    - 3 args, preferred should be a value that changes

Media Queries

- Wraps CSS Rules
- Rules applied or not based on query
- Says if the rules are matched

Screen Width

If CONDITION, apply CSS rules

```
@media (min-width: 1000px) {
  body {
    background-color: red;
  }
}
```

Reduced Motion

- Options are `no-preference` or `reduce`
- Which involves less work?
- Which is "safer"?

```
@media (prefers-reduced-motion: no-preference) {
  .my-element {
    animation: flashy-zoom-in-out 1s;
  }
}
```

Orientation

- If you care past width...

```
@media (orientation: portrait) {
  body {
    display: flex;
    flex-direction: column;
  }
}
```

Printing

A deep rabbithole

- Alternative to generating PDFs
- Not always the best alternative

```
@media print {
  h3 {
    page-break-before: always;
  }
}
```

Notes about floating

`float: left;` (etc)

Used to have inline elements flow around it

- Ex: paragraph of text wrapping around a small image

Do not use `float` for layout

- Was a common fix before flexbox/grids
- Only use to wrap text around an image
- A lot of outdated online advice