M1C2 Flexbox Assignment

During the second section of Module 1, you will have reviewed the very helpful Flexbox tool. Flexbox is another useful tool that developers can use to control the behavior and layout of their HTML elements. You first used Flexbox in the Coding Foundations course. Now that you have reviewed and learned more about Flexbox, it is time to practice! You will use HTML and CSS to complete this assignment. It is expected that you will create and edit files using a text editor and that you will use an external stylesheet. Please complete the following assignment and reach out on the Support App to have a mentor review your work. If you have any questions or need any help, please reach out so we can help you! This assignment must be completed to pass this section of the coursework.

The following is a link to a git repository that contains the starter HTML and CSS for this assignment. Please copy and paste the starting code so you can follow along with the assignment. https://github.com/bottegaMentor/m1c2-flexbox--assignment/tree/main

*Concepts Included in this Assignment: HTML, CSS, Flexbox, HTML Classes, HTML IDs, HTML Div, Img,h1, and Paragraph Tags, CSS Background-Color, CSS Text Color, CSS External Stylesheets and Inline Styling, Flex Direction, Wrap, Basis, Grow, and Shrink, Justify Content, Align Content, Align Self, and Align Items.
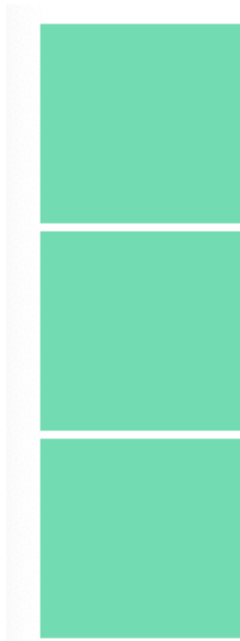
Flexbox 1: Select the class of flexOne. Use flexbox to position the elements in the flexOne div next to each other. Add 5 pixels of margin to the square class only when they are inside flexOne.

Example:

Flexbox 2: Select the class of flexTwo. Use flexbox. Set flex-direction to column. The default is row, which places the elements beside each other. Column puts them back on top of each other. Give the squares inside of flexTwo a margin of 2 pixels.

Example:

Flexbox 3: Select the class of flexThree. Repeat the steps from flexTwo, except this time you are going to set flex-direction to row-reverse. Add a 1, 2, and 3 inside of the respective square divs inside of flexThree so that you can see how the order has now become reversed.

Example:

Flexbox 4: Select the class of flexFour. Use flexbox. Set the div's width to 350px. Give a margin of 5px to the squares inside of flexFour. Set flex-flow to "wrap". Also change the background color to #3d8b6d. You will see that the fourth square automatically wraps to the bottom and starts a new row because it cannot fit in the 450 pixels with the other squares. This would be very useful for a website like Instagram, when you know the format you want for your images or posts but don't know how many you will have. Using flex-wrap, your code automatically handles formatting your elements.

Example:

Flexbox 5: Select the class of flexFive. Add a width of 350px and use flexbox. Select the item class for the divs inside of flexFive. Set the background-color to #0b6643. Add a width and height of 100px. Add a margin of 2px so we can see the individual squares. Now, target the id of "grow". On the id, set flex-grow to 2. This will make it so the leftover space, which would normally be divided equally between the elements or unused by flex, will be applied to the middle item and make it take up twice the extra space than the other elements do.

Example:

Flexbox 6: Select the class of flexSix. Add a height of 400px and a width of 500px to flexSix. Set the background color to #f1f1f1. Set the flex-direction to column so that the child divs will be back on top of each other. On each child div, with the classes of one, two, and three, use flexbox. On one, set justify-content to space-around. On two, set justify-content to space-between, set margin to "50px 0", and set the background-color to #d6d6d6. The color will help us see the rows we are creating. On three, set justify-content to space-evenly. Now you can see the different ways that those justify-content settings will behave.

Example:

Flexbox 7: Select the class of flexSeven. Give it a height of 400px and a width of 500px. Use flexbox and set justify-content to space-between. This will give us 4 columns. We will leave the flex-direction on the default of row and use justify-content to move the elements in a different way. On divs one through four, use flexbox and set the flex-direction to column so that the elements remain on top of each other. On div one, we can set justify-content to center. Two to flex-end, three to flex-start, and four to space-between. Now you can see how we can also arrange items vertically using justify-content if we also set flex-direction to column.

