Step-by-Step Learning Activity

a) Creating a grid using the display property

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

display: grid;

Defining the display property as a grid tells the browser to treat the div container as a grid.

b) Create a gap between grid items using gap

***Key takeaway**: Use the gap property to define the amount of space between grid items.*

***column-gap: value;** – column spacing*

***row-gap: value;** – row spacing*

***gap: columnvalue rowvalue;** – shorthand for both with different values*

***gap: value;** – shorthand for both if they have the same value*

The gird-gap property defines the of space between grid items both horizontally and vertically.

c) Create columns using grid-template-columns

***Key takeaway**: Use grid-template-columns to declare the number of columns and the width of each column.*

***grid-template-columns: auto auto auto;** – creates evenly spaced columns*

***grid-template-columns: 100px 200px 200px;** – creates defines values for the columns*

The grid-template-columns property specifies the number of columns in the grid along with the width of each container. To let items stretch to fill the space use auto as the value.

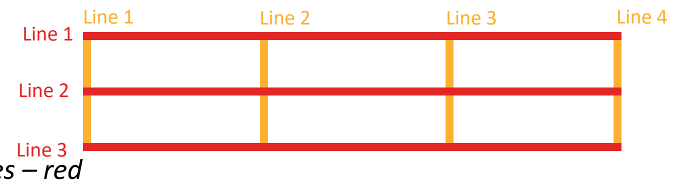
d) Create rows using grid-template-rows

***Key takeaway**: Use grid-template-rows to declare the height of each row.*

***grid-template-rows: 100px 200px;** – creates defines height values for the rows*

The grid-template-columns property declares the height of each item.

e) Grid lines



Column lines – red

Row lines - orange

***Key takeaway**: Use the gridline values to declare where an item will appear in a grid. Use gridline values with the following properties: grid-column-start, grid-column-end, grid-row-start, grid-row-end, grid-column, grid-row and grid-area.*

Grid lines numbers declare where an item sits in the grid using the following properties: grid-column-start, grid-column-end, grid-row-start, grid-row-end, grid-column, grid-row and grid-area.

f) grid-column-start and grid-column-end properties

***Key takeaway**: Use column lines to define where a column starts and ends with the grid-column-start, grid-column-end properties.*

***grid-column-start: 1;** – start at the first column line*

***grid-column-end: 3;** – end at the third column line*

grid-column-start, grid-column-end properties define where a grid item starts and ends using column lines.

g) grid-row-start and grid-row-end properties

***Key takeaway**: Use row lines to define where a row starts and ends with the grid-row-start, grid-row-end properties.*

***grid-row-start: 1;** – start at the first column line*

***grid-row-end: 3;** – end at the third column line*

grid-row-start, grid-row-end properties define where a grid item starts and ends using row lines.

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 - Grid</title>
    </head>
    <body>


    </body>
</html>
```

Step 3: Save the file as grid.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

```
/* Grid 1: grid-template-columns, grid lines */
/* Grid 2: grid-column/row/area */
/* Grid 3: Naming Grid items */
```

Step 7: Save the grid.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 grid.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/grid.css">
```

Step 5: Save the grid.html file.

k) Add the first grid 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>Grid Layouts</h1>
```

Step 2: Add the h2 tag

```
<h2>Example 1: grid-template-columns/rows understanding gridlines</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>Grid item 1</div>
    <div>Grid item 2</div>
    <div>Grid item 3</div>
    <div>Grid item 4</div>
    <div>Grid item 5</div>
    <div>Grid item 6</div>
</div>
```

Step 5: Save the grid.html file.

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

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

l) Create a .grid1-container class and add a background-color, height and padding

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

Add a class to define the parameters of the main div container. Add background, height and padding so the grid styles are easy to see.

Step 1: In the HTML document, add class to the main div, name it grid1-container.

