**Week 04**

**W04 Learning Activities**

1. [CSS Layouts: Flex](https://byui-cse.github.io/wdd130-ww-course/week04/prepare-css-flex.html)
2. [CSS Layouts: Grid](https://byui-cse.github.io/wdd130-ww-course/week04/prepare-css-grid.html)
3. [CSS Layout Basics & Exercises](https://byui-cse.github.io/wdd130-ww-course/week04/prepare-layouts.html)
4. Optional Practice: [Home Page Layout Enhancements](https://byui-cse.github.io/wdd130-ww-course/week04/ponder-course.html) | ✔ [Page Evaluation Tool](https://byui-cse.github.io/wdd130-ww-course/grader/w04-homepage.html)

**Assignment**

1. W04 Assignment: Rafting Site - [Update About Us Page](https://byui-cse.github.io/wdd130-ww-course/week04/prove-rafting-enhance.html) | ✔ [Page Evaluation Tool](https://byui-cse.github.io/wdd130-ww-course/grader/w04-project-aboutus.html)

**CSS Layouts: Flexbox**

**Overview**

The **flex** CSS display property makes creating and controlling responsive layouts easy especially in a single horizontal or vertical container.

Flexbox is a one-dimensional layout method for laying out items in rows or columns. Items flex to fill additional space and shrink to fit into smaller spaces. Flexbox layout is most appropriate to the components of an application, and small-scale layouts, while the Grid layout is intended for larger scale layouts.

"For a long time, the only reliable cross-browser compatible tools available for creating CSS layouts were features like floats and positioning. These work, but in some ways they're also limiting and frustrating. The following simple layout designs are either difficult or impossible to achieve with such tools in any kind of convenient, flexible way:

* Vertically centering a block of content inside its parent.
* Making all the children of a container take up an equal amount of the available width/height, regardless of how much width/height is available.
* Making all columns in a multiple-column layout adopt the same height even if they contain a different amount of content."

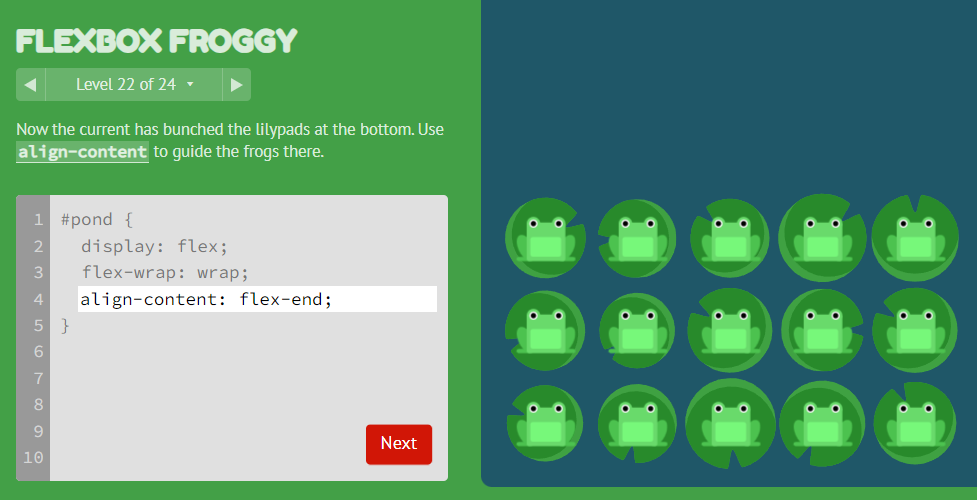
- MDN

**Prepare**

* Video: ▶️ [Learn Flexbox](https://youtu.be/fYq5PXgSsbE) - Web Dev Simplified
* Read: [Flexbox](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox) - MDN

**Activity Instructions**

1. Complete the following online tutorial on CSS Flexbox (24 Levels): [Flexbox Froggy](https://flexboxfroggy.com/)

Screenshot of Flexbox Froggy Level 22

## CSS Layouts: Grid

### Overview

CSS Grid Layout is a two-dimensional layout system for the web that allows you to arrange content content in rows and columns. It includes many features that make building complex layouts straightforward. The goal is for you to understand the basic features of CSS Grid and how to use them to create layouts.

### Prepare

* Video: ▶️ [Grid Layout (Part 1 of 2)](https://www.youtube.com/embed/1FyVcc1pIVU) - (7:15 mins, [Grid Layout - Part 1 Transcript](https://byui-cse.github.io/wdd130-ww-course/text/grid_layout_1.pdf))
* Video: ▶️ [Grid Layout (Part 2 of 2)](https://www.youtube.com/embed/e3_70BJglG4) - (5:59 mins, [Grid Layout - Part 2 Transcript](https://byui-cse.github.io/wdd130-ww-course/text/grid_2.pdf))

Parent elements that contain a grid layout are given the **display: grid** declaration. The immediate children of that parent container can then respond to grid declarations (but grandchildren cannot).

