

CSS (Cascading Style Sheets):

- --> Basic Concepts
- Syntax and Selectors
- Colors
- Text
- Fonts
- Links
- Lists
- Tables

- --> Box Model
- Margin
- Padding
- Border
- Outline

- --> Backgrounds and Borders
- Background Color
- Background Image
- Background Repeat
- Background Position
- Background Attachment
- Border Radius
- Box Shadow

- --> Text and Fonts
- Text Color
- Text Alignment
- Text Decoration
- Text Transform
- Text Indent
- Line Height
- Letter Spacing
- Font Style
- Font Size

- Font Family

- --> **Selectors**

- Universal Selector
- Type Selector
- Class Selector
- ID Selector
- Attribute Selector
- Pseudo-classes
- Pseudo-elements
- Child Selector
- Adjacent Sibling Selector
- --> Layout
- Display
- Positioning (Static, Relative, Absolute, Fixed, Sticky)
- Z-index
- Float
- Clear
- Flexbox
- Grid
- --> Responsive Design
- Media Queries
- Viewport
- Responsive Units (vw, vh, %, etc.)
- --> Animations and Transitions
- Transitions
- Animations
- Keyframes
- --> CSS Functions
- calc()
- var()

- url()
- rgba(), hsla()
- --> Miscellaneous
- CSS Variables (Custom Properties)
- @font-face
- @import
- Shorthand Properties
- --> Advanced Topics
- CSS Preprocessors (LESS, SASS)
- CSS Frameworks (Bootstrap, Tailwind)
- CSS-in-JS
- Custom Properties (CSS Variables)
- This list covers a wide range of topics within CSS, from the basics to advanced concepts. If you want details on any specific topic, let me know! 😊

1. Syntax and Selectors

Theory: CSS (Cascading Style Sheets) is used to style and layout web pages. It allows you to select HTML elements and apply styles to them.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Syntax and Selectors</title>
    <style>
        /* ID selector */
        #header {
            color: blue;
        }

        /* Class selector */
        .content {
            font-size: 18px;
        }

        /* Element selector */
        p {
            color: green;
        }
    </style>
</head>
<body>
    <h1 id="header">This is a header</h1>
    <p class="content">This is a paragraph.</p>
    <p>This is another paragraph.</p>
</body>
</html>
```

2. Colors

Theory: CSS allows you to set colors for various elements using keywords, HEX, RGB, or HSL values.

Html

Copy

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Colors</title>
    <style>
        h1 {
            color: #3498db; /* HEX color */
        }

        p {
            color: rgb(46, 204, 113); /* RGB color */
        }

        .highlight {
            background-color: hsl(120, 100%, 75%); /* HSL color */
        }
    </style>
</head>
<body>
    <h1>Color Example</h1>
    <p>This text is green.</p>
    <p class="highlight">This text has a green background.</p>
</body>
</html>
```

3. Text

Theory: CSS provides properties to style text, including color, alignment, decoration, and transformation.

Text properties in CSS allow you to control the appearance of text, including its color, alignment, decoration (like underline), transformation (like uppercase), and spacing between letters and lines.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Text</title>
    <style>
        p {
            color: red;
            text-align: center; /* Center align */
            text-decoration: underline; /* Underline text */
            text-transform: uppercase; /* Uppercase text */
        }
    </style>
</head>
<body>
    <p>This is some styled text.</p>
</body>
</html>
```

4. Fonts

Theory: You can specify the font family, size, weight, and style using CSS. Font properties allow you to specify the font family (typeface), size, weight (thickness), and style (like italic) of text. This helps in enhancing readability and aesthetic appeal of the text content on web pages.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fonts</title>
    <style>
        p {
            font-family: Arial, sans-serif; /* Font family */
            font-size: 20px; /* Font size */
            font-weight: bold; /* Font weight */
            font-style: italic; /* Font style */
        }
    </style>
</head>
<body>
    <p>This is a paragraph with custom font settings.</p>
</body>
</html>
```

5. Links

Theory: CSS allows you to style links and their various states (hover, visited, active).

Link styling in CSS enables you to define how hyperlinks should look in different states, such as normal, visited, hover, and active states. This improves navigation and user experience on a webpage.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Links</title>
    <style>
        a {
            color: blue; /* Link color */
            text-decoration: none; /* Remove underline */
        }

