FE Dev Homework

Unit 3 - WK 18 | Code a Responsive Website

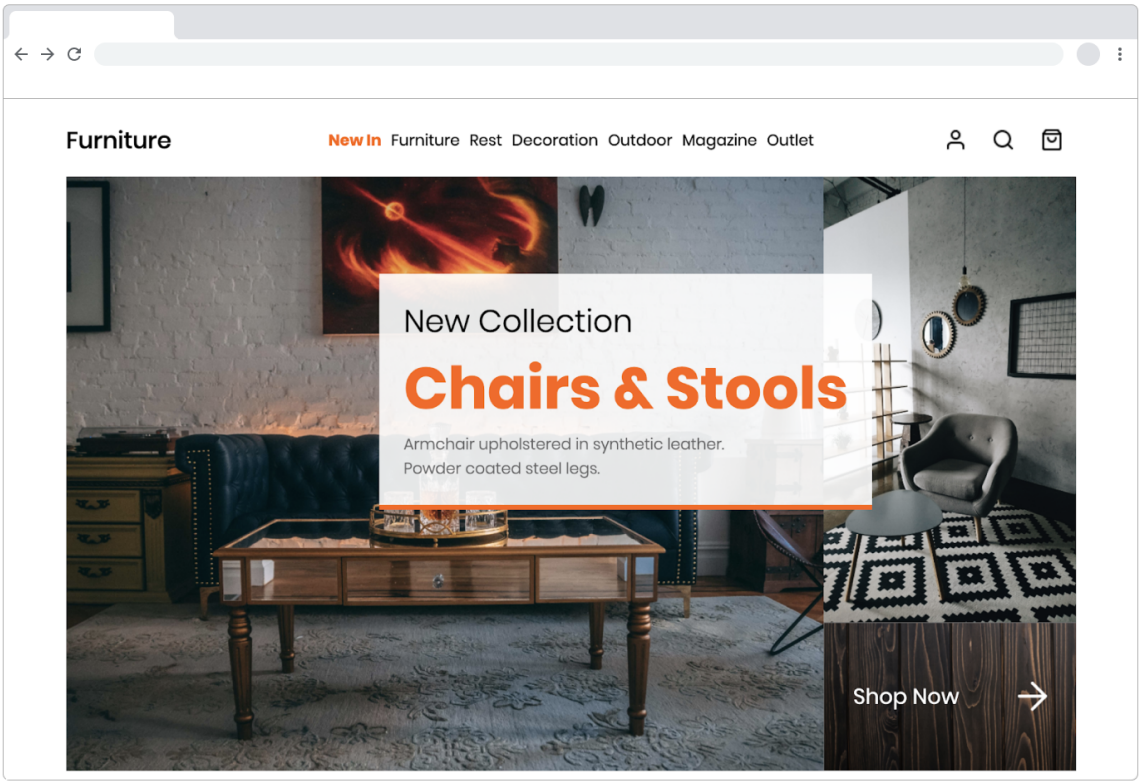
~10 hours | Individual | 7 Parts

(Assignment Due 19.3, April 2, 2022)

Homework 18 Overview

CSS Grids can be used to build simple and skeletal layouts. It can also be used to create polished, professional-level layouts.

In this week's homework, you will use CSS Grids to re-create the design of an online furniture store. You can view the final, responsive design >> here << and the instructions for building this start on the next page.



Part 1

1. In Visual Studio Code, open the files `<>index.html` and `# style.css`.
    - The files are located in the Homework folder in the .zip file.
    - You'll notice that the HTML and CSS for the navigation bar code have already been completed along with some typography-related CSS. That's because this activity focuses on using CSS to create layouts, not designs.
2. The HTML for the grid in the image at the beginning of this activity has already been completed. Review the HTML structure below.

