# FWDP 1000 – Day 9

Course: Web Development 1

Instructor: Martha Villa Martin

# Morning Review

- In your code editor, open **grid.css** from the **day-09** folder.
- Spend 5 minutes comparing the code in this files to the code you wrote for Assignment #6.
- We will review this together after that.

# Fit-content – CSS

In grid.css, notice the use of the **fit-content** value for the h1.

```
.banner h1 {
    width: fit-content;
}
```


https://developer.mozilla.org/en-US/docs/Web/CSS/fit-content

https://ishadeed.com/snippet/fit-content/

# Min & Max Content – CSS

The **min-content** and **max-content** values are similar and handy in certain situations.

https://developer.mozilla.org/en-US/docs/Web/CSS/min-content

https://developer.mozilla.org/en-US/docs/Web/CSS/max-content

# Agenda

- More CSS Grid
- Buttons vs. Links
- Menu Toggle
- SVGs
- Screen Readers

# More CSS Grid

# Explicit Grid

The explicit grid is what is created when we use any of the following:

- grid-template-columns
- grid-template-rows
- grid-template-areas

https://css-tricks.com/difference-explicit-implicit-grids/

# Implicit Grid

An implicit grid is only created if one or both of the following are true:

- We have more grid items than our explicit grid has room for.
- We place a grid item outside of the explicit grid.

# Auto Rows vs Auto Columns

By default, CSS Grid will add new rows when creating an implicit grid.

With grid-auto-flow, you can have it create new columns instead of new rows.

https://developer.mozilla.org/en-US/docs/Web/CSS/grid-auto-flow

https://css-tricks.com/snippets/css/complete-guide-grid/#grid-auto-flow

# Buttons vs. Links

# Buttons vs. Links

The **<button>** element is for changing something on the page.

The **<a>** element is for navigating to another page or within the page.

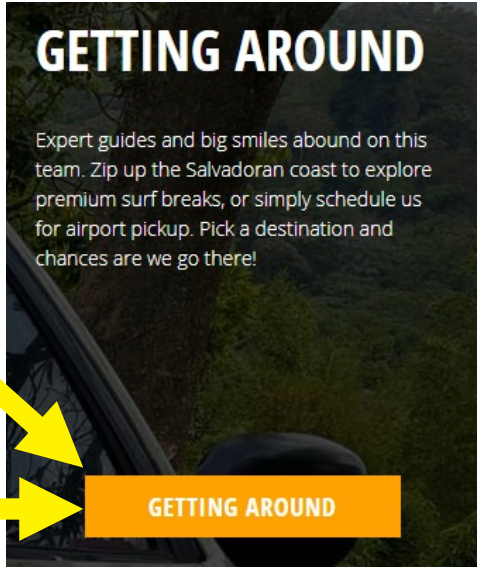
Neither element can be inside the other element.

# “Button” Links

If you want a link to look like a button, add a class and style it accordingly.

```
a.faux-btn {                                    style.css?v…04216e7:593
  clear: both;
  display: block;
  max-width: 16rem;
  margin: ▶ 0 auto;
  padding: ▶ .5em;
  background-color: #FDA200;
  font-family: 'Open Sans Condensed',
     sans-serif;
  font-weight: 700;
  font-size: 1.25em;
  text-transform: uppercase;
  text-align: center;
  letter-spacing: 0.05em;
  color: #fff;
  transition: ▶ all ease 0.3s;
}
```

```
▼<div class="booking-box">
   <h2>Getting Around</h2>
  ▶<div class="excerpt">…</div>
   <a href="https://www.sunzal.com/getting-around/" class="faux-btn">Getting Around</a>
 </div>
```



# Buttons

Buttons are often used in web forms instead of <input type=submit>

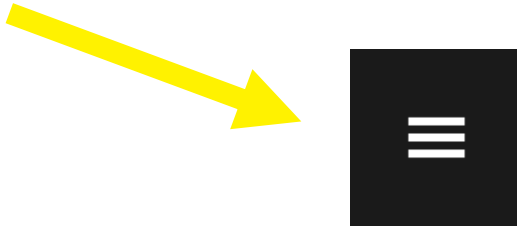
In addition to being used for forms, a **<button>** element can be used for a hamburger menu or any other interactable element on a page.

```
▼<button class="menu-toggle" aria-controls="primary-menu" aria-expanded="false">
  <span class="screen-reader-text">Menu</span>
 ▶<svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
width="32" height="32" viewBox="0 0 32 32">…</svg>
</button>
```

# Menu Toggle

# Hamburger Menu

The “hamburger menu” is a common feature of many sites.



At a basic level, it is a button that toggles opening and closing a menu of links.

