Chapter-03

Introduction to Design approaches

Designing approaches in web development refer to methodologies and techniques used to structure and style web content. Two prominent approaches are table-based design and table-less design.

1. Table-based Design:

In the early days of web development, tables were commonly used to create layouts. Developers would use HTML tables to structure the content on a web page, with rows and columns defining the layout.

Developers would create tables with rows and columns, using HTML tags like <table>, <tr>, and <td> to arrange content. Attributes such as width, height, and border were often used to control the layout and appearance.

Advantages:

Simple to understand and implement, especially for beginners.

Consistent layout across different browsers, as tables were well-supported.

Disadvantages:

Poor accessibility and SEO, as screen readers and search engines struggled to interpret table-based layouts.

Maintenance issues, as making changes to the layout required editing HTML directly, leading to code redundancy and difficulty in updating.

Limited flexibility and scalability, as tables were rigid and not well-suited for responsive design.

2. Table-less Design:

With the evolution of web standards and the introduction of CSS (Cascading Style Sheets), developers began to move away from table-based layouts. Table-less design involves using CSS for layout and positioning, separating content from presentation.

Developers use CSS (and sometimes JavaScript) to style HTML elements without relying on tables. Techniques like floats, flexbox, and grid layout are employed to create complex and flexible designs.

Advantages:

Improved accessibility and SEO, as semantic HTML is easier for screen readers and search engines to understand.

Separation of concerns, as content is separated from presentation, leading to cleaner and more maintainable code.

Greater flexibility and scalability, as CSS provides more powerful layout options and better support for responsive design.

Disadvantages:

Steeper learning curve, especially for those transitioning from table-based design.

Inconsistent rendering across different browsers, requiring additional effort for cross-browser compatibility.

Potential performance issues, particularly with complex layouts or older browsers that may not fully support modern CSS features.

Css vs css3

CSS (Cascading Style Sheets) and CSS3 are both stylesheets used to define the presentation of HTML documents, but there are key differences between them.

CSS (Cascading Style Sheets)

Introduction: CSS was introduced in 1996.

Function: It separates content from presentation, allowing HTML to focus on the content while CSS handles the design and layout.

Syntax and Basic Features:

Selectors: Basic selectors (element, class, ID).

Properties: Basic styling properties like color, font, background, margin, padding, etc.

Box Model: Margin, border, padding, and content.

Positioning: Static, relative, absolute, fixed.

Pseudo-classes and Pseudo-elements: :hover, :first-child, :before, :after.

CSS3

Introduction: CSS3 is the latest evolution of CSS, split into modules for easier development and implementation. It was introduced in the late 2000s.

Function: Builds on CSS by introducing new modules, selectors, and properties to enhance the design capabilities.

Modules: CSS3 is modular, which means it is divided into different sections (modules) such as Selectors, Box Model, Backgrounds and Borders, Text Effects, 2D/3D Transformations, Animations, Multiple Column Layout, User Interface, etc.

Advanced Selectors: New attribute selectors, structural pseudo-classes (e.g., :nth-child, :nth-of-type).

Borders and Backgrounds: Rounded corners (border-radius), multiple backgrounds, background size (background-size), gradient backgrounds.

Text Effects: Text shadows (text-shadow), word wrapping (word-wrap), and improved text decoration.

Animations and Transitions: Smooth transitions (transition), keyframe animations (@keyframes).

Transforms: 2D and 3D transforms (e.g., rotate, scale, translate).

Flexbox and Grid Layout: Modern layout models for more flexible and responsive designs.

Media Queries: Enables responsive design by applying styles based on the viewport size, orientation, resolution, and other media features.

New Color Models: RGBA, and opacity for better color and transparency control.

Css

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>CSS Example</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <h1>Welcome to CSS</h1>



    <p>This is a simple example of CSS.</p>
```

```
</body>
</html>

/* Basic CSS Styles */
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 20px;
}

h1 {
    color: #333;
    font-size: 24px;
    text-align: center;
}

p {
    color: #666;
    font-size: 16px;
}
```

CSS3

Html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>CSS3 Example</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <h1>Welcome to CSS3</h1>

    <p>This is an enhanced example using CSS3.</p>

    <div class="box">CSS3 Box</div>

</body>

</html>
```

```
/* Advanced CSS3 Styles */

body {
```

```css
    font-family: 'Open Sans', sans-serif;

    margin: 0;
    padding: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    text-align: center;
    color: #fff;
}

h1 {
    font-size: 3em;
    margin-bottom: 10px;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
}

p {
    font-size: 1.2em;
    margin-bottom: 20px;
    opacity: 0.9;
}

.box {
    width: 150px;
    height: 150px;
    background-color: #4caf50;
    border-radius: 10px;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 1.2em;
    color: #fff;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    transition: transform 0.3s, background-color 0.3s;
}

.box:hover {
    transform: scale(1.1);
    background-color: #81c784;
```

}

Description:

font-family: 'Open Sans', sans-serif;

Sets the font for the text within the body to "Open Sans", falling back to a generic sans-serif font if "Open Sans" isn't available.

background: linear-gradient(to right, #ff7e5f, #feb47b);

Sets a linear gradient background that transitions from #ff7e5f (a coral color) to #feb47b (a peach color) from left to right.

margin: 0;

Removes the default margin around the body.

padding: 20px;

Adds 20 pixels of padding inside the body.

display: flex;

Enables flexbox layout for the body.

flex-direction: column;

Arranges child elements in a column (vertical stack).

align-items: center;

Centers child elements horizontally.

justify-content: center;

Centers child elements vertically.

height: 100vh;

Sets the height of the body to 100% of the viewport height.

text-align: center;

Centers text horizontally.

color: #fff;

Sets the text color to white.