Example:
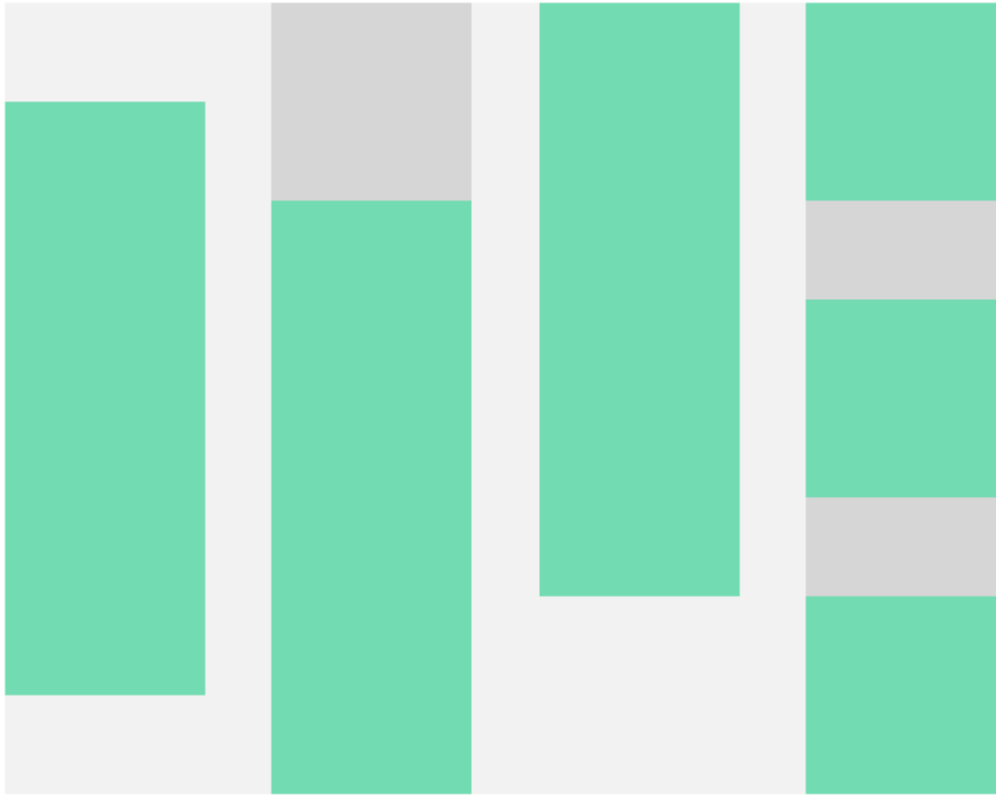
Flexbox 8: Select the class of flexEight. Use flexbox. Give the div a height of 400px and set the backgound-color to #eeeeee. Give it a width of 300px. Now you can select the class of one inside of flexEight. Use flexbox and set align-items to center. Do the same for two and set align-items to flex-end. Set three to the same things and use flex-start for align-items. Now you can see that align items can also let us move elements vertically.

Example:
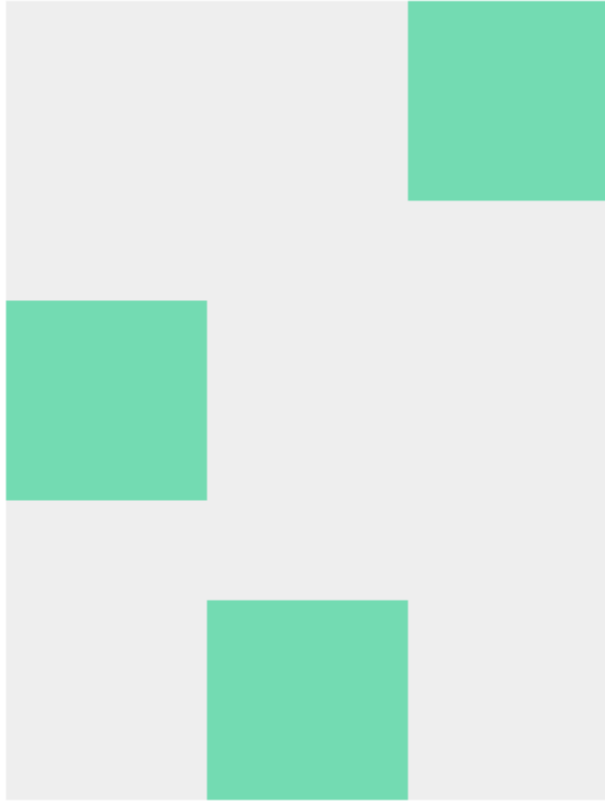
Flexbox 9: Select the class of flexNine. Use flexbox. Set the width to 450px and the height to 250px. Set the background color to #eeeeee. Use align-items to make the elements start at the top. Use justify-content to space the elements evenly. Now you can select the special id. Use the property align-self and set it to flex-end. This way we can override the 3rd square and make it start at the bottom instead of the top.

Example: