Step-by-Step Learning Activity

a) Creating a flexbox using the display property

***Key takeaway**: Use display: flex; to create a flexbox responsive layout.*

display: flex;

By declaring the display to be the flex value a flexbox is created. The flexbox styles allow you to easily create various responsive layouts without having to us the float property.

b) Setting the Flexbox direction

***Key takeaway**: Use the flex-direction property to set the main axis to be either horizontal (single row of columns) or vertical (single column of rows).*

***flex-direction: row;** – sets the main axis to be a single row of columns*

***flex-direction: row-reverse;** – sets the main axis to be a single row of columns that runs in the reverse direction*

***flex-direction: column;** – sets the main axis to be a single column of rows*

***flex-direction: column-reverse;** – sets the main axis to be a single column of rows that runs in the reverse direction*

The flexbox-direction property sets the main axis or how the items will be laid out either horizontally or vertically.

c) Setting the flexbox wrap property

***Key takeaway**: Use the flex-wrap property to specify whether items can continue on the next line, if it is not set, the items will shrink in size.*

flex-wrap: wrap;

flex-wrap: nowrap;

flex-wrap: wrap-reverse;

The flex-wrap property declares if the items can wrap to the next line if they run out of room. The default value id nowrap.

d) Setting the flex-flow property

***Key takeaway**: Use the flex-flow property to declare both the flex-direction and the wrap properties.*

flex-flow: row wrap;

flex-flow: row nowrap;

The flex-flow property lets you declare both the direction and the wrap properties.

e) Setting the flex-grow property

Key takeaway*: Use the flex-grow property to set the ratio of how you want items to grow in relation to each other.*

flex-grow: 1; *– if all items were set to 1 they would grow equally*

flex-grow: 2; *– if all items were set to 1 and one of them were set to 2 the items set to 1 would grow equally and the 2 would grow at 2 times the rate of the others.*

The flex-grow property declares how the item will grow in relation to the other items in the container.

f) Setting the flex-shrink property

Key takeaway*: Use the flex-shrink property to set the ratio of how you want items to shrink in relation to each other.*

flex-shrink: 1; *– if all items were set to 1 they would shrink equally*

flex-shrink: 2; *– if all items were set to 1 and one of them were set to 2 the items set to 1 would shrink equally and the 2 would shrink at 2 times the rate of the others.*

The flex-shrink property declares how the item will shrink in relation to the other items in the container. Items that have a higher shrink value will shrink more than items with a lower shrink value.

g) Setting the flex-basis property

Key takeaway*: Use the flex-basis to set the initial value of a flex item.*

flex-basis: 100px;

flex-basis: 60%;

The flex-basis property declares the size of the item.

h) Create a new HTML page

Key takeaway*: Review how to create a HTML page from scratch.*

Step 1: Downloaded the assets folder and place it in an appropriate place on your computer.

Step 2: Create a new HTML file

```
<!doctype html>
<html lang="en">
    <head>
        <title> Responsive Design - Flexbox </title>
    </head>
    <body>


    </body>
```

```
</html>
```

Step 3: Save the file as flex.html

i) Create a new CSS page

***Key takeaway**: Review how to create a CSS stylesheet from scratch.*

Step 1: Create a new CSS file

Step 2: Set the charset to " UTF-8";

```
@charset "UTF-8";
```

Step 3: Declare the document to be CSS

```
@charset "UTF-8";
/* CSS Document */
```

Step 4: Add global styles such as font-family and box-size to ensure that the width/height are calculated in the width of elements

```
* {
    font-family: Arial, "sans-serif";
        box-sizing: border-box;
}
```

Step 5: Add body styles such as setting the margins to 0 to override browser defaults

```
body {
        margin: 0px;
}
```

Step 6: Add comments to organize your stylesheet

```
/* Flex 1: main axis is set to row/row-reverse */
/* Flex 2: main axis is set to column/column-reverse */
```

Step 7: Save the flex.css file.

j) Attach the external stylesheet to the HTML page

***Key takeaway**: Review how to attach a CSS stylesheet to a HTML page.*

Step 1: Return to the flex.html file. In the head tag, add the link tag

```
<link>
```

Step 2: Specify the relationship to the web page

```
<link rel="stylesheet">
```

Step 3: Specify the type of file that you are linking to

```
<link rel="stylesheet" type="text/css">
```

Step 4: Specify the URL to the file you are linking to

```
<link rel="stylesheet" type="text/css" href="css/flex.css">
```

Step 5: Save the flex.html file.

k) Add the first flexbox section

***Key takeaway**: Practice adding divs to a HTML page.*

Step 1: In the HTML document, add a h1 tag and add Grid Layouts as the content

```
<h1>Flexbox Layouts</h1>
```

Step 2: Add the h2 tag

```
<h2>Example 1: Flex Direction Row</h2>
```

Step 3: Add a div container

```
<div>
</div>
```

Step 4: Add 6 divs inside the previous div container, name and number the content within the divs

```
<div>
    <div>Flex item 1</div>
    <div>Flex item 2</div>
    <div>Flex item 3</div>
    <div>Flex item 4</div>
    <div>Flex item 5</div>
    <div>Flex item 6</div>
</div>
```

Step 5: Save the flex.html file.