```
<div class="grid1-container">
</div>
```

Step 2: Save the grid.html file.

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

```
.grid1-container {
}
```

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

```
.grid1-container {
        background: #58595b;
    }
```

Step 5: Set the padding property to 10px

```
    .grid1-container {
        background: #58595b;
        padding: 10px;
    }
```

Step 6: Set the height property to 400px

```
    .grid1-container {
        background: #58595b;
        padding: 10px;
        height: 400px;
    }
```

Step 7: Save the grid.css file.

Step 8: Preview the grid.html page in the live server.

Step 9: Switch back to HTML document to proceed to next segment.

m) Create a .grid-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.*

Create a class that defines common elements for all the div items. This reduces the amount of lines in your code which reduces the file size and makes it load faster in a browser.

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

```
    <div class="grid1-container">
        <div class="grid-item">Grid item 1</div>
        <div class="grid-item">Grid item 2</div>
        <div class="grid-item">Grid item 3</div>
        <div class="grid-item">Grid item 4</div>
        <div class="grid-item">Grid item 5</div>
        <div class="grid-item">Grid item 6</div>
    </div>
```

Step 2: Save the grid.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).

```
    .grid-item {
    }
```

Step 4: Add a text align property and set it to center

```
.grid-item {
    text-align: center;
}
```

Step 5: Add the padding property, set the value to be 30px

```
.grid-item {
    text-align: center;
    padding: 30px;
}
```

Step 6: Save the grid.css file.

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

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

n) Create a unique .grid-item# class for each div and define a different background color for each div

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

Add a class for each div item. Add different background color so the grid styles are easy to see.

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

```
<div class="grid1-container">
    <div class="grid-item grid1-item1">Grid item 1</div>
    <div class="grid-item grid1-item2">Grid item 2</div>
    <div class="grid-item grid1-item3">Grid item 3</div>
    <div class="grid-item grid1-item4">Grid item 4</div>
    <div class="grid-item grid1-item5">Grid item 5</div>
    <div class="grid-item grid1-item6">Grid item 6</div>
</div>
```

Step 2: Save the grid.html file.

Step 3: In the CSS document, add the unique item classes and set the background color to various shades of grey.

```
.grid1-item1 {
    background: #f1f1f2;
}
.grid1-item2 {
    background: #e6e7e8;
}
.grid1-item3 {
    background: #d0d2d3;
}
.grid1-item4 {
	background: #a6a8ab;
}
.grid1-item5 {
	background: #929497;
}
.grid1-item6 {
	background: #808184;
}
```

Step 4: Save the grid.css file.

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

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

o) Define the .grid1-container as a grid using the display property

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

display: grid;

Defining the display property as a grid tells the browser to treat the dive container as a grid.

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

```
.grid1-container {
	background: #58595b;
	padding: 10px;
	height: 400px;
	display: grid;
}
```

Step 4: Save the grid.css file.

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

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

p) Define the gap between the columns and rows using gap

Key takeaway*: Use the gap property to define the amount of space between grid items.*

column-gap: value; *– column spacing*

column-gap: value; *– row spacing*

gap: columnvalue rowvalue; *– shorthand for both with different values*

gap: value; *– shorthand for both if they have the same value*

The gird-gap property defines the of space between grid items both horizontally and vertically.

Step 1: In the CSS document, add the gap property and give it a value of gap: 10px;

```
.grid1-container {
    background: #58595b;
    padding: 10px;
    height: 400px;
    display: grid;
    gap: 10px;
}
```

Step 2: Save the grid.css file.

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

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

q) Define the number of columns in the grid using grid-template-columns

Key takeaway*: Use grid-template-columns to declare the number of columns and the width of each column.*

grid-template-columns: auto auto auto; *– creates evenly spaced columns*

The grid-template-columns property specifies the number of columns in the grid along with the width of each container. To let items stretch to fill the space use auto as the value.

Step 1: In the CSS document, add the grid-template-columns and set it to have 3 columns of equal widths