### Activity Instructions

1. Complete this 28 lesson tutorial on CSS Grid: [Grid Garden](https://cssgridgarden.com/).

Screenshot of a gridgarden.com Level

#### Optional Resources

* [CSS Grid Property](https://www.w3schools.com/Cssref/pr_grid.php) - w3schools.com
* [Grids](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids) - MDN

## CSS Layouts

### Overview

Mastering CSS Flex and Grid can be challenging. This activity highlights some of the most relevant layouts and the principles behind them.

### Prepare

#### When do I use CSS Flex versus CSS Grid?

Depending on the desired layout, you can often make either method work. You can even use them together, such as using flex to align items within a grid item.

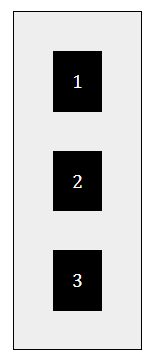
##### Choose Flex

* **Simple Layouts**: CSS Flex is ideal for one-dimensional layouts like navigation, or any group of items that need to be shown side by side or one on top of the other.
* **Alignment**: Flex is great at aligning items horizontally or vertically.
* **Dynamic Content**: Flex is often ideal for layouts with dynamic content where items sizes may vary.

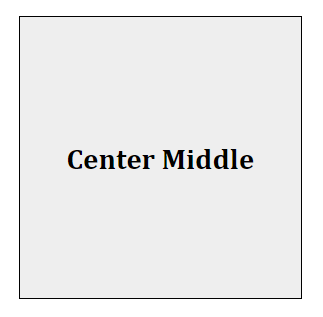
##### Choose Grid

* **Complex Layouts**: CSS Grid is generally more robust, allowing you to create two-dimensional layouts for common structures like tables, calendars, merged cells, and more.
* **Fixed Width Layouts**: Grid is suitable for fixed-width layouts, where item sizes are known.
* **Overlap and Layering**: Grid supports overlapping and the layering of the items.

#### Common Layout Methods

* Define the **direction** of the Flex items using **flex-direction: (row/column);** (row is default).
* display: flex;

flex-direction: column;

* **Centering Horizontally and Vertically**: The following code centers the child item both horizontally and vertically using **CSS Grid**. Of course, this code is applied to the parent, grid container.
* display: grid;

place-items: center;

Centering can also be controlled with the following code for both **Flex** and **Grid**:

display: flex; */\* or grid \*/*

justify-content: center;

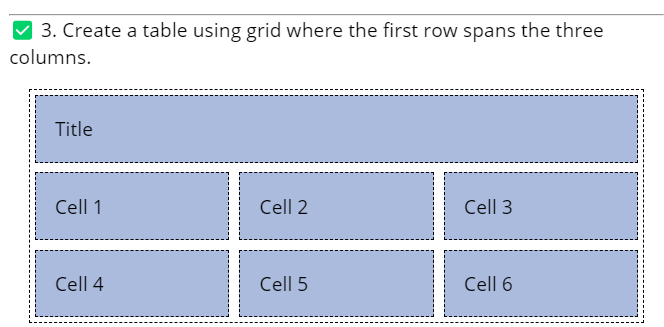
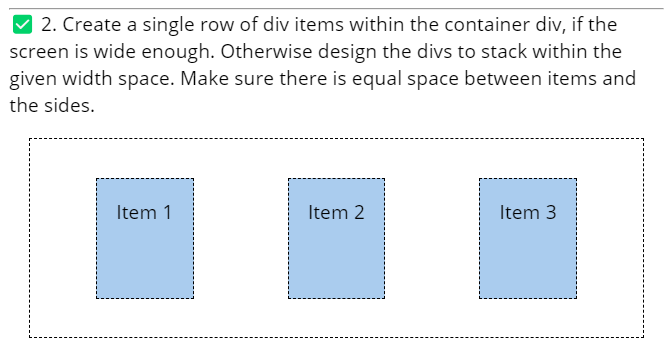
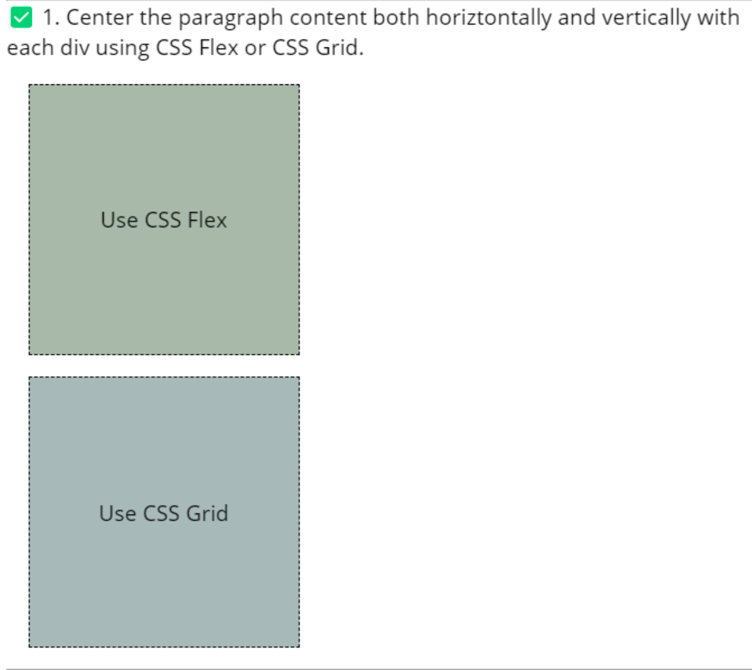
align-items: center;

