Terms

Responsive Design: When you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen or device.

Browser Support: Not all browsers will support the latest CSS code. On browsers that do not fully support the code you can add a prefix to the code to allow the code to function. Use the following for each browser:

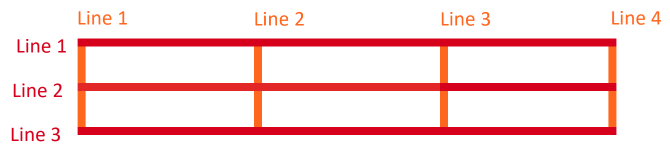
- Explorer prefix: -ms-
- Firefox prefix: -moz-
- Chrome prefix: -webkit-
- Safari prefix: -webkit-
- Opera prefix: -webkit-

Check www.w3schools.com/cssref/css3_browsersupport.asp to see if the target browser supports your CSS, or if you need to add the prefix.

Media Query: Media queries ask the browser if something is true and specifies a style based on the response.

Grid: Grid is an alternative layout to positioning elements manually with position, overflow or float. The grid layout approaches columns and rows holistically, allowing an item's size to be specified in relation to the rest of the items in the container.

- **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.



- **Browser support**: *Fully supported* by Chrome 57.0, Explorer 16.0, Firefox 52.0, Safari 10.0 and Opera 44.0

Flexbox: Flexbox is another alternative to positioning elements manually with position, overflow or float. The flex layout specifies how to handle the either the row or the column not both. It defines the main axis (either row or column) and the cross axis (either row or column).

- **Main axis** – the axis that the items are distributed along, specified by the flex-direction property.
- **Cross axis** – perpendicular to the main axis
- **Cross size** – the width or height of a flex item depending on what the cross axis is. If the main axis is row, the cross axis is column so the cross size would be specified as height.
- **Browser support**: *Fully supported* by Chrome 29.0, Explorer 11.0, Firefox 28.0, Safari 9.0 and Opera 17.0 *Partially supported* by Chrome 21.0 (-webkit-), Explorer 10.0 (-ms-), Firefox 18.0(-moz-), Safari 9.0 (-webkit-)

Structure

Media Query Structure

```
@media only screen and (min-width: 600px) {
    p {
        font-size: 14px;
    }
}
```

Media query tells the browser to apply the styles only if the query in brackets is true.

Nested styles

Only add selectors and properties that have to change.

CSS Grid Properties discussed in class

Below is a brief synopsis of the grid properties discussed in class. For more information such as additional property values visit www.w3schools.com/css/css_grid.asp. For a complete list of current **browser support** visit www.w3schools.com/cssref/css3_browsersupport.asp

Description	Example

display: grid;
Defining the display property as a grid tells the browser to treat the div container as a grid.
	`.container {`
	`  display: grid;`
	`}`

grid-gap property. The gird-gap property defines the of space between grid items both horizontally and vertically.
• **grid-column-gap: value;** – column spacing
• **grid-row-gap: value;** – row spacing
• **grid-gap: columnvalue rowvalue;** – shorthand for both with different values
• **grid-gap: value;** – shorthand for both if they have the same value
	`.grid1-container {`
	`  grid-gap: 10px;`
	`}`

grid-template-columns property. 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.
• **grid-template-columns: auto auto auto;** – creates evenly spaced columns
• **grid-template-columns: 100px 200px 200px;** – creates defines values for the columns
	`.grid1-container {`
	`  grid-template-columns: auto auto auto;`
	`}`

grid-template-rows property. The grid-template-columns property declares the height of each item.
• **grid-template-rows: 100px 200px;** – creates defines height values for the rows
	`.grid1-container {`
	`  grid-template-rows: 100px 100px auto;`
	`}`

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

- **grid-column-start: 1;** – start at the first column line
- **grid-column-end: 3;** – end at the third column line

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

```
.grid1-item1 {
  grid-row-start: 1;
  grid-row-end: 3;
}
```

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

- **grid-column: startvalue / endvalue;**

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

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

- **grid-row: startvalue / endvalue;**

```
.grid2-item1 {
  grid-row: 2 / 5;
}
```

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

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

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

grid-template-areas: property. 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.
grid-template-areas:
'name1 name1 name1'
'name2 name3 name3'
'name2 name3 name3';

```
.grid3-container {
  grid-template-areas:
       'header header header header header'
       'sidemenu column1 column1 column2 column2'
       'sidemenu column3 column3 column4 column4';
}
```

grid-area property. 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.
grid-area: name;

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

justify-content property: The justify-content property defines 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 centred 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

```
.grid-container {
  justify-content: space-evenly;
}
```

align-content property. The align-content property defines 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 centred 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

```
.grid-container {
  align-content: center;
}
```

CSS Flexbox Properties discussed in class

Below is a brief synopsis of the flexbox properties discussed in class. For more information such as additional property values visit www.w3schools.com/css/css3_flexbox.asp For a complete list of current **browser support** visit www.w3schools.com/cssref/css3_browsersupport.asp

Description | **Example**

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.

```
.container {
  display: flex;
}
```

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

- **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

```
.container {
  display: flex;
  flex-direction: row;
}
```

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

- **flex-wrap: wrap;**
- **flex-wrap: nowrap;**
- **flex-wrap: wrap-reverse;**

```
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}
```

flex-flow property. The flex-flow property lets you declare both the direction and the wrap properties.
- **flex-flow: row wrap;**
- **flex-flow: row nowrap;**

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

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

flex-shrink property. 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.
- **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.

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

flex-basis property. The flex-basis property declares the size of the item.
- **flex-basis: 100px;**
- **flex-basis: 60%;**

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

flex property. 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%.

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

order property. 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.

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

justify-content property: The justify-content property defines 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 centred 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

```
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: center;
}
```

align-content property. The align-content property defines 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 centred 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

```
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: center;
  align-content: center;
}
```