```
.grid1-container {
    background: #58595b;
    padding: 10px;
    height: 400px;
    display: grid;
    gap: 10px;
    grid-template-columns: auto auto auto;
}
```

Step 2: Save the grid.css file.

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

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

r) Define the width of columns in the grid using grid-template-columns

Key takeaway*: Use grid-template-columns to declare the number of columns and the width of each column.*

grid-template-columns: 100px 200px 200px; *– creates or defines values for the columns.*

The grid-template-columns property specifies the number of columns in the grid along with the width of each container. To let items stretch to fill the space use auto as the value.

Step 1: In the CSS document, alter the third column to be a specific width

```
.grid1-container {
    background: #58595b;
    padding: 10px;
    height: 400px;
    display: grid;
    gap: 10px;
    grid-template-columns: auto auto 300px;
}
```

Step 2: Save the grid.css file.

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

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

s) Define the height of rows in the grid using grid-template-rows

Key takeaway*: Use grid-template-rows to declare the height of each row.*

grid-template-rows: 100px 200px; *– creates or defines height values for the rows.*

The grid-template-columns property declares the height of each item.

Step 1: In the CSS document, add grid-template-rows and set the property to be 100px for the first 2 rows and let the browser calculate the remaining height

```
.grid1-container {
    background: #58595b;
    padding: 10px;
    height: 400px;
    display: grid;
    gap: 10px;
    grid-template-columns: auto auto 300px;
    grid-template-rows: 100px 100px auto;
}
```

Step 2: Save the grid.css file.

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

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

t) Define how much space each column takes use the grid-column-start and grid-column-end in the item classes

***Key takeaway**: Use column lines to define where a column starts and ends with the grid-column-start, grid-column-end properties.*

***grid-column-start: 1;** – start at the first column line*

***grid-column-end: 3;** – end at the third column line*

grid-column-start, grid-column-end properties define where a grid item starts and ends using column lines.

Step 1: In the CSS document, define the first grid item to take the space of 2 columns

```
.grid1-item1 {
    background: #f1f1f2;
    grid-column-start: 1;
    grid-column-end: 3;
}
```

Step 2: Save the grid.css file.

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

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

u) Define how much space each row takes use the grid-row-start and grid-row-end in the item classes

***Key takeaway**: Use row lines to define where a row starts and ends with the grid-row-start, grid-row-end properties.*

***grid-row-start: 1;** – start at the first column line*

***grid-row-end: 3;** – end at the third column line*

grid-row-start, grid-row-end properties define where a grid item starts and ends using row lines.

Step 1: In the CSS document, define the first grid item to take the space of 2 rows

```
.grid1-item1 {
    background: #f1f1f2;
    grid-column-start: 1;
    grid-column-end: 3;
    grid-row-start: 1;
    grid-row-end: 3;
}
```

Step 2: Save the grid.css file.

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

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

v) Learn grid-column – shorthand for column start and end

***Key takeaway**: Use grid-column as shorthand for the grid-column-start and grid-column-end properties.*

grid-column: startvalue / endvalue;

grid-column is the shorthand for the grid-column-start and grid-column-end properties.

w) Learn grid-row – shorthand for row start and end

Key takeaway*: Use grid-row as shorthand for the grid-row-start and grid-row-end properties.*

grid-row: startvalue / endvalue;

grid-row is the shorthand for the grid-row-start and grid-row-end properties.

x) Learn grid-area – shorthand for row and column start and end

Key takeaway*: Use grid-area as shorthand for both the grid-row-start and grid-column-start the grid-row-end and grid-column-end properties.*

grid-area: rowstartvalue /columnstartvalue/rowendvalue/ columnendvalue;

Grid-area is the shorthand for both the grid-row-start and grid-column-start the grid-row-end and grid-column-end properties.

y) Add the second grid section

Step 1: In the HTML document, add the h2 tag

```
<h2> Example 2: grid-column/row/area </h2>
```

Step 2: Add a div container

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

Step 3: Add 6 divs inside the previous div container, name and number the content within the divs

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

Step 4: Save the grid.css file.

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

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

z) Create a .grid2-container class and add a background-color and padding

Step 1: In the HTML document, add class to the main div, name it grid1-container.

