Training Day 14 Report

12 July 2025

Responsive Design, Media Queries, and Layout in CSS

Responsiveness

Responsiveness means your website adjusts its layout and appearance based on different screen sizes and devices (phones, tablets, desktops).

Media Queries

Media queries allow you to apply CSS rules only when certain conditions are true — like screen width, height, orientation, etc.

Syntax:

```
@media (condition) {
  /* CSS Rules */
}
```
:

Condition	Description
max-width	Apply when screen is **less than or equal** to the given width
min-width	Apply when screen is **greater than or equal** to the given width

Examples:

```
/* Phones (small screens) */
@media (max-width: 600px) {
  body {
    background: pink;
  }
}
/* Tablets */
@media (min-width: 601px) and (max-width: 768px) {
  body {
    background: lightblue;
  }
}
```

```
/* Desktops */
@media (min-width: 769px) {
 body {
  background: lightgreen;
 }
}
```

Viewport

- The viewport is the visible area of the web page.
- Usually set using:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Mobile-First Approach

- Start styling for small devices first (like phones).
- Then use min-width media queries to adjust for larger devices.

CSS

/* Base styles for mobile */

```
body {

 font-size: 14px;

}
```

/* Tablet and up */

```
@media (min-width: 768px) {
 body {
  font-size: 16px;
 }
}
```

Layout Systems in CSS

1. Flexbox – One-dimensional (Row or Column):

```
.container {

 display: flex;

 flex-direction: row; /* or column */

}
```

Used when you need to arrange items in a single direction
-Easy alignment, spacing, and centering

2. CSS Grid – Two-dimensional (Rows + Columns):

```
.container {
  display: grid;
  grid-template-columns: 200px 300px 200px;
  grid-template-rows: 100px 100px;
}
```

- Best for complex layouts in both rows and columns
- More powerful than Flexbox for entire page layout

repeat() Function

Instead of writing 200px 200px 200px, use:

```
grid-template-columns: repeat(3, 200px);
```

fr Unit (Fraction)

- 1fr means 1 part of the remaining space.
- Example:

```
grid-template-columns: 1fr 2fr 1fr;
```

This means: left and right take 1 portion each, middle takes 2 portions.

Gaps Between Grid Items

```
gap: 20px; /* both row and column */

row-gap: 10px;

column-gap: 15px;
```

Auto-Placement in Grid

- Grid auto-places items if you don’t manually position them.
- Useful for dynamic content.

CSS Grid Areas

Let you name parts of the layout (like header, sidebar, footer) and assign elements to them.

Step 1: Define Grid Layout with Area Names

```
.container {
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar content content"
    "footer footer footer";
  grid-template-columns: 1fr 2fr 2fr;
  grid-template-rows: auto auto auto;
  gap: 10px;
}
```

Step 2: Assign Areas to Elements

```
.header {
  grid-area: header;
}
.sidebar {
  grid-area: sidebar;
}
.content {
  grid-area: content;
}
.footer {
  grid-area: footer;
}
```

HTML Example

html

Copy

Edit

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

Table

Topic	**Purpose**
Media Query	Responsive styles
Flexbox	One-direction layout
Grid Layout	Two-direction layout
`fr` Unit	Fractional space division
Grid Areas	Name layout sections
Mobile-First	Base style for small devices
`repeat()`	Repeat values in Grid easily