Step 6: Preview the flex.html page in the live server.

Step 7: Switch back to HTML document to proceed to next segment.

l) Create a .flex1-container class and add a background-color, height and padding

***Key takeaway**: Practice creating classes and attaching them in HTML.*

Step 1: In the HTML document, add class to the main div, name it flex1-container.

```
<div class="flex1-container">
</div>
```

Step 2: Save the flex.html file.

Step 3: In the CSS document, add the new class below the grid 1 comment

```
.flex1-container {
}
```

Step 4: Set the background property to a grey using hex values

```
.flex1-container {
    background:#58595b;
}
```

Step 5: Set the height property to 500px

```
.flex1-container {
    background: #58595b;
    padding: 10px;
    height: 500px;
}
```

Step 6: Save the flex.css file.

Step 7: Preview the flex.html page in the web browser.

Step 8: Switch back to HTML document to proceed to next segment.

m) Create a .flex-item class and define common properties shared between the div items

***Key takeaway**: Create a class that defines common elements for all the div items to ensure your code is efficient and loads quickly.*

Step 1: In the HTML document, add a new class attribute called flex-item to the divs inside the container div.

```
<div class="flex1-container">
    <div class="flex-item">Grid item 1</div>
    <div class="flex-item">Grid item 2</div>
    <div class="flex-item">Grid item 3</div>
    <div class="flex-item">Grid item 4</div>
    <div class="flex-item">Grid item 5</div>
    <div class="flex-item">Grid item 6</div>
</div>
```

Step 2: Save the flex.html file.

Step 3: In the CSS document, add a new class above the grid one comment. (this will be applied to more than this example so it should be kept with general styles).

```
.flex-item {
}
```

Step 4: Add a background-color property and set it to grey

```
.flex-item {
    background-color: #DCDCDC;
}
```

Step 5: Add a margin and set the top and bottom to be 10px and the right and left to be 5px

```
.flex-item {
```

```
        background-color: #DCDCDC;
        margin: 10px 5px;
    }
```

Step 6: Add the padding property, set the value to be 10px

```
    .flex-item {
        background-color: #DCDCDC;
        margin: 10px 5px;
        padding: 10px;
    }
```

Step 7: Add a text align property and set it to center

```
    .flex-item {
        background-color: #DCDCDC;
        margin: 10px 5px;
        padding: 10px;
        text-align: center;
    }
```

Step 8: Save the flex.css file.

Step 9: Preview the flex.html page in the live server.

Step 10: Switch back to HTML document to proceed to next segment.

n) Create a unique .flex1-item# class for each div

Step 1: In the HTML document, add a new class to each of the div items.

```
    <div class="flex1-container">
        <div class="flex-item flex1-item1">Grid item 1</div>
        <div class="flex-item flex1-item2">Grid item 2</div>
        <div class="flex-item flex1-item3">Grid item 3</div>
        <div class="flex-item flex1-item4">Grid item 4</div>
        <div class="flex-item flex1-item5">Grid item 5</div>
        <div class="flex-item flex1-item6">Grid item 6</div>
    </div>
```

Step 2: Save the flex.html file.

Step 3: In the CSS document, add the unique item classes

```
    .flex1-item1 {
    }
    .flex1-item2 {
    }
    .flex1-item3 {
```

```
}
.flex1-item4 {
}
.flex1-item5 {
}
.flex1-item6 {
}
```

Step 4: Save the flex.css file.

Step 5: Preview the flex.html page in the live server.

Step 6: Switch back to HTML document to proceed to next segment.

o) Define the .flex1-container as a flexbox using the display property

***Key takeaway**: Use display: flex; to define a div as a flexbox.*

Step 1: In the CSS document, add the display property and set its value to be a grid

```
.flex1-container {
    background: #58595b;
    height: 500px;
    display: flex;
}
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

p) Define main axis using flex-direction – row

***Key takeaway**: Use the flex-direction property to set the main axis to be either horizontal (single row of columns) or vertical (single column of rows).*

Step 1: In the CSS document, add the flex-direction property and set it to row

```
.flex1-container {
    background:#58595b;
    height: 500px;
    display: flex;
    flex-direction: row;
}
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

q) Define whether the flexbox can wrap to a new line using flex-wrap

***Key takeaway**: Use the flex-wrap property to specify whether items can continue on the next line, if it is not set, the items will shrink in size.*