```
<div class="grid2-container">
</div>
```

Step 2: Save the grid.html file.

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

```
.grid2-container {
}
```

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

```
.grid2-container {
    background: #58595b; }
```

Step 5: Set the padding property to 10px

```
.grid2-container {
    background: #58595b;
    padding: 10px; }
```

Step 6: Save the grid.css file.

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

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

aa) Add the .grid-item class to the new 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 grid-item to the divs inside the container div.

```
<div class="grid2-container">
    <div class="grid-item">Grid item 1</div>
    <div class="grid-item">Grid item 2</div>
    <div class="grid-item">Grid item 3</div>
    <div class="grid-item">Grid item 4</div>
    <div class="grid-item">Grid item 5</div>
    <div class="grid-item">Grid item 6</div>
</div>
```

Step 2: Save the grid.css file.

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

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

ab) Create a unique .grid-item# class for each div and define a different background color for each div

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

Add a class for each div item. Add different background color so the grid styles are easy to see.

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

```
<div class="grid2-container">
    <div class="grid-item grid2-item1">Grid item 1</div>
    <div class="grid-item grid2-item2">Grid item 2</div>
    <div class="grid-item grid2-item3">Grid item 3</div>
    <div class="grid-item grid2-item4">Grid item 4</div>
    <div class="grid-item grid2-item5">Grid item 5</div>
    <div class="grid-item grid2-item6">Grid item 6</div>
</div>
```

Step 2: Save the grid.html file.

Step 3: In the CSS document, add the unique item classes and set the background color to various shades of grey.

```
.grid2-item1 {
    background: #f1f1f2;
}
.grid2-item2 {
    background: #e6e7e8;
}
.grid2-item3 {
    background: #d0d2d3;
}
.grid2-item4 {
    background: #a6a8ab;
}
.grid2-item5 {
    background: #929497;
}
.grid2-item6 {
    background: #808184;
}
```

Step 4: Save the grid.css file.

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

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

ac) Define the .grid2-container as a grid using the display property

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

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

```
.grid2-container {
    background:#58595b;
    padding: 10px;
    display: grid;
}
```

Step 2: Save the grid.css file.

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

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

ad) Define the gap between the columns and rows using gap

***Key takeaway**: Use the gap property to define the amount of space between grid items.*

***column-gap: value;** – column spacing*

***column-gap: value;** – row spacing*

***gap: columnvalue rowvalue;** – shorthand for both with different values*

***gap: value;** – shorthand for both if they have the same value*

The gird-gap property defines the of space between grid items both horizontally and vertically.

Step 1: In the CSS document, add the gap property and give it a value of gap: 10px;

```
.grid2-container {
    background: #58595b;
    padding: 10px;
    display: grid;
    gap: 10px;
}
```

Step 2: Save the grid.css file.

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

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

ae) Define the number of columns in the grid using grid-template-columns

***Key takeaway**: Use grid-template-columns to declare the number of columns and the width of each column.*

Step 1: In the CSS document, add the grid-template-columns and set it to have 3 columns of equal widths

```
.grid2-container {
    background: #58595b;
```

```
        padding: 10px;
        height: 400px;
        display: grid;
        gap: 10px;
        grid-template-columns: auto auto auto;
    }
```

Step 2: Save the grid.css file.

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

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

af) Define how much space each column takes using shorthand for column start and end – grid-column

Key takeaway*: Use grid-column as shorthand for the grid-column-start and grid-column-end properties.*

Grid-column: startvalue / endvalue;

Grid-column is the shorthand for the grid-column-start and grid-column-end properties.

Step 1: In the CSS document, define the first grid item to take the space of all 3 columns

```
    .grid2-item1 {
        background:#f1f1f2;
        grid-column: 1 / 4;
    }
```

Step 2: Define the second grid item to take the space of 2 columns