# Exercise: Menu Toggle

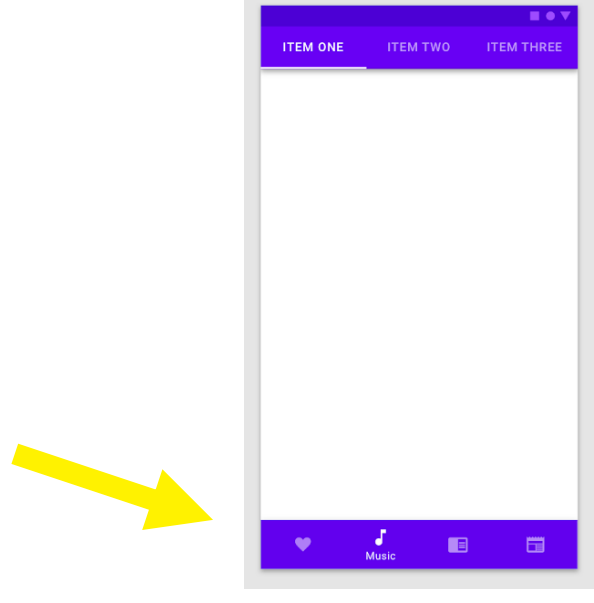
Many of our files already have this functionality…

…let's see how to create one ourselves.

# Exercise: Thumb Navigation

A popular alternative to the traditional hamburger menu is the thumb navigation…

…let's see how to create that.



# More Advanced JavaScript

Open up the navigation.js file in our files.

…let’s see what is different from our simple version.

# SVGs

# SVGs

Scalable Vector Graphics (SVGs) are generally what we use for creating basic graphics and icons on the web.

You can create them in Illustrator or find SVGs online.

An SVG is simply code that renders like an image.

# SVG as Image

SVGs can be saved as **.svg** files and used in **<img>** elements:

```
<img src="logo.svg" alt="Company Logo">
```

They can also be used in CSS as background images:

```
.header {
    background-image: url(logo.svg);
}
```

# Inline SVG

Inline SVG is superior to using an <img> element because you can manipulate the SVG in your CSS.

Take the SVG code and paste it into your HTML file:

```
<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">
    <path d="M24 6h-24v-4h24v4zm0 4h-24v4h24v-4zm0 8h-24v4h24v-4z"/>
</svg>
```

# SVG Icon Websites

These are two websites for quickly finding SVG icons:

- https://iconmonstr.com/
- https://icomoon.io/

Font Awesome is also very popular but loading the whole library is unnecessary if you only need a few SVGs.

# Inline SVG – CSS

Now you can target the **<svg>** to add styles or the **<path>** element to set the color using the **fill** CSS property.

```
svg {
    margin: 0 1rem;
}

svg path {
    fill: red;
}
```

# Inline SVG – CSS

Use the **currentColor** value to inherit the **color** value from its ancestor properties.

```
.logo path {
    fill: currentColor;
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/color_value#currentcolor_keyword

# Inline SVG – Title

Add a **<title>** element to inline **<svg>** elements to function like the **title** attribute for an **<img>** element.

```
<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">
    <title>Company Logo</title>
    <path d="M24 6h-24v-4h24v4zm0 4h-24v4h24v-4zm0 8h-24v4h24v-4z"/>
</svg>
```

This will make the title appear when hovering a cursor over the SVG and allow screen readers to explain the SVG.

# Inline SVG – Label

If you don’t want the title to appear when hovering, add the **aria-label** attribute in the **<svg>** tag instead.

```
<svg aria-label="Company Logo" xmlns="http://www.w3.org/2000/svg" width="24"
height="24" viewBox="0 0 24 24">
     <path d="M24 6h-24v-4h24v4zm0 4h-24v4h24v-4zm0 8h-24v4h24v-4z"/>
</svg>
```

This allows screen readers to explain the SVG.

# Inline SVG – Hidden

If you have an SVG as a decorative icon with text accompanying it, you can hide the SVG from screen readers.

