CSS Frameworks

Now that you've done some CSS, we'll take a look at CSS frameworks. In this document, we'll take a quick look at ZURB Foundation (has similar features to Bootstrap but with a lot more accessibility features baked in).

A framework is **a set of pre-configured CSS classes** which allow you to rapidly build a webpage layout. This allows you, as a developer or designer, to build a quick prototype without having to spend most of your time fiddling with CSS for your layout. Frameworks DO NOT mean that you don't need to style at all, but it just makes the layout building faster and easier.

There are a lot of frameworks out there, but we're only going to look at ZURB Foundation and a little for TailwindCSS (bigger learning curve due to the number of classes). A lot of them, if not most of them, use the grid system (explained below and not to be confused with CSS Grid). The important thing to know is how to understand and read the documentation because if you can use one framework, you can easily use any other framework because it's really about just using the classes in the manner described in the documentation. There is no special knowledge really needed to use a framework.

Word of caution

Just because frameworks are available and popular, do not lose sight of the importance of having your own knowledge of HTML and CSS. **Frameworks come and go, but understanding of HTML and CSS transfers across all frameworks.** Frameworks are just meant to be a tool to make things easier and it's nice to have some familiarity with one or two but core knowledge of HTML and CSS is more important (even when looking for work). Occasionally, there are bugs or oversights in a framework and CSS knowledge can save you on those occasions. Additionally, if you only need little things styled here and there, it is impractical to use a framework when vanilla CSS would work as well and with more efficiency.

ZURB Foundation

ZURB Foundation is a popular framework (another very popular one is Bootstrap). ZURB is a full-featured framework and includes some of the following features and more:

- forms
- buttons
- slider
- switches (turn a checkbox or radio button into a toggle switch)
- menus (dropdown, drilldown, accordion, and more)
- pagination styles

All these extra features mean that ZURB is a somewhat heavier solution. For example, ZURB requires you to include the following files in your HTML page if you want to use all the features:

- ***foundation.css*** (use ***foundation.min.css*** on production sites)

- ***jquery.js***
- ***what-input.min.js***
- ***foundation.min.js***
- ***app.js*** **(required to run the Foundation scripts)**

There is also an ***app.css*** in the template files but you technically don't need it as it's meant for you to put your custom CSS there.

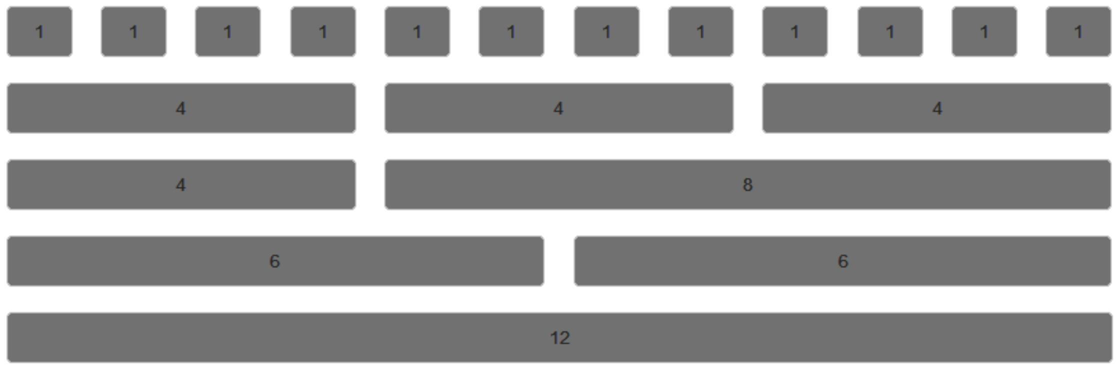
To get ZURB, go to https://get.foundation/. On the main page, click **Download Foundation 6** and extract the ZIP file. You can view all the CSS and JS files you'll need for all features by opening the *index.html* in your code editor and seeing which files have been added. (Note that you can also work with ZURB using Sass but you won't get an intro to Sass until the next module.) Find the full documentation at https://get.foundation/sites/docs/index.html. (Get used to reading documentation because you'll need to do it a lot during your professional life.)

In an HTML page, add the files from the required list above (put the ZURB CSS and JS files in a ***zurb*** folder if you wish. As another option (for ***foundation.min.css*** and ***foundation.min.js*** only), you can use the compressed CSS and JS CDN links you can find in the documentation under **Sites Docs > Getting Started > Installation > CDN Links** (see the right sidebar on the *Installation* page). If you want to use all the features, you **need to download the files**. If any feature in the documentation has "JS" next to it, you'll need all of the JS files listed above for it to work.

One nice thing about ZURB is that many of the interactive components have some accessibility features baked in. You should still double check and add in any that are missing but most are about 80% to 90% there for keyboard users. It is best practice in any case to always check the accessibility of your page. Don't just assume everything is good.

