Terms

Mobile First: To help load times for smaller screens you should design for the mobile device first and use a media query for the larger screens. The reason for this is the rendering rules for Cascading Style sheets. The browser will render the code from top to bottom. If mobile styles are listed first, they will be rendered before all others. If the desktop styles are rendered as a media query, they won't be rendered at all for mobile devices as they won't meet the queries parameters.

Viewport: The viewport is the area that is visible of a website when browsing on a device or desktop. Adding the viewport meta tag to the head of your HTML document will set the width of your webpage to be the width of the device. You can use the vw (viewport width: 1vw = 1% of the viewport width) or vh (viewport height: 1vw = 1% of the viewport height) as a responsive unit of measurement.

- Add the following to the HTML head <meta name="viewport" content="width=device-width, initial-scale=1.0">
- Use vw or vh as your unit of measurement

Structure

Mobile First Structure

```
.grid-header {
    background-image: url("../img/beach-165213_banner-phone.jpg");
    background-repeat: no-repeat;
    background-position: center;
    grid-area: header;
}
```

Use appropriately sized images for the target device. Choose the smallest sizing for your initial styling.

```
@media only screen and (min-width: 1200px) {
    .grid-header {
        background-image: url("../img/beach-165213_banner-desktop.jpg");
    }
}
```

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 Properties discussed in class

Below is a brief synopsis of the properties discussed in class. For more information such as additional property values visit www.w3schools.com/cssref/default.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-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;`
	`}`

background-repeat property. sets whether or not the image will tile inside the container or display as a single image.	`.grid-header {`
	`  background-image: url("../img/beach-165213_banner-`
	`       phone.jpg");`
	`  background-repeat: no-repeat;`
	`}`

background-position property. Sets how the image will be positioned in the container. Values include:
• left top
• left center
• left bottom
• right top
• right center
• right bottom
• center top
• center center
• center bottom
If you only specify one value, the other will be "center"
	`.grid-header {`
	`  background-image: url("../img/beach-165213_banner-`
	`       phone.jpg");`
	`  background-repeat: no-repeat;`
	`  background-position: center;`
	`}`

background-size: cover;
This property and value resizes the background image to match the dimensions of the container; may cause cropping or stretching.

```
.grid-header {
  background-image: url("../img/beach-165213_banner-
       phone.jpg");
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
}
```

linear-gradient property. Sets a container to have a linear gradient.

- **background-image: linear-gradient(red, yellow, blue);** adds a gradient to the container background
- **background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.6));** to bottom specifies the direction the gradient runs and using the rgba colour value lets you add transparency to the gradient.

```
.grid-header {
  background-image: linear-gradient(to bottom,
rgba(0, 0, 0,  0), rgba(0, 0, 0, 0.6)),
url("../img/beach-165213_banner-    phone.jpg");
}
```

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;
}
```

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;
}
```

Width property. To make a video or image responsive set the value of the width to be 100% and the height to be auto. Note if you do not want the video to get bigger than 100% use max-width: 100% instead.

```
video {
  width: 100%;
  height: auto;
}
```

Explore on your own

Below is a list of additional property values you may want to explore.

Description | **Example**

Color Values:

- → Hexadecimal colors: #ff0000;
- → RGB colors: rgb(red, green, blue)
- → RGBA colors: rgba(red, green, blue, alpha)
- → HSL colors: hsl(hue, saturation, lightness)
- → HSLA colors: hsla(hue, saturation, lightness, alpha)
- → Predefined/Cross-browser color names: red

To explore these color values, go to www.w3schools.com/cssref/css_colors_legal.asp.

```
p {
  color: #ff0000;
}
```

Units of Measurement - relative values:

- → em
- → rem
- → vw
- → vh
- → %

To explore these units of measurement, go to www.w3schools.com/cssref/css_units.asp

```
table {
  margin: 5vw 10vw 5vw 10vw;
}
```