        a:hover {
            color: red; /* Link color on hover */
        }

        a:visited {
            color: purple; /* Visited link color */
        }
    </style>
</head>
<body>
    <a href="#">This is a link</a>
</body>
</html>
```

6. Lists

Theory: CSS can be used to style lists, including bullet points, numbering, and custom markers.

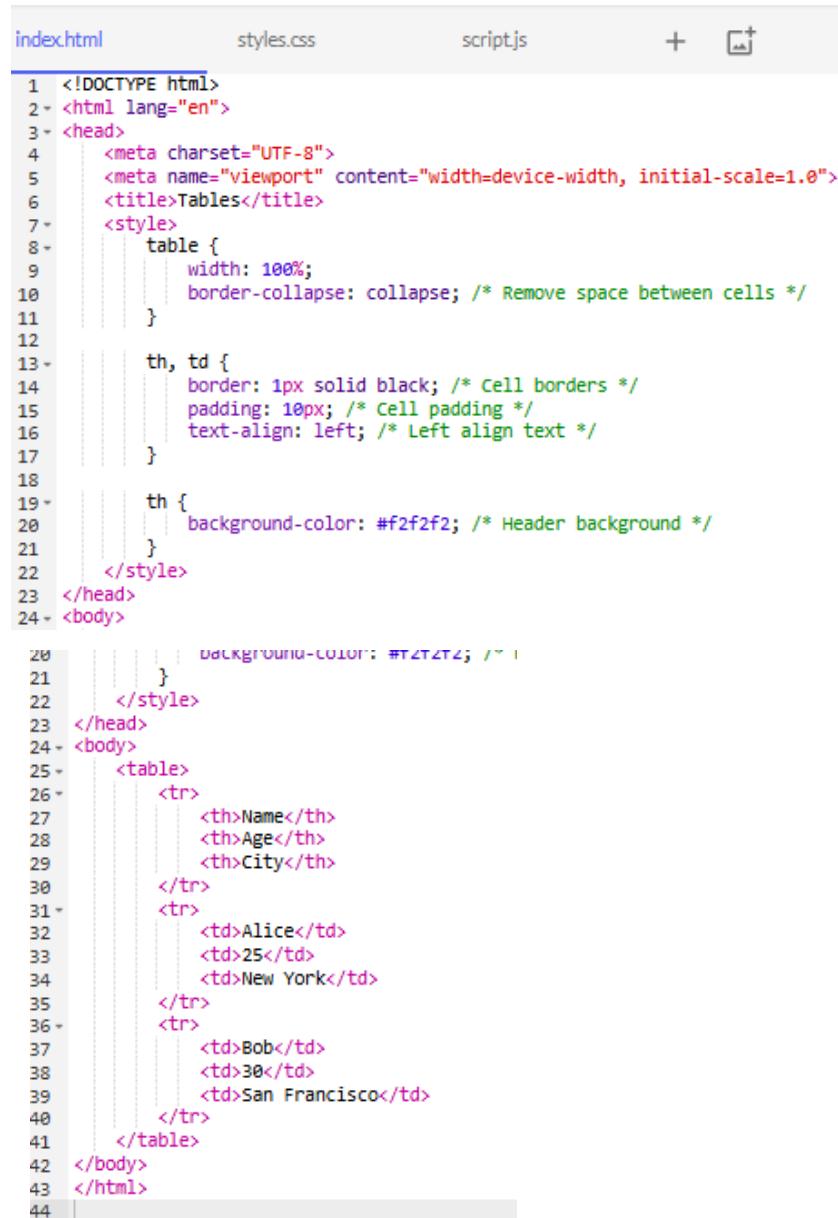
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lists</title>
    <style>
        ul {
            list-style-type: square; /* Bullet type for unordered list */
        }

        ol {
            list-style-type: upper-roman; /* Numbering type for ordered list */
        }
    </style>
</head>
<body>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>

    <ol>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
    </ol>
</body>
</html>
```

7. Tables

Theory: CSS can style tables, including borders, spacing, and alignment. Tables are used to display tabular data in a structured format. CSS allows you to style tables by setting properties for borders, spacing, alignment, and background colors to make the table data more readable and visually appealing.



```
index.html          styles.css          script.js      +  □  
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4      <meta charset="UTF-8">  
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">  
6      <title>Tables</title>  
7      <style>  
8          table {  
9              width: 100%;  
10             border-collapse: collapse; /* Remove space between cells */  
11         }  
12  
13         th, td {  
14             border: 1px solid black; /* Cell borders */  
15             padding: 10px; /* Cell padding */  
16             text-align: left; /* Left align text */  
17         }  
18  
19         th {  
20             background-color: #f2f2f2; /* Header background */  
21         }  
22     </style>  
23 </head>  
24 <body>  
25     <table>  
26         <tr>  
27             <th>Name</th>  
28             <th>Age</th>  
29             <th>City</th>  
30         </tr>  
31         <tr>  
32             <td>Alice</td>  
33             <td>25</td>  
34             <td>New York</td>  
35         </tr>  
36         <tr>  
37             <td>Bob</td>  
38             <td>30</td>  
39             <td>San Francisco</td>  
40         </tr>  
41     </table>  
42 </body>  
43 </html>
```

| Name | Age | City |
|-------|-----|---------------|
| Alice | 25 | New York |
| Bob | 30 | San Francisco |

- --> Box Model
- Margin
- Padding
- Border
- Outline

Box Model

Theory: The CSS box model is a fundamental concept that describes how elements are structured and displayed on a web page. Every element is essentially a rectangular box, and the box model consists of the following components:

1. **Content:** The actual content of the element, such as text or images.
2. **Padding:** Space between the content and the border.
3. **Border:** A border that surrounds the padding and content.
4. **Margin:** Space outside the border, separating the element from other elements.
5. **Outline:** A line drawn outside the border to make the element stand out, but not affecting the layout.

Example with Explanation:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Box Model Example</title>
    <style>
        .box {
            width: 200px; /* Width of the content area */
            padding: 20px; /* Space inside the border */
            border: 5px solid black; /* Border thickness and color */
            margin: 15px; /* Space outside the border */
            outline: 2px dashed red; /* Outline style and color */
        }
    </style>
</head>
<body>
    <div class="box">
        This is a box model example.
    </div>
</body>
</html>
```

Explanation:

1. **Content:** The actual text inside the `div` element.
2. **Padding:** `padding: 20px;` creates 20px space inside the border, around the content.
3. **Border:** `border: 5px solid black;` adds a solid black border with a thickness of 5px around the padding.
4. **Margin:** `margin: 15px;` adds 15px space outside the border, separating this element from others.
5. **Outline:** `outline: 2px dashed red;` adds a dashed red line outside the border, but it doesn't affect the layout or the position of the element.

Component Breakdown:

1. Margin

Theory: The margin is the outermost layer, creating space between the element's border and adjacent elements. It can be set for all sides (top, right, bottom, left) individually or together.

Example:

```
<div class="margin-example">
    This element has a margin.
</div>

<style>
    .margin-example {
        margin: 20px; /* 20px margin on all sides */
    }
</style>
```

Explanation: The `margin` property sets a uniform margin of 20px on all sides of the element.

2. Padding

Theory: Padding is the space between the content and the border of an element. It ensures the content doesn't touch the border directly. It can also be set for all sides individually or together.

Example:

```
<div class="padding-example">
    This element has padding.
```

```
</div>

<style>
    .padding-example {
        padding: 10px; /* 10px padding on all sides */
    }
</style>
```

Explanation: The `padding` property sets a uniform padding of 10px on all sides of the element.

3. Border

Theory: The border surrounds the padding and content. It can have various styles (solid, dashed, dotted), widths, and colors.

Example:

```
<div class="border-example">
    This element has a border.
</div>

<style>
    .border-example {
        border: 2px solid blue; /* 2px solid blue border */
    }
</style>
```

Explanation: The `border` property sets a 2px solid blue border around the element.

4. Outline

Theory: The outline is similar to a border but does not affect the element's size or layout. It is drawn outside the border and can help highlight elements.

Example:

```
<div class="outline-example">
    This element has an outline.
</div>

<style>
    .outline-example {
        outline: 3px dotted red; /* 3px dotted red outline */
    }
</style>
```

Explanation: The `outline` property sets a 3px dotted red outline around the element, which doesn't affect the element's layout or size.

- --> Backgrounds and Borders
- Background Color
- Background Image
- Background Repeat
- Background Position
- Background Attachment
- Border Radius
- Box Shadow

Backgrounds and Borders in CSS

CSS allows you to style the backgrounds and borders of HTML elements in various ways. Let's go through each of the listed properties with a brief theory and an example.

1. Background Color

Theory: The `background-color` property sets the background color of an element.

Example:

```

<!DOCTYPE html>
<html>
<head>
    <style>
        .example {
            background-color: lightblue;
        }
    </style>
</head>
<body>
    <div class="example">This is a div with a lightblue background color.</div>
</body>
</html>

```

The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab is active and contains the following code:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5      .example {
6          background-color: lightblue;
7      }
8  </style>
9 </head>
10 <body>
11 <div class="example">This is a div with a lightblue background color.</div>
12 </body>
13 </html>
14

```

To the right of the code editor, there is a preview window showing a single `<div>` element with a lightblue background and white text containing the text "This is a div with a lightblue background color."

2. Background Image

Theory: The `background-image` property sets an image as the background of an element.

Example:

The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab is active and contains the following code:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5      .example {
6          background-image: url('background.jpg');
7      }
8  </style>
9 </head>
10 <body>
11 <div class="example">This is a div with a background image.</div>
12 </body>
13 </html>
14

```

To the right of the code editor, there is a preview window showing a single `<div>` element with a background image and white text containing the text "This is a div with a background image."

The screenshot shows a browser developer tools window. On the left, the code editor displays the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
.example {
    background-image: url('https://th.bing.com/th/id/R.42a3f746e65bbacb555201c4890070c2?rik=C3KrlJfuV3KB0A&rlv=http%3a%2f%2fgetwallpapers.com%2fwallpaper%2ffull%2ff%2fk%2f8%2f86367.jpg&ehk=yetQ9yzSa2NTPb0oggs5p36e08gRax%2bqcU37mY4Mdc4%3d&risl=8&pid=ImgRaw&r=0');
}
</style>
</head>
<body>
<div class="example">This is a div with a background image.</div>
</body>
</html>
```

On the right, the preview pane shows a blue rectangular area with the text "This is a div with a background image." and a small red icon at the top.

3. Background Repeat

Theory: The `background-repeat` property defines if/how a background image will be repeated.

Example:

The screenshot shows a code snippet in a dark-themed code editor. It includes an "Html" button and a "Copy" button. The code example is as follows:

```
<!DOCTYPE html>
<html>
<head>
<style>
.example {
    background-image: url('background.jpg');
    background-repeat: no-repeat;
}
</style>
</head>
<body>
<div class="example">This is a div with a non-repeating background image.
</div>
</body>
</html>
```

4. Background Position

Theory: The `background-position` property sets the initial position of the background image.

```
hey.html > html > head > style > .example
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          .example {
6              background-image: url('background.jpg');
7              background-repeat: no-repeat;
8              background-position: center center;
9          }
10     </style>
11 </head>
12 <body>
13     <div class="example">This is a div with a centered background image.</div>
14 </body>
15 </html>
16
```

This

5. Background Attachment

Theory: The `background-attachment` property sets whether a background image is fixed or scrolls with the rest of the page.

```
hey.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          .example {
6              background-image: url('background.jpg');
7              background-attachment: fixed;
8          }
9      </style>
10 </head>
11 <body>
12     <div class="example" style="height: 200px;">
13         | This is a div with a fixed background image.
14     </div>
15     <div style="height: 800px;">
16         | Scroll down to see the effect.
17     </div>
18 </body>
19 </html>
20
```

← → ⏪ http://127.0.0.1:3000/hey.html

This is a div with a fixed background image.

Scroll down to see the effect.

Scrolling Effect :

The screenshot shows a code editor with several tabs: 'Code.html', 'hey.html', 'script.js', and 'styles.css'. The 'hey.html' tab contains the following HTML and CSS:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     .example {
6       background-image: url('cccc.jpeg');
7       background-attachment: fixed;
8     }
9   </style>
10 </head>
11 <body>
12   <div class="example" style="height: 200px;">
13     This is a div with a fixed background image.
14   </div>
15   <div style="height: 800px;">
16     Scroll down to see the effect.
17   </div>
18 </body>
19 </html>
20

```

The browser window shows a dark, cloudy sky over a mountain range. A tooltip says 'Scroll down to see the effect.'

6. Border Radius

Theory: The `border-radius` property defines the radius of the element's corners.

The screenshot shows a code editor with tabs: 'index.html', 'styles.css', and 'script.js'. The 'index.html' tab contains the following HTML:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     .example {
6       background-color: lightblue;
7       border: 20px solid black;
8       border-radius: 50px;
9       padding: 20px;
10    }
11   </style>
12 </head>
13 <body>
14   <div class="example">This is a div with rounded corners.</div>
15 </body>
16 </html>
17

```

The browser window shows a light blue rectangular box with rounded corners. A tooltip says 'This is a div with rounded corners.'