The following instructions are for the newer CSS flexbox-based *XY Grid* layout system (in the left sidebar: Setup > XY Grid).

For building a layout, you'll need to use *XY Grid*. As is very common in frameworks, ZURB uses a grid system (12-column). Think of this like the column grids used when building a design. ZURB classes are named based on how many of those 12 columns the container will span.



For example, if you want a two-column layout with <main> being twice the width of the sidebar, you would need:

```
<div class="grid-x">
  <main id="main" class="large-8 cell">
```

```
    I'm two thirds wide (8/12 = 2/3)
  </main>
  <aside id="sidebar" class="large-4 cell">I'm one third wide</aside>
</div>
```

The wrapper container which defines a *row* of containers should have the class `grid-x`. Note that a row means that containers are lined up along the x-axis, hence using `grid-x` as the class name. Within a grid, the child elements should have the `cell` class. In the above example, the `"large-8 cell"` class indicates that the <main> should span 8 columns in large screens.

Correspondingly, you can define the layout for small and medium screen sizes using `.small-#` and `.medium-#` classes alongside the `.cell` class.

The default breakpoints (from the documentation) are:

- Small: any screen
- Medium: >= 640px (min-width: 40em)
- Large: >= 1024px (min-width: 64em)

ZURB is built on a mobile-first philosophy. This means that your media queries are using min-width at the given breakpoints if you want things to "shift" at the same time as your layout.

If you need to change the order of containers, simply use the CSS `order` property since the layout system is based on flexbox.

Note: You can also define a vertical grid using `grid-y` for the column container in place of `grid-x`.

In the case of a two-column layout where you want the <main> to be displayed on the right side of the page, you can use:

```
<div class="row">
  <main id="main" class="large-8 columns">
    I'm two thirds wide (8/12 = 2/3)
  </main>
  <aside id="sidebar" class="large-4 columns">I'm one third
wide</aside>
</div>
```

Then use the CSS **order** property to pull the <aside> left.

There are many other features you can try such as modal windows (Reveal) or accordions. Note that some of the features require JS. Any features requiring JS will have a little "JS" flag next to the documentation link in the sidebar. These are features which require the JS files listed earlier (**plus the jQuery library**):

- ***foundation.min.js***
- ***what-input.min.js***

The JS files not only provide the functionality for those interactive features, but they also handle some of the accessibility stuff (e.g. adding appropriate ARIA properties, adding standard keyboard shortcuts).

Other frameworks

Bootstrap: try it yourself!

Bootstrap (another very popular framework) is similar to ZURB. Go to the Bootstrap site (http://getbootstrap.com/) and take a look at the docs (the **Documentation** link in the menu then go to **Layout > Grid**). You can see that the grid naming system is similar to ZURB, except instead of "large-4" for Bootstrap you would use "col-lg-4". Additionally, each row of columns uses the class "row" and your rows are wrapped with a container using the "container" class. You should read the definitions for large, medium, small, and extra-small to see which screen widths they apply to. There are many options, so make sure you check the documentation.

To use Bootstrap, you can download the files to use locally or use the CDN (see the Getting Started page). Try recreating the layout you built with ZURB. You can also explore some of the other features/components.

Note: Although frameworks make your work easier, don't forget about usability and accessibility when you use a framework. Bootstrap clearly states in its documentation that keyboard navigation accessibility is NOT implemented for some components (e.g. accordions and tabs) and you need to add your own Javascript to implement this for those that are missing (following the WAI-ARIA authoring practices, depending on the design component)

Note^2: Don't just blindly keep default styling. Modify the styling as needed for better usability and to customize the look. Frameworks provide helper classes but you can still totally change the look (and you should).

Tailwind CSS: try it yourself!

Tailwind CSS is a framework that is getting more and more popular. Unlike ZURB or Bootstrap which has a lot of built-in styling for each component, Tailwind CSS has a lot of **utility classes**. These are CSS classes which only serve one purpose (e.g. adding rounded corners), allowing you to have flexibility in styling. This does result in a lot of CSS classes in the HTML code.

To try it out, you can use it straight out of the box by using the CDN link (when you click on the Docs, you can find installation/CDN link info right at the start of the page).

Try out the following documentation pages to get you started: (there are a **lot** of classes, so you'll always be referring back to the documentation)

- The fundamental utility-first approach:
  - https://tailwindcss.com/docs/utility-first
- Layout-related stuff:
  - https://tailwindcss.com/docs/responsive-design
  - https://tailwindcss.com/docs/flex-basis (for sizing)
  - https://tailwindcss.com/docs/padding
  - https://tailwindcss.com/docs/margin
- Interesting stuff:
  - https://tailwindcss.com/docs/adding-custom-styles
  - https://tailwindcss.com/docs/dark-mode