Assignment 4: CSS Flexbox

The purpose of this assignment is to practice making layouts with CSS Flexbox, particularly layouts that would be difficult or impossible to make with CSS Grid.

As usual, I've provided starter files with some pre-written code and screenshots/videos to show how to site should look when it's done. DO NOT edit the code in **index.html** or **style.css**. (Except to change the **title** tag to have your own name.) All code for this assignment should be written in **my-styles.css.**

As you can see, the images are placed in the document dynamically by a JavaScript at the end of **index.html.** You don't have to understand how this script works! Use the **Inspect** tool in the browser to see the HTML structure of the final rendered document. Each time you reload the page, a new set of random images will be populated.

- *Note: the images are fetched from a free API, and there's a small chance your IP address may get temporarily blocked if you refresh the page too rapidly. Or the IP may also get blocked if too many students are working on the assignment at the same time in the Langara lab, but we'll cross that bridge when we get there.*

As we're experimenting with an intrinsic design approach in this class, the stylesheet should NOT be organized with the traditional small/medium/large media query breakpoints. (Even though that's how I organized the demo screenshots.) That said, the layout we're trying to achieve cannot currently be done without media queries; we'll use them in a slightly different way than we have in past assignments.

I've provided a few variables; use these as necessary in your CSS code.

Task 1: Navigation Menu

The navigation menu should have the following layout:

- Items should be aligned horizontally in a row with a small gap between each; they should NOT wrap to new rows.
- The items should be ordered from right to left, opposite of English reading order. *However,* the item "Attributions" should be placed in the right-most spot. Your solution should use the **order** attribute ***only once***.
- When there's extra space in the container (when the viewport is wide) the "The Art of Photography" item should fill all the extra space. The other items should have a fixed width.
- When there is not enough space for all the items to fit at their default width (when the viewport is narrow) all the items should shrink proportionally, but the "Attributions" item should shrink more than the others.

Task 2: Attributions and Other Links

The items in the **aside** should have the following layout:

- They should be laid out in a row with all items growing horizontally by equal amounts to fill the space and no gaps between.
- When there's not enough room for all the items to fit on one row at their default width, they should wrap to a new row.
- As the items shrink, the height of each item should grow *upwards* to accommodate its text contents, but not more than it has to.

Task 3: Photography Gallery

This part of the page should have the following layout:

- Each photo has a different width, but they should all grow to fill the space in the row with no gaps at the edges. When they won't all fit in a row, they should wrap to a new row.
- There should be a small amount of space between each item.
- When wrapping causes the last row to have only one image, **figure out** a way to prevent that image from stretching across the whole row. (Note that there's currently no perfect solution to this problem; you'll see why when you try. Just do your best.)

Task 4: The Art of Photography

As you can see from the screenshot and video, the layout for this part should somewhat resemble a ***masonry*** layout, which means there are clearly defined vertical grid lines but mismatched horizontal alignment between items. There's currently no easy way to achieve this layout with CSS, but we can fake it with Flexbox (as long as we feel okay about using some hacky code).

Note: a native CSS system for making masonry layouts has been in the works for a long time, but for some reason its release has been significantly delayed.

Here's a rough guide to how the layout should work:

- The **width** of the columns will have to be set manually to be 1/3 of the container width. (This should result in three columns.) You can use the **gap** property to set a gap between the items, both horizontally and vertically.
- The top and bottom of the container area should have a ragged layout; there should NOT be alignment between the tops or bottoms of the columns. However, the vertical distance between items within a column should be uniform.

- The tricky thing about this layout is that when you reduce the width of the viewport, the columns will eventually wrap to a ***fourth row,*** with will appear outside the container and off the screen entirely! This happens because the items get taller as they get narrower (because all the text has to fit inside) but the container itself does *not* get taller! As far as I can tell, there is no way to have the container's height automatically grow as the viewport gets narrower. So, you'll have to do it with media queries, but the good news is that with CSS's new nesting capabilities, you can simply add media queries inside the style block, like this:

  height: ***original height***

  @media screen and (max-width: ***a bit narrower***) {
  height: ***a bit taller***
  }
  @media screen and (max-width: ***even narrower still***) {
  height: ***even taller still***
  }
- When the viewport narrows to a certain width the layout should switch to **two columns** instead of three. For very narrow viewports, **only one column** should be used.
- Hint: when you're testing the layout, keep an eye on the **"Art"** section. This is the last item in the flex container, so if it wraps outside the container, you won't be able to see it anymore. Make sure you can see it always, regardless of the viewport width.

Grading

- All tasks
  - [1 mark] appropriate use of given variables; space between grid items matches screenshots and video
- Task 1
  - [1 mark] items ordered properly
  - [2 marks] items grow and shrink properly
- Task 2
  - [1 mark] items are properly vertically aligned
  - [1 mark] items wrap properly
- Task 3
  - [1 mark] items resize themselves to fill each row
  - [1 mark] last item does *not* fill the row when its alone
- Task 4
  - [3 marks] width set for three, two, and one-column layouts
  - [1 mark] ragged top and bottom of container area
  - [1 mark] no overflow from container

Total: 13

As usual, up to -25% may be deducted for improper hand in, disorganized work, etc.

Hand In

Rename the project folder to **a4-firstname-lastname** and remove unnecessary files like the screenshots, these instructions, etc. Zip the folder and hand in to Brightspace.