```
<svg focusable="false" aria-hidden="true" xmlns="http://www.w3.org/2000/svg"
width="24" height="24" viewBox="0 0 24 24">
     <path d="M24 6h-24v-4h24v4zm0 4h-24v4h24v-4zm0 8h-24v4h24v-4z"/>
</svg>
<p>Success! Your order went through.</p>
```

# Accessible SVGs

The rules and best practices are regularly changing for making SVGs accessible.

Here are some resources:

https://www.smashingmagazine.com/2021/05/accessible-svg-patterns-comparison/

https://css-tricks.com/accessible-svgs/

https://a11y-guidelines.orange.com/en/articles/accessible-svg/

# Screen Readers

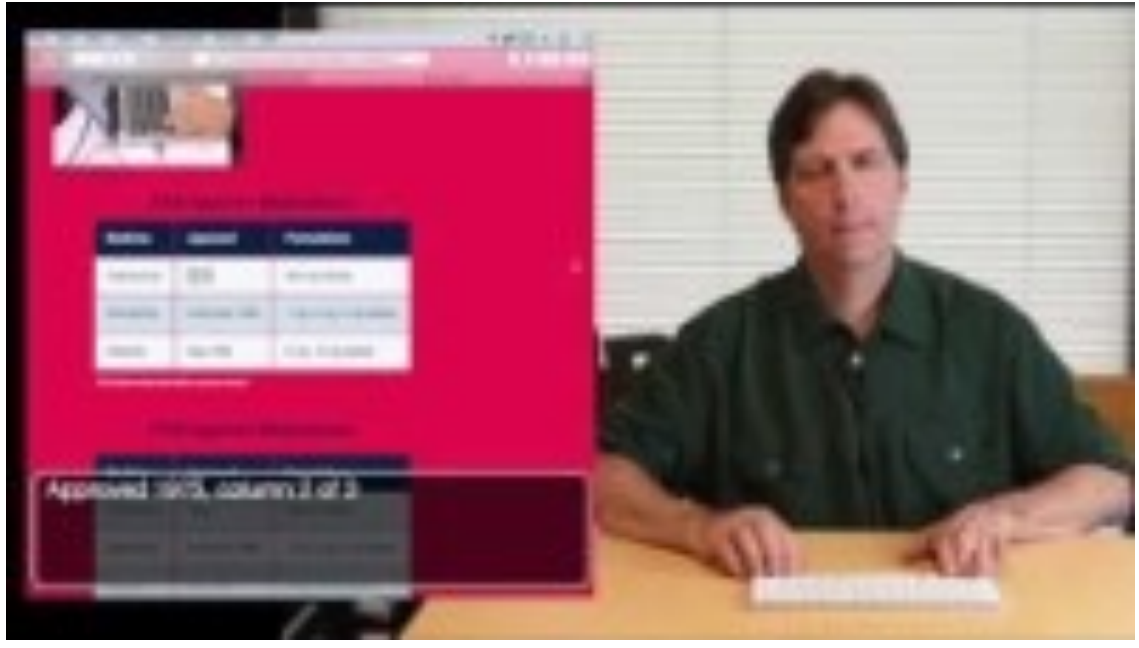
# Screen Readers

Screen readers are a type of assistive technology (AT) that can render text and images as speech or braille output.

These are some commonly used screen readers:

- JAWS -- https://www.freedomscientific.com/products/software/jaws/
- NVDA -- https://www.nvaccess.org/download/
- Voice Over -- https://www.apple.com/ca/accessibility/vision/

# Screen Readers



https://www.youtube.com/watch?v=dEbl5jvLKGQ

# More Screen Reader Info

This article shows each of the top three screen readers being used with a breakdown of Pros / Cons:

https://css-tricks.com/comparing-jaws-nvda-and-voiceover/

This article explains how to setup your device to test screen readers:

https://www.sarasoueidan.com/blog/testing-environment-setup/

# Helping Screen Readers

In addition to writing valid, semantic HTML, you can help screen readers further.

Here are two examples:

- Use CSS to make hidden text accessible to screen readers.
- Add a “skip to content” link.

# Screen Reader Text

Sometimes there is text you want screen readers and search engines to read but you don’t want visible on the webpage.

For example:

```
<a href="your-url">Read more<span class="screen-reader-text"> about cute kittens</span></a>
```

The class “screen-reader-text” can be used to hide the text visually but allow screen readers to explain the link further.

# Screen Reader Text - Example

There are many examples of CSS code to use, I prefer this one used by WordPress:

https://make.wordpress.org/accessibility/handbook/markup/the-css-class-screen-reader-text/

Copy and paste the CSS into your normalize.css file and use the “screen-reader-text” class on any website you create.

# Skip to Content Link

The “screen-reader-text” CSS also helps with creating visually hidden “skip” links. For example:

```
<a class="screen-reader-text" href="#site-main">Skip to content</a>
```

When keyboard navigating using the ‘tab’ key, the link will become visible and clickable.

# Accessibility is Broad

Think broadly about accessibility. Make your website **easy** to use and **pleasant** to use for as many people as possible.

Follow the four principles of accessibility:

1. Perceivable
2. Operable
3. Understandable
4. Robust

# Prefers Reduced Motion

For instance, when adding animation to your sites, consider the users that want less motion…

```
@media (prefers-reduced-motion) {
    /* add code here */
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-reduced-motion