Box Shadow Theory

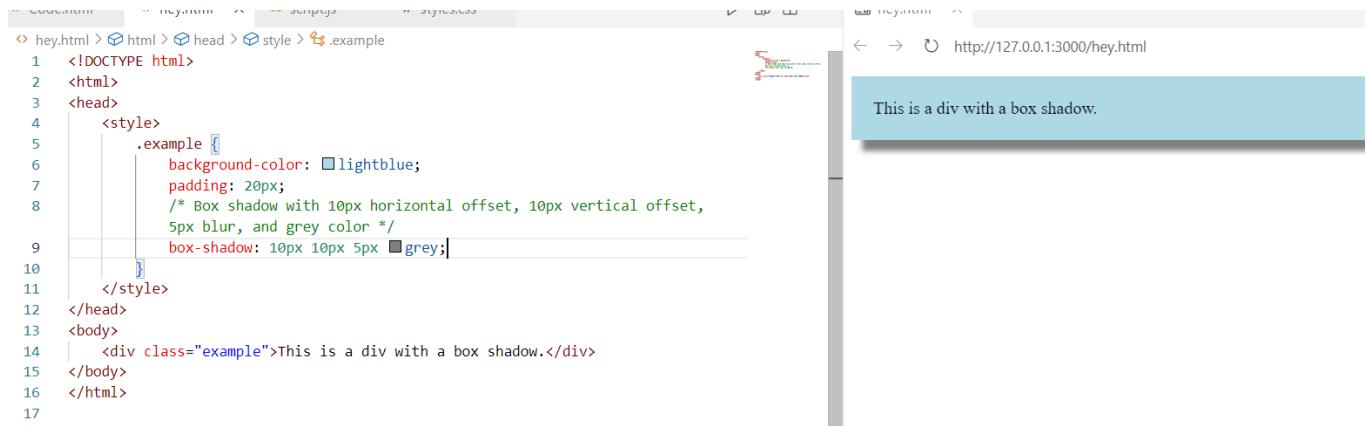
Theory: The `box-shadow` property adds shadow effects around an element's frame. You can specify the shadow's color, size, blur, and spread.

Syntax:

```
box-shadow: h-offset v-offset blur spread color;
```

- **h-offset (horizontal offset):** Moves the shadow horizontally. Positive values move the shadow to the right, and negative values move it to the left.
- **v-offset (vertical offset):** Moves the shadow vertically. Positive values move the shadow down, and negative values move it up.

- **blur:** Defines the blur radius of the shadow. Higher values create more blur.
- **spread:** Defines the spread radius. Positive values expand the shadow, and negative values shrink it.
- **color:** Specifies the color of the shadow.



```

hey.html > html > head > style > .example
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          .example {
6              background-color: lightblue;
7              padding: 20px;
8              /* Box shadow with 10px horizontal offset, 10px vertical offset,
9                 5px blur, and grey color */
10             box-shadow: 10px 10px 5px grey;
11         }
12     </style>
13 </head>
14 <body>
15     <div class="example">This is a div with a box shadow.</div>
16 </body>
17 </html>

```

- --> Text and Fonts
- Text Color
- Text Alignment
- Text Decoration
- Text Transform
- Text Indent
- Line Height
- Letter Spacing
- Font Style
- Font Size
- Font Family

CSS: Text and Fonts

Understanding how to style text and fonts in CSS is essential for creating visually appealing web pages. Below are key properties with brief explanations and examples for each.

1. Text Color

Theory: The `color` property sets the color of the text content of an element.

Example:

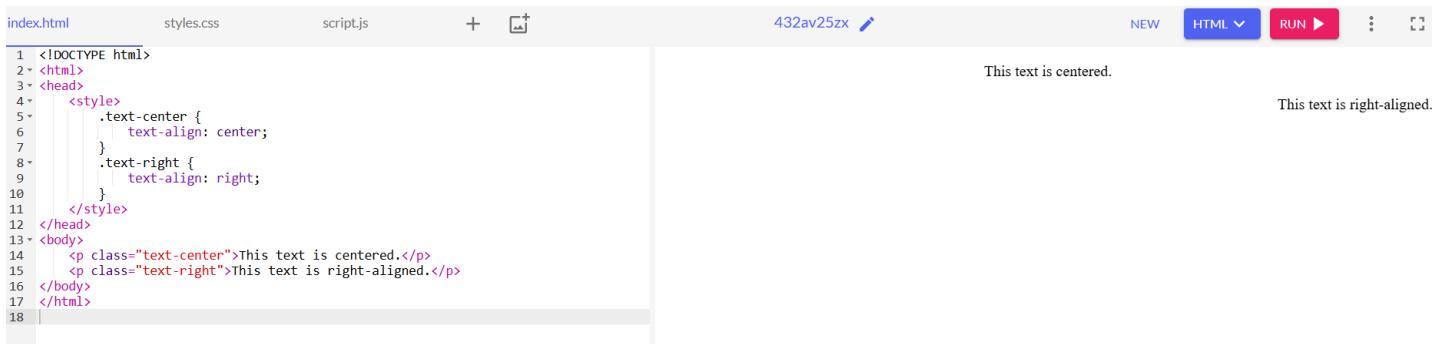


```
index.html          styles.css          script.js      +  +
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <style>
5      .text-color {
6        color: blue;
7      }
8    </style>
9  </head>
10 <body>
11   <p class="text-color">This text is blue.</p>
12 </body>
13 </html>
14 |
```

This text is blue.

2. Text Alignment

Theory: The `text-align` property sets the horizontal alignment of text within an element.



```
index.html          styles.css          script.js      +  +
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <style>
5      .text-center {
6        text-align: center;
7      }
8      .text-right {
9        text-align: right;
10     }
11    </style>
12  </head>
13  <body>
14   <p class="text-center">This text is centered.</p>
15   <p class="text-right">This text is right-aligned.</p>
16  </body>
17 </html>
18 |
```

This text is centered.

This text is right-aligned.

