Assignment 3: Advanced CSS Grid

The purpose of this assignment is to get practice using some of CSS grid's more advanced features and a few other miscellaneous CSS features.

I've provided a starter project; open this in Visual Studio Code and inspect the files to see how they work.

In **index.html,** you can immediately change the value of the **title** tag to include your own name. There's are also TODO tags indicating places where you will make changes later in the assignment. Besides these places, DO NOT edit the code in **index.html!** For all CSS selections, use the advanced selectors we learned and DO NOT add more classes or IDs.

Also, notice that there's a link to a CSS file called **my-styles.css,** a file that doesn't yet exist in the project. Go ahead and create this file now; this is where most of your code will be written.

This site will have both a small and large screen layout, so use a mobile-first CSS approach.

I've provided some pre-written styles in **style.css.** DO NOT modify this code except in two places where **TODO** comments have been placed. More information about this will be provided...

Task 1: Grid Children

... in this task! Above both TODO comments in **style.css**, you'll see that the **display** has been set to **grid.** As you know, CSS Grid containers always treat the **immediate, direct children** as elements in the grid layout. At both places, delete the ***** selector and replace with a selector that does the following:

- Selects only the immediate children of the container, NOT all descendants;
- EXCLUDES the h2 heading from the selection. Figure out how to do this using the **:not** pseudo-class: https://developer.mozilla.org/en-US/docs/Web/CSS/:not

Task 2: Font Awesome Icon

In **index.html**, find the TODO comment that marks where the site's logo is supposed to go. Using **Font Awesome,** find an appropriate game-themed icon to represent the logo and place it in the HTML code. Add the class **logo** to the element. If you're new to Font Awesome, a quick tutorial on how to do this can be found here: https://www.youtube.com/watch?v=vgAE5gMzEe8&t=292s

- When creating a Kit, you don't have to enter a **domain,** you can leave that field blank. (You can also skip all the personalization stuff, it doesn't really matter)
- You can stop watching the tutorial at 00:04:45.

Task 3: Header Grid

Let's first tackle the grid that controls the h1 heading, tagline, logo, and navigation menu. Implement the following:

- Small-screen layout:
  - The **height** of the header should be **80%** of the viewport height, but it should also have a **minimum height** of **20rem.** Implement this using the **vh** unit and **clamp** function.
  - All elements (headings, nav, etc.) should rest at the **bottom** of the header. Figure out how to do this using **CSS Grid's align** properties (such as align-content, align-items, or align-self.)
  - The navigation items should grow so that all together they fill the entire row, but wrap to a new row if the space gets too narrow; at the smallest screen size, they should occupy two or three rows. Figure out how to do this using the **repeat** function.
- Large-screen layout:
  - Using **grid-template-areas,** organize the layout of the header elements as in the screenshots and video.
    - The tagline should be confined to a column that is only as wide as the minimum width of the tagline's text. (The minimum width it can take without breaking the words.) Figure out how to do this using **grid-template-columns** and the **min-content** value.
    - Once again, make sure everything is sitting as low as possible in the header by using **align** properties.

Task 4: Introduction Grid

The next task is to set up the grid for the **Introduction** section.

- DO NOT use **grid-template-areas** for this. Instead, use the **repeat** function to create **six equally-sized** columns that respond naturally to the width of their container. Lay out the subsections and figure as shown in the screenshots and video.
- The **figure** itself should also be a **grid** and use the **subgrid** feature to take a column template that aligns with its parent: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout/Subgrid
    - Use advanced selectors to select the images within the **figure**.
- The **figcaption** should be placed over the images using the **overlapping grid** technique. (DO NOT use **position: absolute** for this, even though it could be a good solution.)
    - Use advanced selectors to select the images within the **figure**.
- Notice in the screenshots that the headings of the subsections are pretty close to the description text (the descenders of the "Types" heading actually overlap with the text below.) Figure out how to implement this using a **negative margin value.**

Task 5: List of Games Grid

The **List of Games** section should use a grid with the following properties:

- Items grow or shrink in response to resizing the viewport BUT don't necessarily fill the entire row; empty cells may appear at the end of the row if there's room for them.
- If the items get too narrow, they wrap to a new row.

Task 6: Attributions Grid

- The list of attributions in the footer should, in the large-screen layout, use a **grid** with **five** columns, each sized to fit the **maximum width** of their content. The column width should be **fixed** and the items should NOT wrap. Figure out how to do this using the **repeat** function!
- The items should be **justified** to the **right side** using the appropriate **CSS Grid** property.
- Use the **calc** function to make sure the space between each grid item is exactly **2x** the width of the **--small-space** variable.

Grading

- Task 1
  - [1 mark] Selector using :not
- Task 2
  - [1 mark] Font Awesome icon
- Task 3
  - [1 mark] Responsive height with clamp and vh
  - [1 mark] Grid aligned to bottom
  - [1 mark] Navigation is responsive as described
  - [1 mark] Grid areas used
  - [1 mark] Tagline column sized properly
- Task 4
  - [1 mark] Subgrid used
  - [1 mark] Overlapping grid elements for figcaption
  - [1 mark] Negative margin for text position
- Task 5
  - [1 mark] Grid is responsive and described
- Task 6
  - [1 mark] Grid layout as described
  - [1 mark] Calc used effectively for spacing between items

Total: 13

As usual, up to -25% may be deducted for improper hand in, poor organization, poorly-formatted code, etc. Please ask me if in doubt.

Hand In

Remove all unnecessary files from the project folder (such as the screenshots, video, and these instructions.) As usual, rename the folder to **a3-firstname-lastname,** Zip the folder, and hand in to Brightspace.