Step 1: In the CSS document, add the flex-wrap property to the .flex1-container and set it to wrap

```
.flex1-container {
    background:#58595b;
    height: 500px;
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
}
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

r) Define how to align items horizontally in a flex container

***Key takeaway**: Use the justify-content property to declare how the items are distributed within the container when there is extra horizontal space.*

Note that if the flex container is set to grow the items will grow to fill the space, negating the justify-content property.

***justify-content: space-evenly;** – the items are spaced evenly in the container*

***justify-content: space-around;** – the items are spaced evenly in the container*

***justify-content: space-between;** – the outer items are aligned on the outside edge of the container and remaining items have equal space between the outer items and the inner items in the container*

***justify-content: center;** – the items are centered inside the container separated by the grid-gap space*

***justify-content: start;** – the items are left justified inside the container separated by the grid-gap space*

***justify-content: end;** the items are right justified inside the container separated by the grid-gap space*

The justify-content property defines how the items are distributed within the container when there is extra horizontal space.

Note that if the flex container is set to grow the items will grow to fill the space, negating the justify-content property.

Step 1: In the CSS document, add the justify-content property to the .flex1-container and set it to center

```
.flex1-container {
    background:#58595b;
```

```
        height: 500px;
        display: flex;
        flex-direction: row;
        flex-wrap: wrap;
        justify-content: center;
    }
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

s) Define how to align items vertically in a flex container

***Key takeaway**: Use the align-content property to declare how the items are distributed vertically within the container when there is extra vertical space.*

Note you must specify a height in order to see this in action. The default is to let the item stretch to fill the container.

***align-content: space-evenly;** – the vertical spacing of items are spread out evenly in the container*

***align-content: space-around;** – the vertical spacing of items are spread out evenly in the container*

***align-content: space-between;** – the outer items are aligned on the top and bottom edges of the container and remaining items have equal space between the outer items and the inner items in the container*

***align-content: center;** – the vertical spacing of items are centered inside the container separated by the grid-gap space*

***align-content: start;** – the vertical spacing of items are left justified inside the container separated by the grid-gap space*

***align-content: end;** – the vertical spacing of items are right justified inside the container separated by the grid-gap space*

The align-content property defines how the items are distributed vertically within the container when there is extra vertical space.

Note you must specify a height in order to see this in action. The default is to let the item stretch to fill the container.

Step 1: In the CSS document, add the justify-content property to the .flex1-container and set it to center

```
.flex1-container {
    background:#58595b;
    height: 500px;
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: center;
```

```
        align-items: center;
    }
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

t) Define how a flex item grows using flex-grow

***Key takeaway**: Use the flex-grow property to set the ratio of how you want items to grow in relation to each other.*

Step 1: In the CSS document, add the flex-grow property to the .flex1-item1 and set it to 1

```
    .flex1-item1 {
        flex-grow: 1;
    }
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

u) Define initial value of a flex item using flex-basis

***Key takeaway**: Use the flex-basis to set the initial value of a flex item.*

Step 1: In the CSS document, add the flex-basis property to the .flex1-item6 and set it to 1

```
    .flex1-item6 {
        flex-basis: 250px;
    }
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

v) Define how a flex item shrinks using flex-shrink

***Key takeaway**: Use the flex-shrink property to set the ratio of how you want items to shrink in relation to each other.*

Step 1: In the CSS document, add the flex-shrink property to the .flex1-item6 and set it to 0

```
    .flex1-item6 {
        flex-basis: 250px;
        flex-shrink: 0;
```

```
}
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

w) Define main axis using flex-direction – column

***Key takeaway**: Use the flex-direction property to set the main axis to be either horizontal (single row of columns) or vertical (single column of rows).*

x) Setting the flex property – shorthand for basis, grow, shrink

***Key takeaway**: Use flex as shorthand for defining flex-grow, flex-shrink and flex basis.*

flex: 1 1 60%;

flex is shorthand for flex-grow, flex-shrink and flex basis. 1 indicates equal growing or shrinking and the basis sets the initial height to 60%.

y) Setting the order property

***Key takeaway**: Use the order property to reorder the flex items. All items in the container must be redefined for this property to function.*

Order: 1;

The order property allows you to change the order of the items. All items in the container must be redefined for this property to function.

z) Add the second flexbox section

Step 1: In the HTML document, add the h2 tag below the previous div container

```
<h2>Example 2: Flex Direction Column</h2>
```

Step 2: Add a div container

```
<div>
</div>
```

Step 3: Add 4 divs inside the previous div container, name and number the content within the divs

```
<div>
    <div>Flex item 1</div>
    <div>Flex item 2</div>
    <div>Flex item 3</div>
    <div>Flex item 4</div>
</div>
```

Step 4: Save the flex.html file.

