CSS Grid Review

CSS Grid Basics

- **Definition**: CSS Grid is a two-dimensional layout system used to create complex layouts in web pages. Grids will have rows and columns with gaps between them. To define a grid layout, you would set the display property to grid.

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

- **The fr (Fractional) Unit**: This unit represents a fraction of the space within the grid container. You can use the fr unit to create flexible grids.
- **Creating Gaps Between Tracks**: There are three ways to create gaps between tracks in CSS grid. You can use the column-gap property to create gaps between columns. You can use the row-gap property to create gaps between rows. Or you can use the gap shorthand property to create gaps between both rows and columns.
- **grid-template-columns**: This is used to set lines names and sizing for the grid track columns.

```
.container {
  display: grid;
  width: 100%;
  grid-template-columns: 30px 1fr;
}
```

- **grid-template-rows**: This is used to set lines names and sizing for the grid track rows.
- **grid-auto-flow**: The determines how auto placed items fit in the grid.

```
.container {
  display: grid;
  width: 100%;
```

```
 grid-auto-flow: column;
}
```

- **grid-auto-columns**: This is used to set the size for columns created implicitly.

```
.container {
 display: grid;
 width: 100%;
 grid-auto-columns: auto;
}
```

- **place-items**: This is used to align items for both block and inline directions.
- **align-items**: This is used to set the alignment for the items in a grid container.
- **repeat() Function**: This function is used to repeat sections in the track listing. Instead of writing grid-template-columns: 1fr 1fr 1fr; you can use the repeat() function instead.

```
.container {
 display: grid;
 grid-template-columns: repeat(3, 1fr);
}
```

- **Explicit Grid**: You can specify the number of lines or tracks using the grid-template-columns or grid-template-rows properties.
- **Implicit Grid**: When items are placed outside of the grid, then rows and columns are automatically created for those outside elements.
- **minmax() Function**: This function is used to set the minimum and maximum sizes for a track.

```
.container {
 display: grid;
 grid-template-columns: repeat(4, 1fr);
 grid-auto-rows: minmax(150px, auto);
```

}

- **Line-based Placement**: All grids have lines. To specify where the item begins on a line, you can use the grid-column-start and grid-row-start properties. To specify where the item ends on the line, you can use the grid-column-end and grid-row-end properties. You can also choose to use the grid-column or grid-row shorthand properties instead.

Here is an example of using the grid-column property to make an element stretch across all columns.

```
.element {
  grid-column: 1 / -1;
}
```

- **grid-template-areas**: The property is used to provide a name for the items you are going to position on the grid.

```
<div class="container">
  <div class="header">Header</div>
  <div class="sidebar">Sidebar</div>
  <div class="main">Main Content</div>
  <div class="footer">Footer</div>
</div>


.container {
  display: grid;
  grid-template-columns: 200px 1fr;
  grid-template-rows: auto 1fr auto;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
```

```
  gap: 20px;
}

.header {
  grid-area: header;
  background-color: #4CAF50;
  padding: 10px;
  color: white;
}

.sidebar {
  grid-area: sidebar;
  background-color: #f4f4f4;
  padding: 10px;
}

.main {
  grid-area: main;
  background-color: #e0e0e0;
  padding: 10px;
}

.footer {
  grid-area: footer;
  background-color: #4CAF50;
  padding: 10px;
```

```
  color: white;
}
```

Debugging CSS

- **DevTools (Developer Tools)**: DevTools allow you to inspect and modify your CSS in real-time. The Styles pane shows all the CSS rules applied to the selected element, including inherited styles. You can toggle individual properties on and off, edit values, and even add new rules directly in the browser. This immediate feedback is incredibly useful for experimenting with different styles without modifying your source code.
- **CSS Validators**: Validators are used to check your CSS against the official specifications and reports any errors or warnings. A popular validator you can use is the W3C CSS Validator.
- **Debugging Responsive Designs**: The DevTools has an option to allow you to simulate how your site looks on various screen sizes and devices. This can help you identify breakpoint issues or styles that don't scale well across different viewport sizes.