### Activity Instructions

1. Navigate to this CodePen: [WDD Layout Exercises](https://codepen.io/BYU-Idaho/pen/zYgezLm)
2. Log into your CodePen account. If you do not have an account, you can create one for free. An account allows you to save pens to your own account.

Accounts at CodePen are free. You can use your Google, GitHub, Facebook, or an email account of your choice to sign up.

It is recommended that you obtain an account to manage your own snippets of work and fork, i.e., copy, useful pens that you discover. You are not required to get an account as you can view and edit the code in the pen for free, however, your changes will not be saved for future use and reference.

1. Complete the three exercises to the best of your ability. The following images are screenshots of the solutions:Check Your Understanding

Example: [Layout Exercises - Solution](https://codepen.io/BYU-Idaho/pen/poMYVwK) - CodePen

[**Back**](https://byui-cse.github.io/wdd130-ww-course/week04/index.html)

Copyright © Brigham Young University-Idaho | All rights reserved

**W04: Using CSS Flex and Grid on the Course Home Page**

**Overview**

This optional activity has you apply learning activity concepts to enhance the home page with modern **layout** techniques.

**Instructions**

1. Open your course home page, **index.html**, and stylesheet **styles.css**, for editing.
2. Refactor the CSS for the combined rule of **header**, **footer** to have a maximum width of 840 pixels and to center the content on the page. You will add in the main section later.Check Your Understanding
3. header, footer {
4. max-width: 840px;
5. margin: 0 auto;

}

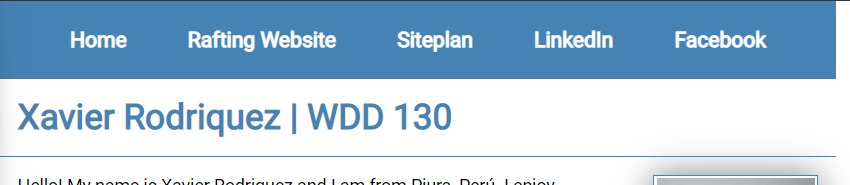
This means you should delete the **max-width** decoration from **header**, **main**, and **footer** selectors if necessary.

1. Use **CSS Flex** to make the navigation links display horizontally across the screen. The menu should be justified with even spacing and centered within the navigation bar. Remember to apply the CSS Flex declaration to the container.Check Your Understanding
2. nav {
3. background-color: steelblue;
4. padding: 1rem;
5. display: flex;
6. justify-content: space-evenly;
7. align-items: center;

}

1. If you have not already added a link to your LinkedIn profile (or to LinkedIn in general) and to your Facebook profile (or to Facebook in general), add those links to the **nav** navigation menu.
2. Alter the anchor tags to be block elements with padding and no text decoration. You are free to apply other relevant styling as well.Check Your Understanding
3. nav a {
4. display: block;
5. padding: 0.5rem;
6. text-decoration: none;
7. font-size: 1.2rem;
8. font-weight: 700;
9. color: #fff; /% shorthand for white %/

}

1. Add a bottom border to the heading **h1** element.Screenshot of navigation bar example.

Be sure to test your page locally often using the **Live Server** extension in VS Code. This will launch your default browser to the **localhost** address, and you can see your changes as you make them. localhost is a special address that your computer uses to refer to itself.

1. In your HTML file, encase the **main** and **aside** elements in a **div** element with a **class** attribute named "**grid**".
2. In your CSS file, add the **.grid** selector to the **header, footer** CSS selector to also make it part of the **max-width** and **margin**.

header, .grid, footer {

1. Make this new **div** a CSS **grid** display with the **aside** on the left and the **main** element on the right as shown below.