# Dark Mode

If users have set a preference for dark mode or light mode in their operating system or browser, we can write styles accordingly…

```
@media (prefers-color-scheme: dark) {
    /* add code here */
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-color-scheme

# Dark Mode Buttons

Alternatively, you can let users toggle dark mode on and off for only your site.

The simplest way to do this is use JavaScript to add and remove a class on **<body>.**

https://css-tricks.com/a-complete-guide-to-dark-mode-on-the-web/

https://github.com/GoogleChromeLabs/dark-mode-toggle

# New Media Queries

This article shows some of the media queries I mentioned here and some new ones that are coming to browsers.

Check the browser support to see which you can use now.

https://www.smashingmagazine.com/2023/08/css-accessibility-inclusion-user-choice/

# Text over Images

Read this two-part article about designing accessible text over images.

It has tons of great examples worth looking at for inspiration.

https://www.smashingmagazine.com/2023/08/designing-accessible-text-over-images-part1/

https://www.smashingmagazine.com/2023/08/designing-accessible-text-over-images-part2/

# Testing for Accessibility

Here are some ways to test that your site is accessible:

- Navigate your website using just a keyboard.
- Use an accessibility checklist and tools:
  - https://www.a11yproject.com/checklist/
  - https://a11y-101.com/
  - https://wave.webaim.org/
  - Chrome's Lighthouse in Developer Tools

# Accessibility Guides

This is a fantastic guide for accessibility:

https://www.smashingmagazine.com/2021/03/complete-guide-accessible-front-end-components/

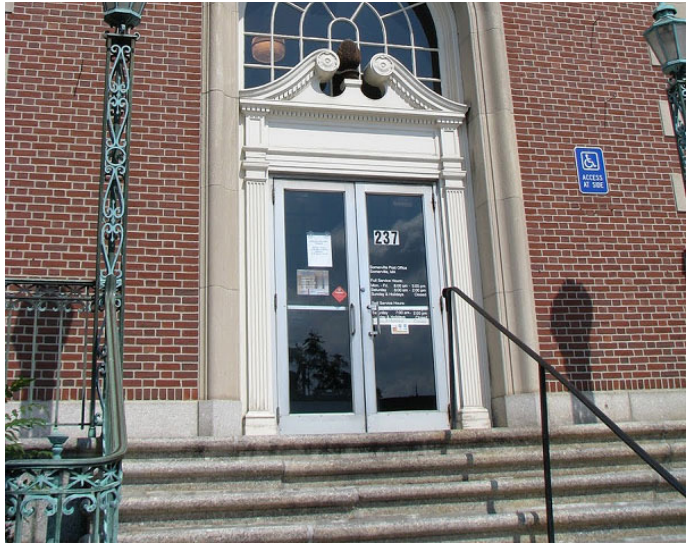
A lot of what it covers involves interactivity like HTML forms and JavaScript which you will cover more in future courses.

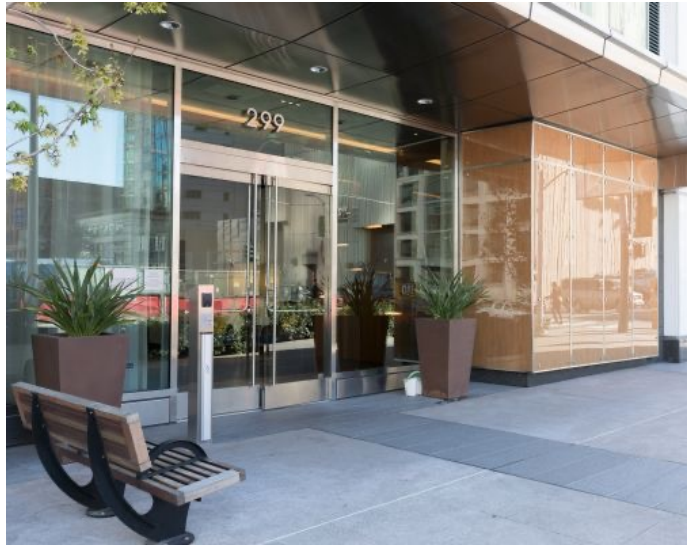
This has a **ton** of resources on accessibility too:

https://stephaniewalter.design/blog/accessibility-resources-tools-articles-books-for-designer/

# Think Broadly about Accessibility

Digital accessibility benefits everyone in the same way accessibility in the physical world does.



# Lab Time for Project

# Country Website

No assignment today.

Use the rest of the day to work on your Country Website.

It is due May 26.

# Peer Review

Before submitting your Country Website, consider asking a classmate to look at the page and your code.

Do the same for them.

Give each other suggestions and see if you can find any issues with the code, design, functionality, accessibility, etc.

# Next Class

Next class will be our final class for the Web Development 1 course.

There will be time in the morning to review anything we have covered in the course.

Are there topics you would like to review?

# QUESTIONS & ANSWERS