3. Text Decoration

Properties:

- `text-decoration`: Adds decorations to the text (e.g., `underline`, `overline`, `line-through`, `none`).

```

<!DOCTYPE html>
<html>
<head>
<style>
    .underline {
        text-decoration: underline;
    }
    .line-through {
        text-decoration: line-through;
    }
</style>
</head>
<body>
    <p class="underline">This text is underlined.</p>
    <p class="line-through">This text has a line through it.</p>
</body>
</html>

```

4. Text Transform

Properties:

- **text-transform:** Controls the capitalization of text. Possible values are **capitalize, uppercase, lowercase, none.**



The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab contains the following code:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <style>
5         .uppercase {
6             text-transform: uppercase;
7         }
8         .capitalize {
9             text-transform: capitalize;
10        }
11     </style>
12 </head>
13 <body>
14     <p class="uppercase">this text is uppercase.</p>
15     <p class="capitalize">this text is capitalized.</p>
16 </body>
17 </html>
18

```

The styles.css tab contains the following CSS rules:

```

.uppercase {
    text-transform: uppercase;
}
.capitalize {
    text-transform: capitalize;
}

```

The right side of the editor shows the rendered output of the HTML code. It displays two paragraphs: "THIS TEXT IS UPPERCASE." in all caps and "This Text Is Capitalized." where the first letter of each word is capitalized.

5. Text Indent

Properties:

- `text-indent`: Specifies the indentation of the first line of text in a block.

Example:

The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     .text-indent {
6       text-indent: 50px;
7     }
8   </style>
9 </head>
10 <body>
11   <p class="text-indent">This text is indented by 50px.</p>
12 </body>
13 </html>
```

The styles.css tab has no content. The script.js tab also has no content. On the right side of the editor, there is a status bar with the identifier "432av25zx" and a pen icon, followed by "HTML" and "RUN" buttons. A tooltip "This text is indented by 50px." is displayed near the indented text in the preview area.

6. Line Height

Properties:

- `line-height`: Sets the amount of space between lines of text.

The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     .line-height {
6       line-height: 1.5;
7     }
8   </style>
9 </head>
10 <body>
11   <p class="line-height">This text has a line height of 1.5.</p>
12 </body>
13 </html>
```

The styles.css tab has no content. The script.js tab also has no content. A tooltip "This text has a line height of 1.5." is displayed near the text in the preview area.

7. Letter Spacing

Properties:

- `letter-spacing`: Sets the space between characters in text.



A screenshot of a code editor showing a file named index.html. The code defines a CSS rule for letter-spacing and applies it to a paragraph. A callout box points to the paragraph text with the caption "This text has 2px letter spacing."

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5   .letter-spacing {
6     letter-spacing: 2px;
7   }
8 </style>
9 </head>
10 <body>
11   <p class="letter-spacing">This text has 2px letter spacing.</p>
12 </body>
13 </html>
14 |
```

8. Font Style

Properties:

- **font-style**: Sets the style of the font, such as **normal**, **italic**, or **oblique**.



A screenshot of a code editor showing a file named index.html. It contains CSS rules for italic and normal styles and applies them to paragraphs. Callout boxes point to each paragraph with their respective styles.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5   .italic {
6     font-style: italic;
7   }
8   .normal {
9     font-style: normal;
10  }
11 </style>
12 </head>
13 <body>
14   <p class="italic">This text is italic.</p>
15   <p class="normal">This text is normal.</p>
16 </body>
17 </html>
18 |
```

9. Font Size

Properties:

- **font-size**: Sets the size of the text.

The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab contains the following HTML and CSS code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     .font-small {
6       font-size: 12px;
7     }
8     .font-large {
9       font-size: 24px;
10    }
11  </style>
12 </head>
13 <body>
14   <p class="font-small">This text is small.</p>
15   <p class="font-large">This text is large.</p>
16 </body>
17 </html>
18
```

The styles.css tab contains the following CSS rule:

```
.font-small {
  font-size: 12px;
}

.font-large {
  font-size: 24px;
}
```

The script.js tab is empty.

The right panel displays two paragraphs of text: "This text is small." and "This text is large.", demonstrating the effect of the CSS rules defined in styles.css.

10. Font Family

Properties:

- `font-family`: Specifies the typeface to be used for text.

The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab contains the following HTML and CSS code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     .font-family {
6       font-family: 'Arial', sans-serif;
7     }
8   </style>
9 </head>
10 <body>
11   <p class="font-family">This text is in Arial font.</p>
12 </body>
13 </html>
14
```

The styles.css tab contains the following CSS rule:

```
.font-family {
  font-family: 'Arial', sans-serif;
}
```

The script.js tab is empty.

The right panel displays a single paragraph of text: "This text is in Arial font.", demonstrating the effect of the CSS rule defined in styles.css.

● --> Selectors

- Universal Selector
- Type Selector
- Class Selector
- ID Selector
- Attribute Selector
- Pseudo-classes
- Pseudo-elements
- Child Selector
- Adjacent Sibling Selector

1. Universal Selector

Theory: The universal selector (*) selects all elements on the page.

index.html styles.css script.js +

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     * {
6       color: blue;
7     }
8   </style>
9 </head>
10 <body>
11   <p>This text is blue.</p>
12   <div>This text is also blue.</div>
13   <div>This text is also blue.</div>
14 </body>
15 </html>
16
```

This text is blue.
This text is also blue.
This text is also blue.
srrtut

2. Type Selector

Theory: The type selector selects all elements of a given type.