    ** Students, make sure you review the index.html activity as you write your CSS selectors.*

```
<section id="heroGrid">
  <div id="featuredProduct" class="heroGridArea">
    <div id="headline">
      <h3 id="newCollection">New Collection</h3>
      <h2>Chairs & Stools</h2>
      <p>Armchair upholstered in synthetic leather. Powder coated steel legs.</p>
    </div>
  </div>
  <div id="secondaryProduct" class="heroGridArea"></div>
  <div id="CTA" class="heroGridArea">
    <h4>Shop Now</h4>
    <img src="images/arrow-right.png">
  </div>
</section>
```

Step 1: Define the Grid

1. In style.css, create a selector that targets the `heroGrid` ID.

```
/* Grid CSS ID*/

#heroGrid {  /* Grid CSS code */

}
```

- Add the following properties to define the overall size of the grid:

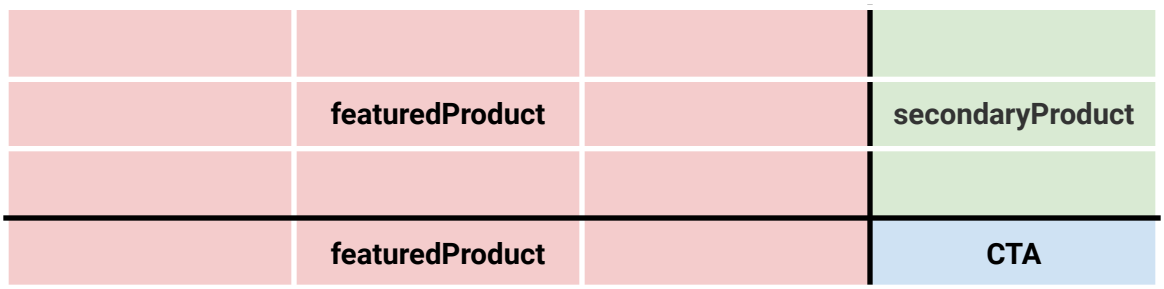
Property	Value
`margin`	`0px 5%`
`display`	`grid`
`min-height`	`500px`
`height`	`70vh`

Now that you have sized the CSS Grid, it's time to add more CSS properties.
Use the `grid-template-rows:`, `grid-template-columns:`, and `grid-template-areas:` properties to create the structure of the CSS Grid.

2. The `#heroGrid` ID at the top of the page has two rows and two columns.
    - The first row and first column each occupy `75%` of their respective areas.
    - The second row and second column each occupy the remaining `25%` of their respective areas.
    - Use the values above for the `grid-template-rows:` and `grid-template-columns:` properties.

    CSS hint: *Do not use commas between percentages.*

    Hero grid layout visual:



3. Next, add to your CSS. `#heroGrid{ grid-template-areas: }`.

    Hint: The `featuredProduct,` occupies the top and bottom left areas, the `secondaryProduct,` is in the top right and the `CTA` is in the bottom right.

Step 2: Add Content to the CSS Grid

HTML reference:

```
25    </nav>
26    <section id="heroGrid">
27      <div id="featuredProduct" class="heroGridArea">
28        <div id="headline">
29          <h3 id="newCollection">New Collection</h3>
30          <h2>Chairs & Stools</h2>
31          <p>Armchair upholstered in synthetic leather. Powder coated steel legs.</p>
32        </div>
33      </div>
34      <div id="secondaryProduct" class="heroGridArea"></div>
35      <div id="CTA" class="heroGridArea">
36        <h4>Shop Now</h4>
37        <img src="images/arrow-right.png">
38      </div>
39    </section>
```

1. Create three CSS selectors that target the IDs of each area within the grid from the HTML: `featuredProduct`, `secondaryProduct`, and `CTA`.
    - Within each selector, define the value for the `grid-area:` property so that each CSS selector and the CSS contained within it is associated with the correct area within the grid.

    Hint: *Spelling and capitalization in the syntax matter here, so pay attention.*

2. Add a `background-image:` to each ID selector made above using the images provided in the `images` folder. `featuredProduct = furniture1.png`, `secondaryProduct = furniture2.png`, and `CTA = woodbg.png`.

3. Add the following properties and values to the `#CTA` ID selector:
    - These will be used to style content within this grid area.