There are at least two ways to do this. Choose one. Not both.

* + Use **grid-template-columns** to define the columns and their widths.
  + Use **grid-column** and **grid-row** to define the grid items (main and aside) position.

Check Your Understanding

.grid {

display: grid;

align-items: center;

}

main {

grid-column: 2/3;

margin: 1rem;

}

aside {

width: 20rem;

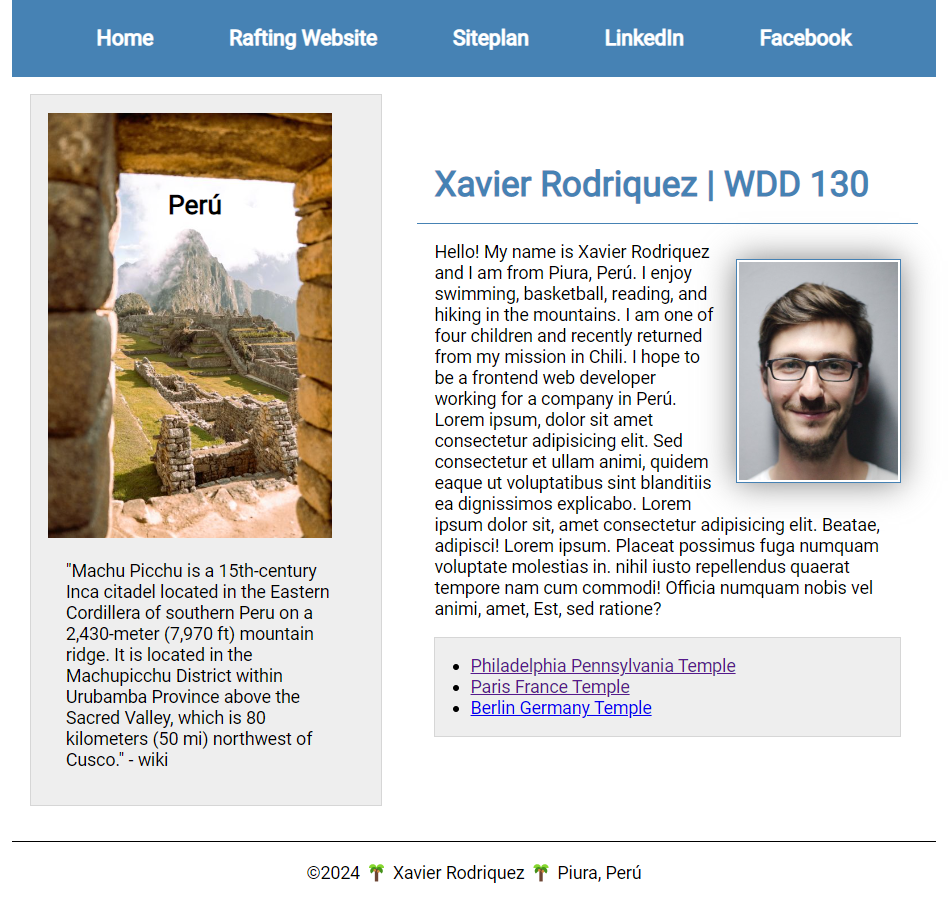
position: relative;

grid-column: 1/2;

grid-row: 1/2;

}

Note that the **aside** selector needs to apply the **grid-row** whereas the **main** selector did not. This is because of the order in the HTML structure.

Screenshot of the home page layout with grid.

Video Solution: ▶️ [Home Page Layout using CSS Flex and Grid](https://video.byui.edu/media/t/1_77ovedmw)

**Testing**

* Every page in this course will be expected to pass the [Create a single row of div items within the container div, if the screen is wide enough. Otherwise design the divs to stack within the given width space. Make sure there is equal space between items and the sides.development standards checklist](https://byui-cse.github.io/wdd130-ww-course/dev-standards.html).
* Validate the HTML by running the **Web Developer** extension in your browser and **Tools -> Validate Local HTML**. Correct any errors.
* Validate the CSS by using **Validate Local CSS**. Correct any errors that are reported.
* Check for color contrast violations at the AA level using the DevTools **CSS Overview** tool. Remember, you can find learn about this tool and how to use it in the development standards checklist.
* Commit and sync your changes to your wdd130 GitHub Pages enabled repository.
* Enter your GitHub username into this [evaluation tool](https://byui-cse.github.io/wdd130-ww-course/grader/w04-homepage.html) to get a basic Page Evaluation of your HTML.