The screenshot shows a code editor with three tabs: index.html, styles.css, and script.js. The styles.css tab is active, displaying the following CSS rule:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     p {
6       color: green;
7     }
8   </style>
9 </head>
10 <body>
11   <p>This text is green.</p>
12   <p>This text is also green.</p>
13   <div>This text is not green.</div>
14 </body>
15 </html>
16
```

To the right of the code, three paragraphs are displayed with different colors:

- "This text is green." (green text)
- "This text is also green." (green text)
- "This text is not green." (black text)

3. Class Selector

Theory: The class selector selects all elements with a specific class attribute. It uses a period (.) followed by the class name.

The screenshot shows a code editor with three tabs: index.html, styles.css, and script.js. The styles.css tab is active, displaying the following CSS rule:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     .highlight {
6       background-color: yellow;
7     }
8   </style>
9 </head>
10 <body>
11   <p class="highlight">This paragraph is highlighted.</p>
12   <p>This paragraph is not highlighted.</p>
13 </body>
14 </html>
15
```

To the right of the code, two paragraphs are shown:

- "This paragraph is highlighted." (yellow background)
- "This paragraph is not highlighted." (white background)

4. ID Selector

Theory: The ID selector selects a single element with a specific ID attribute. It uses a hash (#) followed by the ID name.

The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab is active, displaying the following HTML and CSS code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     #unique {
6       color: red;
7     }
8   </style>
9 </head>
10 <body>
11   <p id="unique">This text is red.</p>
12   <p>This text is not red.</p>
13 </body>
14 </html>
```

The CSS rule `#unique { color: red; }` targets elements with the ID "unique". The first paragraph (`<p id="unique">This text is red.</p>`) has its text color set to red, while the second paragraph (`<p>This text is not red.</p>`) does not.

5. Attribute Selector

Theory: The attribute selector selects elements with a specific attribute or attribute value.

The screenshot shows a code editor interface with three tabs: index.html, styles.css, and script.js. The index.html tab is active, displaying the following HTML and CSS code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     [type="text"] {
6       border: 1px solid black;
7     }
8   </style>
9 </head>
10 <body>
11   <input type="text" value="This has a border.">
12   <input type="button" value="This does not have a border.">
13 </body>
14 </html>
```

The CSS rule `[type="text"] { border: 1px solid black; }` targets input elements with the attribute `type="text"`. The first input field (`<input type="text" value="This has a border.">`) has a black border, while the second input field (`<input type="button" value="This does not have a border.">`) does not.

6. Pseudo-classes

Theory: Pseudo-classes select elements based on their state or position within the document.

```
index.html          styles.css          script.js      +   Hover over this link.  
1  <!DOCTYPE html>  
2  <html>  
3  <head>  
4  <style>  
5    a:hover {  
6      color: orange;  
7    }  
8  </style>  
9 </head>  
10 <body>  
11   <a href="#">Hover over this link.</a>  
12 </body>  
13 </html>
```

7. Pseudo-elements

Theory: Pseudo-elements select and style a part of an element.

```
index.html          styles.css          script.js      +   This is an example paragraph.  
1  <!DOCTYPE html>  
2  <html>  
3  <head>  
4  <style>  
5    p::first-letter {  
6      font-size: 24px;  
7      color: red;  
8    }  
9  </style>  
10 </head>  
11 <body>  
12   <p>This is an example paragraph.</p>  
13 </body>  
14 </html>
```

8. Child Selector

Theory: The child selector (>) selects elements that are direct children of a specified element.

```
index.html          styles.css          script.js      +   [+]  
1  <!DOCTYPE html>  
2  <html>  
3  <head>  
4  <style>  
5  div > p {  
6  |   color: blue;  
7  }  
8  </style>  
9  </head>  
10 <body>  
11 <div>  
12 <p>This text is blue.</p>  
13 <span>This text is not blue.</span>  
14 </div>  
15 <p>This text is not blue.</p>  
16 </body>  
17 </html>  
18
```

This text is blue.
This text is not blue.
This text is not blue.

9. Adjacent Sibling Selector

Theory: The adjacent sibling selector (+) selects an element that is immediately preceded by a specified element.

```
index.html          styles.css          script.js      +   [+]  432bbbdk2 [ ]  
1  <!DOCTYPE html>  
2  <html>  
3  <head>  
4  <style>  
5  h1 + p {  
6  |   color: purple;  
7  }  
8  </style>  
9  </head>  
10 <body>  
11 <h1>This is a heading.</h1>  
12 <p>This text is purple because it directly follows a heading.</p>  
13 <p>This text is not purple.</p>  
14 </body>  
15 </html>  
16
```

This is a heading.
This text is purple because it directly follows a heading.
This text is not purple.

- --> Layout
- Display
- Positioning (Static, Relative, Absolute, Fixed, Sticky)
- Z-index
- Float
- Clear
- Flexbox
- Grid