Property	Value
`justify-content`	`space-around`
`cursor`	`pointer`

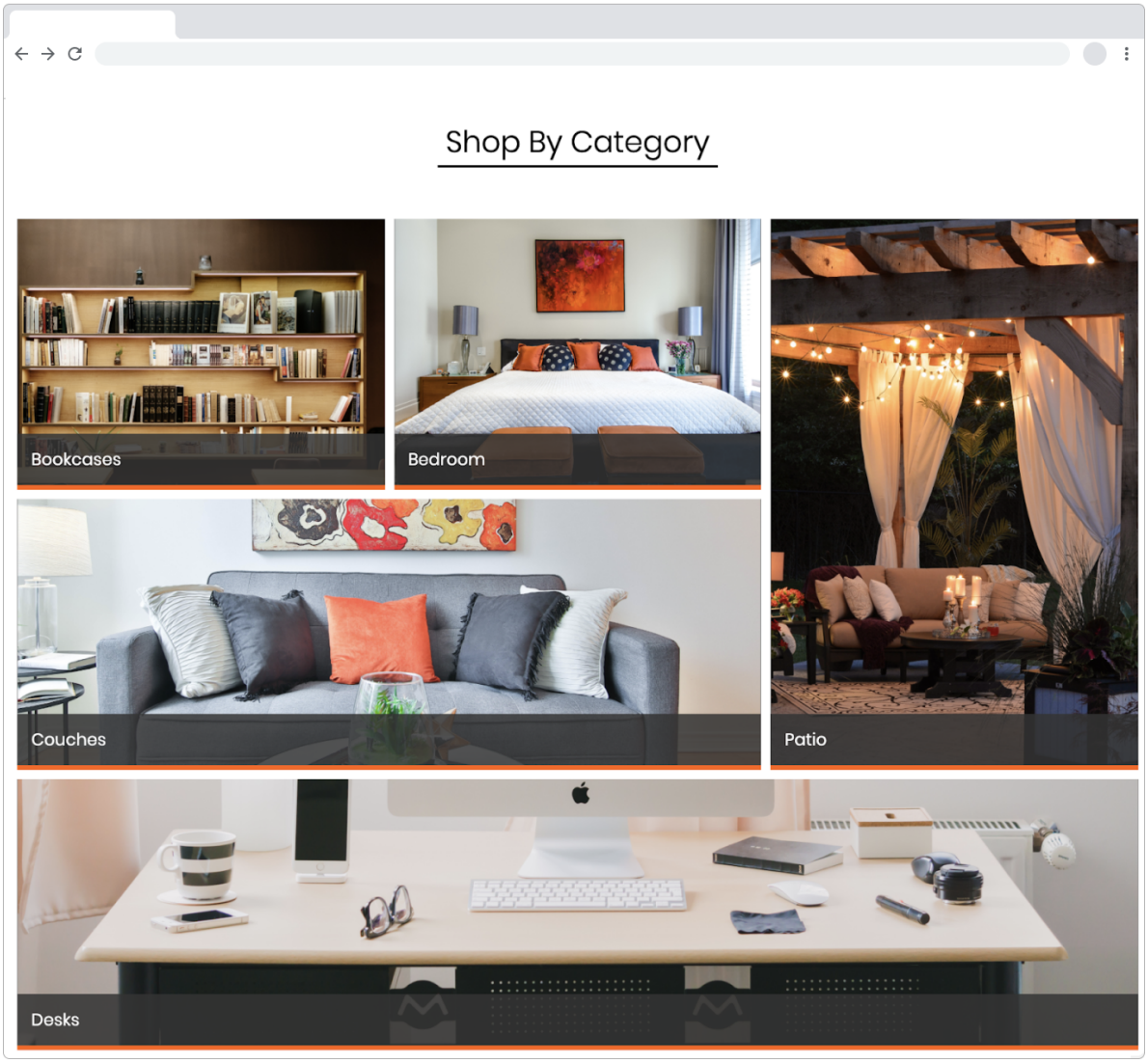
4. Notice the repeated .heroGridArea class on each grid area in <> index.html. You can use CSS to target all instances of that class and style each instance the same.
    - Create a CSS selector that targets the .heroGridArea class and add the following CSS properties:

```
background-position: center;
background-repeat: no-repeat;
background-size: cover;
align-items: center;
display: flex;
position: relative;
```