```
    .grid2-item2 {
        background:#e6e7e8;
        grid-column: 1 / 3;
    }
```

Step 3: Define the third grid item to take the space of 1 column, starting on the third grid line

```
    .grid2-item3 {
        background:#d0d2d3;
        grid-column: 3 / 3;
    }
```

Step 4: Save the grid.css file.

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

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

ag) Define how much space each row takes using shorthand for row start and end – grid-row

Key takeaway*: Use grid-row as shorthand for the grid-row-start and grid-row-end properties.*

Grid-row: startvalue / endvalue;

Grid-row is the shorthand for the grid-row-start and grid-row-end properties.

Step 1: In the CSS document, define the third grid item to take the space of 2 rows, starting on the second row line

```
.grid2-item3 {
    background:#d0d2d3;
    grid-column: 3 / 3;
    grid-row: 2 / 5;
}
```

Step 2: Save the grid.css file.

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

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

ah) Define how much space each row and column takes using shorthand for row and column start and end – grid-area: 4/1/4/3;

Key takeaway*: Use grid-area as shorthand for both the grid-row-start and grid-column-start the grid-row-end and grid-column-end properties.*

grid-area: rowstartvalue /columnstartvalue/rowendvalue/ columnendvalue;

Grid-area is the shorthand for both the grid-row-start and grid-column-start the grid-row-end and grid-column-end properties.

Step 1: In the CSS document, define the sixth grid item to take the space of 2 rows, starting on the 4th row line

```
.grid2-item6 {
    background:#808184;
    grid-area: 4/1/4/3;
}
```

Step 2: Save the grid.css file.

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

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

ai) Learn how to define a grid using grid-template-areas

Key takeaway*: Use the grid-template-areas property to easily see the pattern of your grid in your code.*

grid-template-areas:

```
'name1 name1 name1'
'name2 name3 name3'
'name2 name3 name3';
```

Defining the grid area using names can make it easier to easily see the pattern of your grid by laying out the pattern of the grid using named values.

aj) Learn how to name an item for use in grid-template-areas

Key takeaway*: Use the grid-area property to define an items name for use with laying out a grid using the grid-template-areas property.*

grid-area: name;

The grid-area defines what you want your grid item to be named. Using these names, you can create a pattern for your div using the grid-template-areas property.

ak) Add the third grid section

Step 1: In the HTML document, add the h2 tag

```
<h2> Example 3: Naming Grid Items</h2>
```

Step 2: Add a div container

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

Step 3: Add 6 divs inside the previous div container, name and number the content within the divs

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

Step 4: Save the grid.html file.

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

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

al) Create a .grid3-container class and add a background-color and padding

Step 1: In the HTML document, add class to the main div, name it grid1-container.

```
<div class="grid3-container">
</div>
```

Step 2: Save the grid.html file.

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

```
.grid3-container {
        }
```

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

```
        .grid3-container {
            background: #58595b;
        }
```

Step 5: Set the padding property to 10px

```
        .grid3-container {
            background: #58595b;
            padding: 10px;
        }
```

Step 6: Save the grid.css file.

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

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

am) Add the .grid-item class to the new div items

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

```
        <div class="grid3-container">
            <div class="grid-item">Grid item 1</div>
            <div class="grid-item">Grid item 2</div>
            <div class="grid-item">Grid item 3</div>
            <div class="grid-item">Grid item 4</div>
            <div class="grid-item">Grid item 5</div>
            <div class="grid-item">Grid item 6</div>
        </div>
```

Step 2: Save the grid.css file.

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

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

an) Create a unique .grid-item# class for each div and define a different background color for each div

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

```
        <div class="grid3-container">
            <div class="grid-item grid3-item1">Grid item 1</div>
            <div class="grid-item grid3-item2">Grid item 2</div>
            <div class="grid-item grid3-item3">Grid item 3</div>
            <div class="grid-item grid3-item4">Grid item 4</div>
            <div class="grid-item grid3-item5">Grid item 5</div>
            <div class="grid-item grid3-item6">Grid item 6</div>
        </div>
```

Step 2: Save the grid.html file.

Step 3: In the CSS document, add the unique item classes and set the background color to various shades of grey.