Step 5: Preview the flex.html page in the live server.

Step 6: Switch back to HTML document to proceed to next segment.

aa) Create a .flex2-container class and add a background-color and height

Step 1: In the HTML document, add class to the main div, name it flex1-container.

```
<div class="flex2-container">
</div>
```

Step 2: Save the flex.html file.

Step 3: In the CSS document, add the new class below the grid 1 comment

```
.flex2-container {
}
```

Step 4: Set the background property to a grey using hex values

```
.flex2-container {
    background:#58595b;
}
```

Step 5: Set the height property to 700px

```
.flex2-container {
    background: #58595b;
    height: 700px;
}
```

Step 6: Save the flex.css file.

Step 7: Preview the flex.html page in the live server.

Step 8: Switch back to HTML document to proceed to next segment.

ab) Create a .flex-c-item class and define common properties shared between the div items

Step 1: In the HTML document, add a new class attribute called flex-item to the divs inside the container div.

```
<div class="flex1-container">
    <div class="flex-c-item">Grid item 1</div>
    <div class="flex-c-item">Grid item 2</div>
    <div class="flex-c-item">Grid item 3</div>
    <div class="flex-c-item">Grid item 4</div>
    <div class="flex-c-item">Grid item 5</div>
    <div class="flex-c-item">Grid item 6</div>
</div>
```

Step 2: Save the flex.html file.

Step 3: In the CSS document, add a new class above the the grid one comment. (this will be applied to more than this example so it should be kept with general styles).

```
.flex-c-item {
}
```

Step 4: Add a background-color property and set it to grey

```
.flex-c-item {
    background-color: #DCDCDC;
}
```

Step 5: Add a margin and set the top and bottom to be 10px and the right and left to be 5px

```
.flex-c-item {
    background-color: #DCDCDC;
    margin: 5px 10px;
}
```

Step 6: Add the padding property, set the value to be 10px

```
.flex-c-item {
    background-color: #DCDCDC;
    margin: 5px 10px;
    padding: 30px;
}
```

Step 7: Add a text align property and set it to center

```
.flex-c-item {
    background-color: #DCDCDC;
    margin: 5px 10px;
    padding: 30px;
    text-align: center;
}
```

Step 8: Save the flex.css file.

Step 9: Preview the flex.html page in the live server.

Step 10: Switch back to HTML document to proceed to next segment.

ac) Create a unique .flex2-item# class for each div

Step 1: In the HTML document, add a new class to each of the div items.

```
<div class="flex2-container">
    <div class="flex-c-item flex2-item1">Grid item 1</div>
    <div class="flex-c-item flex2-item2">Grid item 2</div>
    <div class="flex-c-item flex2-item3">Grid item 3</div>
    <div class="flex-c-item flex2-item4">Grid item 4</div>
```

```
    </div>
```

Step 2: Save the flex.html file.

Step 3: In the CSS document, add the unique item classes

```
.flex2-item1 {
}
.flex2-item2 {
}
.flex2-item3 {
}
.flex2-item4 {
}
```

Step 4: Save the flex.css file.

Step 5: Preview the flex.html page in the live server.

Step 6: Switch back to HTML document to proceed to next segment.

ad) Define the .flex2-container as a flexbox using the display property

***Key takeaway**: Use display: flex; to define a div as a flexbox.*

Step 1: In the CSS document, add the display property and set its value to be a grid

```
.flex2-container {
    background: #58595b;
    height: 700px;
    display: flex;
}
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

ae) Define main axis using flex-direction – column

***Key takeaway**: Use the flex-direction property to set the main axis to be either horizontal (single row of columns) or vertical (single column of rows).*

Step 1: In the CSS document, add the flex-direction property and set it to column

```
.flex2-container {
    background:#58595b;
    height: 700px;
    display: flex;
```

```
        flex-direction: column;
    }
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

af) Define the way the flex shrinks grows and basis using the flex shorthand

Step 1: In the CSS document, add the flex shorthand and set the values to 1 1 and 60%. 1 indicates equal growing or shrinking and the basis sets the initial height to 60%

```
    .flex2-item1 {
        flex: 1 1 60%;
    }
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.

ag) Re-order the items using the order property

Step 1: In the CSS document, add the order property to each item class, set the value to something other than its original value.

```
    .flex2-item1 {
        flex: 1 1 60%;
        order: 4;
    }
    .flex2-item2 {
        order: 1;
    }
    .flex2-item3 {
        order: 2;
    }
    .flex2-item4 {
        order: 3;
}
```

Step 2: Save the flex.css file.

Step 3: Preview the flex.html page in the live server.

Step 4: Switch back to HTML document to proceed to next segment.