1. RELATIVE

15 16 17 18 19 20

1. **Relative**

2. **Absolute**

3. **Fixed**

4. **Sticky**

5. **Static**

6. **Flexbox**

7. **Grid**

8. **Position**

9. **Display**

10. **Z-index**

11. **Float**

12. **Clear**

13. **Overflow**

14. **Transform**

15. **Transition**

16. **Animation**

17. **Keyframes**

18. **Font**

19. **Color**

20. **Background**

21. **Border**

22. **Outline**

23. **Box-Shadow**

24. **Filter**

25. **Transform**

26. **Transition**

27. **Animation**

28. **Keyframes**

29. **Font**

30. **Color**

31. **Background**

32. **Border**

33. **Outline**

34. **Box-Shadow**

35. **Filter**

36. **Transform**

37. **Transition**

38. **Animation**

39. **Keyframes**

40. **Font**

41. **Color**

42. **Background**

43. **Border**

44. **Outline**

45. **Box-Shadow**

46. **Filter**

47. **Transform**

48. **Transition**

49. **Animation**

50. **Keyframes**

51. **Font**

52. **Color**

53. **Background**

54. **Border**

55. **Outline**

56. **Box-Shadow**

57. **Filter**

58. **Transform**

59. **Transition**

60. **Animation**

61. **Keyframes**

62. **Font**

63. **Color**

64. **Background**

65. **Border**

66. **Outline**

67. **Box-Shadow**

68. **Filter**

69. **Transform**

70. **Transition**

71. **Animation**

72. **Keyframes**

73. **Font**

74. **Color**

75. **Background**

76. **Border**

77. **Outline**

78. **Box-Shadow**

79. **Filter**

80. **Transform**

81. **Transition**

82. **Animation**

83. **Keyframes**

84. **Font**

85. **Color**

86. **Background**

87. **Border**

88. **Outline**

89. **Box-Shadow**

90. **Filter**

91. **Transform**

92. **Transition**

93. **Animation**

94. **Keyframes**

95. **Font**

96. **Color**

97. **Background**

98. **Border**

99. **Outline**

100. **Box-Shadow**

101. **Filter**

102. **Transform**

103. **Transition**

104. **Animation**

105. **Keyframes**

106. **Font**

107. **Color**

108. **Background**

109. **Border**

110. **Outline**

111. **Box-Shadow**

112. **Filter**

113. **Transform**

114. **Transition**

115. **Animation**

116. **Keyframes**

117. **Font**

118. **Color**

119. **Background**

120. **Border**

121. **Outline**

122. **Box-Shadow**

123. **Filter**

124. **Transform**

125. **Transition**

126. **Animation**

127. **Keyframes**

128. **Font**

129. **Color**

130. **Background**

131. **Border**

132. **Outline**

133. **Box-Shadow**

134. **Filter**

135. **Transform**

136. **Transition**

137. **Animation**

138. **Keyframes**

139. **Font**

140. **Color**

141. **Background**

142. **Border**

143. **Outline**

144. **Box-Shadow**

145. **Filter**

146. **Transform**

147. **Transition**

148. **Animation**

149. **Keyframes**

150. **Font**

151. **Color**

152. **Background**

153. **Border**

154. **Outline**

155. **Box-Shadow**

156. **Filter**

157. **Transform**

158. **Transition**

159. **Animation**

160. **Keyframes**

161. **Font**

162. **Color**

163. **Background**

164. **Border**

165. **Outline**

166. **Box-Shadow**

167. **Filter**

168. **Transform**

169. **Transition**

170. **Animation**

171. **Keyframes**

172. **Font**

173. **Color**

174. **Background**

175. **Border**

176. **Outline**

177. **Box-Shadow**

178. **Filter**

179. **Transform**

180. **Transition**

181. **Animation**

182. **Keyframes**

183. **Font**

184. **Color**

185. **Background**

186. **Border**

187. **Outline**

188. **Box-Shadow**

189. **Filter**

190. **Transform**

191. **Transition**

192. **Animation**

193. **Keyframes**

194. **Font**

195. **Color**

196. **Background**

197. **Border**

198. **Outline**

199. **Box-Shadow**

200. **Filter**

201. **Transform**

202. **Transition**

203. **Animation**

204. **Keyframes**

205. **Font**

206. **Color**

207. **Background**

208. **Border**

209. **Outline**

210. **Box-Shadow**

211. **Filter**

212. **Transform**

213. **Transition**

214. **Animation**

215. **Keyframes**

216. **Font**

217. **Color**

218. **Background**

219. **Border**

220. **Outline**

221. **Box-Shadow**

222. **Filter**

223. **Transform**

224. **Transition**

225. **Animation**

226. **Keyframes**

227. **Font**

228. **Color**

229. **Background**

230. **Border**

231. **Outline**

232. **Box-Shadow**

233. **Filter**

234. **Transform**

235. **Transition**

236. **Animation**

237. **Keyframes**

238. **Font**

239. **Color**

240. **Background**

241. **Border**

242. **Outline**

243. **Box-Shadow**

244. **Filter**

245. **Transform**

246. **Transition**

247. **Animation**

248. **Keyframes**

249. **Font**

250. **Color**

251. **Background**

252. **Border**

253. **Outline**

254. **Box-Shadow**

255. **Filter**

256. **Transform**

257. **Transition**

258. **Animation**

259. **Keyframes**

260. **Font**

261. **Color**

262. **Background**

263. **Border**

264. **Outline**

265. **Box-Shadow**

266. **Filter**

267. **Transform**

268. **Transition**

269. **Animation**

270. **Keyframes**

271. **Font**

272. **Color**

273. **Background**

274. **Border**

275. **Outline**

276. **Box-Shadow**

277. **Filter**

278. **Transform**

279. **Transition**

280. **Animation**

281. **Keyframes**

282. **Font**

283. **Color**

284. **Background**

285. **Border**

286. **Outline**

287. **Box-Shadow**

288. **Filter**

289. **Transform**

290. **Transition**

291. **Animation**

292. **Keyframes**

293. **Font**

294. **Color**

295. **Background**

296. **Border**

297. **Outline**

298. **Box-Shadow**

299. **Filter**

300. **Transform**

301. **Transition**

302. **Animation**

303. **Keyframes**

304. **Font**

305. **Color**

306. **Background**

307. **Border**

308. **Outline**

309. **Box-Shadow**

310. **Filter**

311. **Transform**

312. **Transition**

313. **Animation**

314. **Keyframes**

315. **Font**

316. **Color**

317. **Background**

318. **Border**

319. **Outline**

320. **Box-Shadow**

321. **Filter**

322. **Transform**

323. **Transition**

324. **Animation**

325. **Keyframes**

326. **Font**

327. **Color**

328. **Background**

329. **Border**

330. **Outline**

331. **Box-Shadow**

332. **Filter**

333. **Transform**