Step 3: Add Some Polish

1. If you were to open <> index.html in your browser, the hero image portion would almost look complete except for the fact that the headline isn't in the right place.
2. In order to align the headline , add a border: 5px solid #FF6200;.
3. Next, add a transparent background-color:.
    - You could do this with a transparent image or you can use the rgba() function to define red, green, blue, and alpha (opacity) values for a color.
    - Set the value for background-color to rgba(255,255,255,0.8);.
    - This will set the background color to white at 80% opacity.
4. Position the headline so that it's floating above and between the featuredProduct and secondaryProduct grid areas.
    - ***Hint**: You will need to use position: absolute; along with top and right values to achieve this.*
5. Add a z-index value to move the headline above its parent and then add padding to create some spacing.

Next, you will add a second grid, both HTML and CSS, to your webpage to showcase the different product types of your online furniture store. See below for a preview.



Part 2

1. Before creating our second grid, add a `<div>` with the class `subHeading`.
    - Create a `<h3>` tag and add the text “Shop By Category” within this `<div>`.
2. This might already be in the HTML - if it is, go ahead to Step 1.

Step 1: Create the Grid and its Content

1. Next, add a `<section>` tag and give it the ID `categoryGrid`.
2. Create the following HTML structure inside your newly created `<section>` tag:

```
<section id="categoryGrid">
  <div id="bookcases" class="categoryGridArea">
    <div class="heading">Bookcases</div>
  </div>
  <div id="bedroom" class="categoryGridArea">
    <div class="heading">Bedroom</div>
  </div>
  <div id="couches" class="categoryGridArea">
    <div class="heading">Couches</div>
  </div>
  <div id="patio" class="categoryGridArea">
    <div class="heading">Patio</div>
  </div>
  <div id="desks" class="categoryGridArea">
    <div class="heading">Desks</div>
  </div>
</section>
```

3. That's all the HTML structure you will need for your second grid.
4. This might already be in your HTML - if it is, go ahead to Step 2.

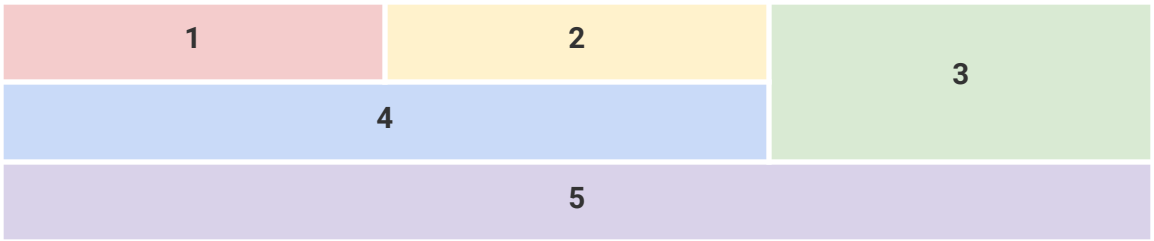
Step 2: Format and Style the Grid

1. Next, open `style.css`.
    - Again, you'll also find this file in the .zip file within the Activities folder.
2. Create a selector that targets the `subHeading` class. Add the following:

Property	Value
`text-align`	`center`
`border-bottom`	`2px solid black`
`width`	`300px`
`margin`	`50px auto`

3. Create a selector that targets your second grid using its ID.
    - Add `margin: 0px 5%;` and `display: grid;` to create your grid.

