Module 3-3

CSS Grid & Responsive Design

CSS Grids: Introduction

By defining a grid, we create a two dimensional layout composed of columns and rows allowing us to better organize our web page’s contents.

CSS Grids: Defining

To define a grid we must specify a display attribute with a value of grid:

```
.myGrid {

        display: grid;

}
```

In this example, the CSS code will specify using a selector by class, that an html element with a class name of myGrid be defined as a grid.

CSS Grids: Columns and Grid Areas

grid-template-columns: This property defines the number of columns (and their respective width).

grid-template-areas: Matches each area of the grid to a specific HTML element. It also defines the number of rows.

A blank or empty area can be designated with a period:

“footer footer .”

```
.container {
    display: grid;
    grid-template-columns: 200px 1fr 200px;
    grid-template-areas:
        "header header header"
        "menu-nav main upcoming-events"
        "footer footer footer"
    ;
    height: 100vh;
    grid-gap: 10px;
}
```

CSS Grids: Associating Grid Areas with HTML Elements

```
.container {
    display: grid;
    grid-template-columns: 200px 1fr 200px;
    grid-template-areas:
        "header header header"
        "menu-nav main upcoming-events"
        "footer footer footer"
    ;
    height: 100vh;
    grid-gap: 10px;
}
```

```
header {
    grid-area: header;
}
nav#menu-nav {
    grid-area: menu-nav;
}
main {
    grid-area: main;
}
aside {
    grid-area: upcoming-events;
}
footer {
    grid-area: footer;
}
```

An html element with the <header> tag will be associated with the header section of the grid.

An html element with the <aside> tag will be associated with the upcoming-events section of the grid.

Let’s create a grid layout

Responsive Design

- Responsive Design is the use of CSS to define different screen layouts (mobile, tablet, screen, etc.)

- We can test responsive design on Chrome, by using responsive mode:
  - F12 to access the developer tools
  - CTRL + SHIFT + m to enter responsive mode

Media Query (Stacking)

The media query is used to define the rules used to render your page:

```
@media (max-width: 768px) {
/* css content  */
}

@media (max-width: 500px) {
/* css content  */
}
```

The CSS rule in this block will kick in if your width is between 500 and 768 inclusive.

The CSS rules in this block will kick if your width is less than 500 px inclusive.

Media Query (Overlapping)

Sometimes two distinct media queries are applicable at the same time, consider the following scenario

```
@media (max-width: 768px) {
/* css content  */
}

@media (min-width: 500px) {
/* css content  */
}
```

If the width of the page is 600 px, CSS selectors from both blocks will apply since:

500 < 600 < 768

Let’s apply some media queries