```
    .grid3-item1 {
        background: #f1f1f2;
    }
    .grid3-item2 {
        background: #e6e7e8;
    }
    .grid3-item3 {
        background: #d0d2d3;
    }
    .grid3-item4 {
        background: #a6a8ab;
    }
    .grid3-item5 {
        background: #929497;
    }
    .grid3-item6 {
        background: #808184;
    }
```

Step 4: Save the grid.css file.

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

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

ao) Define the .grid3-container as a grid using the display property

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

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

```
    .grid3-container {
        background:#58595b;
        padding: 10px;
        display: grid;
    }
```

Step 2: Save the grid.css file.

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

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

ap) Define the gap between the columns and rows using gap

***Key takeaway**: Use the gap property to define the amount of space between grid items.*

Step 1: In the CSS document, add the gap property and give it a value of gap: 10px;

```
.grid3-container {
    background: #58595b;
    padding: 10px;
    display: grid;
    gap: 10px;
}
```

Step 2: Save the grid.css file.

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

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

ar) Define the grid-area names for each div item

***Key takeaway**: Use the grid-area property to define an items name for use with laying out a grid using the grid-template-areas property.*

Step 1: In the CSS document, define the grid area name for the item1

```
.grid3-item1 {
    background:#f1f1f2;
    grid-area: header;
}
```

Step 2: Define the grid area name for the item2

```
.grid3-item2 {
    background:#e6e7e8;
    grid-area: sidemenu;
}
```

Step 3: Define the grid area name for the item3

```
.grid3-item3 {
    background:#d0d2d3;
    grid-area: column1;
}
```

Step 4: Define the grid area name for the item4

```
.grid3-item4 {
        background:#a6a8ab;
        grid-area: column2;
}
```

Step 5: Define the grid area name for the item5

```
.grid3-item5 {
        background:#929497;
        grid-area: column3;
}
```

Step 6: Define the grid area name for the item6

```
.grid3-item6 {
        background:#808184;
        grid-area: column4;
}
```

Step 7: Save the grid.css file.

Step 8: Preview the grid.html page in the live server.

Step 9: Switch back to HTML document to proceed to next segment.

as) Define the grid layout using the grid-template-areas property

***Key takeaway**: Use the grid-template-areas property to easily see the pattern of your grid in your code.*

grid-template-areas:

'name1 name1 name1'

'name2 name3 name3'

'name2 name3 name3';

The grid rows are defined using '', columns are separates by spaces, grid area names are used to identify the content for each grid cell.

Defining the grid area using names can make it easier to easily see the pattern of your grid by laying out the pattern of the grid using named values.

Step 1: In the CSS document, add the grid-template-areas property

```
.grid3-container {
        background:#58595b;
        padding: 10px;
        display: grid;
        grid-template-areas:
                ''
                ''
```

```
                    '';
        gap: 10px;
}
```

Step 2: Define the first row of a 3-column grid using the area names

```
.grid3-container {
        background:#58595b;
        padding: 10px;
        display: grid;
        grid-template-areas:
                'header header header'
                ''
                '';
        gap: 10px;
}
```

Step 3: Define the second row using the area names

```
.grid3-container {
        background:#58595b;
        padding: 10px;
        display: grid;
        grid-template-areas:
                'header header header'
                'sidemenu column1 column2'
                '';
        gap: 10px;
}
```

Step 4: Define the third row using the area names

```
.grid3-container {
        background:#58595b;
        padding: 10px;
        display: grid;
        grid-template-areas:
                'header header header'
                'sidemenu column1 column2'
                'sidemenu column3 column4';
        gap: 10px;
}
```

Step 5: Save the grid.css file.

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

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

at) Explore horizontal alignment properties using the justify-content property

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

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 gap space*

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

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

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

au) Explore vertical alignment properties using the align-content property

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

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 gap space*

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

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

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