4. Use the `grid-template-rows`, `grid-template-columns`, and `grid-template-areas` properties to create the structure of the grid.
    - You will have three columns and three rows in this layout (note the illustration below).
    - Make your row `300px` tall.
    - Each column should occupy one-third of the area (use fractional units).
    - The area should look like the diagram below:



5. Next, create **five** selectors that target each grid area you created in `index.html`.
    - *Hint: Remember to target the **ID** for each area.*
6. Add a `background-image` to each grid area selector using the images provided in the `images` folder.
7. Create a selector for the common grid area class `categoryGridArea`.
    - Add the following properties and values to this selector:

```
background-position: center;
background-size: cover;
background-repeat: no-repeat;
margin: 5px;
cursor: pointer;
display: grid;
align-content: end;
```

8. Lastly, create a selector to target the `heading` class with the following:

```
background-color: rgba(0,0,0,0.7);
padding: 15px;
height: 25px;
color: white;
font-size: 18px;
border-bottom: 5px solid #FF6200;
```

9. Now, open `index.html` in your browser and take a look at both grids.
10. You should notice that you've created a great-looking layout using nested grids!

Part 3

One of the final things you will do is to make your furniture layout responsive for mobile viewers.

Step 1: Make Our Navigation Responsive

1. Open the `index.html` and `style.css` files.
    - You can find the files in the .zip file within the activities folder.
2. Write a media query that targets a screensize with the max-width of 1000px.

> **Note:** You must write your CSS selectors *within the media query*. You're doing it right if the selectors you're writing are {between the media query brackets}, not outside them.

Correct selector placement:

```
@media only screen and (max-width: 1000px) {
// Your new css selectors go here
}
```

3. In your newly written media query, write a CSS selector that targets the id of navigation:
    - In your CSS selector, give it the property display with a value of none. Write a CSS selector that targets the class of icons.
    - Give our CSS selector the property margin-left with a value of auto.

Step 2: Make Your Grid Responsive

1. Write a media query that targets a screensize with the max-width of 600px.
2. Write a CSS selector that targets the id of headline:
    - Give your selector the CSS property of position with a value of static.
    - Give your selector the CSS property of margin with a value of 20px.
3. Write a CSS selector that targets the id of heroGrid:
    - Give your CSS selector the CSS property of grid-template-areas.
        i. Rewrite the template areas so they display as follows:
            - “featuredProduct featuredProduct”
            - “secondaryProduct CTA”
        ii. This will cause our grid to restructure and display better on mobile devices.
4. Write a CSS selector that targets the id of categoryGrid:
    - Give your CSS selector the property of grid-template-rows with a value of 300px 300px 300px 300px 300px.
    - **Note:** You need to create more rows in your grid for your content to occupy. In the previous grid’s template areas (Step 3, above), you have two rows. You will be creating five rows in your adjusted grid-template-areas.
    - Give your CSS selector the property of grid-template-columns with a value of 1fr.
    - Give your CSS selector the CSS property of grid-template-areas:
        i. Rewrite the template areas so they display as follows:
            - "bookcases"
            - "bedroom"
            - "patio"
            - "couches"
            - "desks"

5. Refresh the page.
6. Check out your work!

Part 4

In order for your homework to be graded, you'll need to submit a link to your website, once you get it online. The final step of homework is to put your code on GitHub.

Follow the instructions on the 17.3 activity "Upload Your Webpage To the GitHub Cloud"

BONUS #1: Share Portfolio Progress

Share your progress on your portfolio website and/or case study slide decks to receive feedback. Feel free to share links to any or all of the following that you would like feedback on:

1. Case study slide deck of a project. Remember this is what you will be walking hiring managers through during interviews. Don't try to include everything, edit the attributes you include to tell your best story!
2. Figma frames of how you plan to design your various portfolio website pages.
3. Link to progress portfolio site along with figma link with screenshots of each page to make it easier to add comments to.

This is not required.

Required Deliverables

1. **A link to your working GitHub Pages website**
2. **BONUS:** Link to Portfolio Site

Submission

Submit a **link to your GitHub Pages website** to https://www.bootcampspot.com

Resources

Pull up these resources as you are coding to help enforce your code syntax and troubleshoot issues via documentation.

- MDN Flexbox basics
- MDN basic concepts of Grid layouts
- CSS grid-template-**rows** property
- CSS grid-template-**columns** property