H1 (Header)

font-size: 3em;

Sets the font size to 3 times the base font size.

margin-bottom: 10px;

Adds 10 pixels of space below the header.

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);

Adds a shadow to the text, with a 2-pixel offset horizontally and vertically, and a 4-pixel blur radius, using a semi-transparent black color.

P (Paragraph)

font-size: 1.2em;

Sets the font size to 1.2 times the base font size.

margin-bottom: 20px;

Adds 20 pixels of space below the paragraph.

opacity: 0.9;

Sets the opacity of the text to 90%, making it slightly transparent.

Box (Class)

width: 150px;

Sets the width of the box to 150 pixels.

height: 150px;

Sets the height of the box to 150 pixels.

background-color: #4caf50;

Sets the background color of the box to a shade of green (#4caf50).

border-radius: 10px;

Rounds the corners of the box with a radius of 10 pixels.

display: flex;

Enables flexbox layout for the box.

align-items: center;

Centers content inside the box vertically.

justify-content: center;

Centers content inside the box horizontally.

font-size: 1.2em;

Sets the font size inside the box to 1.2 times the base font size.

color: #fff;

Sets the text color inside the box to white.

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

Adds a shadow below the box with a 4-pixel vertical offset and an 8-pixel blur radius, using a semi-transparent black color.

transition: transform 0.3s, background-color 0.3s;

Sets a transition effect for transform and background-color changes, with a duration of 0.3 seconds.

Box Hover State

transform: scale(1.1);

Scales the box to 110% of its original size when hovered.

background-color: #81c784;

Changes the background color of the box to a lighter shade of green (#81c784) when hovered.

CSS Properties

Layout Properties

1. **display:** Defines how an element is displayed on the page (e.g., block, inline, flex, grid).

- **display: block;** - Element is displayed as a block.
- **display: inline;** - Element is displayed inline with the surrounding text.
- **display: flex;** - Element is a flex container.
- **display: grid;** - Element is a grid container.

2. **position:** Specifies the positioning method used for an element (static, relative, absolute, fixed, sticky).

- **position: relative;** - Element is positioned relative to its normal position.
- **position: absolute;** - Element is positioned relative to the nearest positioned ancestor.

3. top, right, bottom, left: Offsets for positioned elements.

- top: 10px; - Positions the element 10px from the top.

4. float: Specifies whether an element should float to the left, right, or not at all.

- float: left; - Element floats to the left.

5. clear: Prevents elements from floating around a cleared element.

- clear: both; - Prevents elements from floating on either side.

6. z-index: Sets the stack order of elements (higher numbers are on top).

- z-index: 10; - Element has a stack order of 10.

Box Model Properties

1. width, height: Defines the dimensions of an element.

- width: 100px; - Sets the width to 100px.

2. margin: Sets the outer space around an element.

- margin: 10px; - 10px margin on all sides.

3. padding: Sets the inner space between an element's content and its border.

- padding: 10px; - 10px padding on all sides.

4. border: Sets the border around an element.

- border: 1px solid black; - 1px solid black border.

5. box-sizing: Alters the default CSS box model to include padding and border in the element's total width and height.

- box-sizing: border-box; - Includes padding and border in width and height.

Typography Properties

1. font-family: Specifies the font for text.

- font-family: Arial, sans-serif; - Sets the font to Arial or sans-serif.

2. font-size: Sets the size of the font.

- font-size: 16px; - Sets the font size to 16px.

3. font-weight: Defines the thickness of the text.

- font-weight: bold; - Makes the text bold.

4. color: Sets the color of the text.

- color: #333333; - Sets the text color to a dark gray.

5. line-height: Sets the height of a line of text.

- line-height: 1.5; - Sets the line height to 1.5 times the font size.

6. text-align: Specifies the horizontal alignment of text.

- text-align: center; - Centers the text.

7. text-decoration: Adds decoration to text (e.g., underline, overline, line-through).

- text-decoration: underline; - Underlines the text.

Background Properties

1. background-color: Sets the background color of an element.

background-color: #f0f0f0; - Light gray background color.

2. background-image: Sets an image as the background.

background-image: url('image.jpg'); - Uses image.jpg as the background.

3. background-repeat: Specifies if/how a background image will be repeated.

background-repeat: no-repeat; - No repeat of the background image.

4. background-size: Sets the size of the background image.

background-size: cover; - Background image covers the entire element.

Flexbox Properties

1. flex-direction: Defines the direction of the flex items.

- flex-direction: row; - Flex items are in a row.

2. justify-content: Aligns flex items along the main axis.

- justify-content: center; - Centers flex items.

3. align-items: Aligns flex items along the cross axis.

- align-items: center; - Centers flex items vertically.

Grid Properties

1. grid-template-columns, grid-template-rows: Defines the columns and rows of the grid.

- grid-template-columns: 1fr 2fr; - Creates columns with 1 part and 2 parts width.
- grid-gap: Sets the gap between grid items.
- grid-gap: 10px; - 10px gap between grid items.