334. **Transition**

335. **Animation**

336. **Keyframes**

337. **Font**

338. **Color**

339. **Background**

340. **Border**

341. **Outline**

342. **Box-Shadow**

343. **Filter**

344. **Transform**

345. **Transition**

346. **Animation**

347. **Keyframes**

348. **Font**

349. **Color**

350. **Background**

351. **Border**

352. **Outline**

353. **Box-Shadow**

354. **Filter**

355. **Transform**

356. **Transition**

357. **Animation**

358. **Keyframes**

359. **Font**

360. **Color**

361. **Background**

362. **Border**

363. **Outline**

364. **Box-Shadow**

365. **Filter**

366. **Transform**

367. **Transition**

368. **Animation**

369. **Keyframes**

370. **Font**

371. **Color**

372. **Background**

373. **Border**

374. **Outline**

375. **Box-Shadow**

376. **Filter**

377. **Transform**

378. **Transition**

379. **Animation**

380. **Keyframes**

381. **Font**

382. **Color**

383. **Background**

384. **Border**

385. **Outline**

386. **Box-Shadow**

387. **Filter**

388. **Transform**

389. **Transition**

390. **Animation**

391. **Keyframes**

392. **Font**

393. **Color**

394. **Background**

395. **Border**

396. **Outline**

397. **Box-Shadow**

398. **Filter**

399. **Transform**

400. **Transition**

401. **Animation**

402. **Keyframes**

403. **Font**

404. **Color**

405. **Background**

406. **Border**

407. **Outline**

408. **Box-Shadow**

409. **Filter**

410. **Transform**

411. **Transition**

412. **Animation**

413. **Keyframes**

414. **Font**

415. **Color**

416. **Background**

417. **Border**

418. **Outline**

419. **Box-Shadow**

420. **Filter**

421. **Transform**

422. **Transition**

423. **Animation**

424. **Keyframes**

425. **Font**

426. **Color**

427. **Background**

428. **Border**

429. **Outline**

430. **Box-Shadow**

431. **Filter**

432. **Transform**

433. **Transition**

434. **Animation**

435. **Keyframes**

436. **Font**

437. **Color**

438. **Background**

439. **Border**

440. **Outline**

441. **Box-Shadow**

442. **Filter**

443. **Transform**

444. **Transition**

445. **Animation**

446. **Keyframes**

447. **Font**

448. **Color**

449. **Background**

450. **Border**

451. **Outline**

452. **Box-Shadow**

453. **Filter**

454. **Transform**

455. **Transition**

456. **Animation**

457. **Keyframes**

458. **Font**

459. **Color**

460. **Background**

461. **Border**

462. **Outline**

463. **Box-Shadow**

464. **Filter**

465. **Transform**

466. **Transition**

467. **Animation**

468. **Keyframes**

469. **Font**

470. **Color**

471. **Background**

472. **Border**

473. **Outline**

474. **Box-Shadow**

475. **Filter**

476. **Transform**

477. **Transition**

478. **Animation**

479. **Keyframes**

480. **Font**

481. **Color**

482. **Background**

483. **Border**

484. **Outline**

485. **Box-Shadow**

486. **Filter**

487. **Transform**

488. **Transition**

489. **Animation**

490. **Keyframes**

491. **Font**

492. **Color**

493. **Background**

494. **Border**

495. **Outline**

496. **Box-Shadow**

497. **Filter**

498. **Transform**

499. **Transition**

500. **Animation**

501. **Keyframes**

502. **Font**

503. **Color**

504. **Background**

505. **Border**

506. **Outline**

507. **Box-Shadow**

508. **Filter**

509. **Transform**

510. **Transition**

511. **Animation**

512. **Keyframes**

513. **Font**

514. **Color**

515. **Background**

516. **Border**

517. **Outline**

518. **Box-Shadow**

519. **Filter**

520. **Transform**

521. **Transition**

522. **Animation**

523. **Keyframes**

524. **Font**

525. **Color**

526. **Background**

527. **Border**

528. **Outline**

529. **Box-Shadow**

530. **Filter**

531. **Transform**

532. **Transition**

533. **Animation**

534. **Keyframes**

535. **Font**

536. **Color**

537. **Background**

538. **Border**

539. **Outline**

540. **Box-Shadow**

541. **Filter**

542. **Transform**

543. **Transition**

544. **Animation**

545. **Keyframes**

546. **Font**

547. **Color**

548. **Background**

549. **Border**

550. **Outline**

551. **Box-Shadow**

552. **Filter**

553. **Transform**

554. **Transition**

555. **Animation**

556. **Keyframes**

557. **Font**

558. **Color**

559. **Background**

560. **Border**

561. **Outline**

562. **Box-Shadow**

563. **Filter**

564. **Transform**

565. **Transition**

566. **Animation**

567. **Keyframes**

568. **Font**

569. **Color**

570. **Background**

571. **Border**

572. **Outline**

573. **Box-Shadow**

574. **Filter**

575. **Transform**

576. **Transition**

577. **Animation**

578. **Keyframes**

579. **Font**

580. **Color**

581. **Background**

582. **Border**

583. **Outline**

584. **Box-Shadow**

585. **Filter**

586. **Transform**

587. **Transition**

588. **Animation**

589. **Keyframes**

590. **Font**

591. **Color**

592. **Background**

593. **Border**

594. **Outline**

595. **Box-Shadow**

596. **Filter**

597. **Transform**

598. **Transition**

599. **Animation**

600. **Keyframes**

601. **Font**

602. **Color**

603. **Background**

604. **Border**

605. **Outline**

606. **Box-Shadow**

607. **Filter**

608. **Transform**

609. **Transition**

610. **Animation**

611. **Keyframes**

612. **Font**

613. **Color**

614. **Background**

615. **Border**

616. **Outline**

617. **Box-Shadow**

618. **Filter**

619. **Transform**

620. **Transition**

621. **Animation**

622. **Keyframes**

623. **Font**

624. **Color**

625. **Background**

626. **Border**

627. **Outline**

628. **Box-Shadow**

629. **Filter**

630. **Transform**

631. **Transition**

632. **Animation**

633. **Keyframes**

634. **Font**

635. **Color**

636. **Background**

637. **Border**

638. **Outline**

639. **Box-Shadow**

640. **Filter**

641. **Transform**

642. **Transition**

643. **Animation**

644. **Keyframes**

645. **Font**

646. **Color**

647. **Background**

648. **Border**

649. **Outline**

650. **Box-Shadow**

651. **Filter**

652. **Transform**

653. **Transition**

654. **Animation**

655. **Keyframes**

656. **Font**

657. **Color**

658. **Background**

659. **Border**

660. **Outline**

661. **Box-Shadow**

662. **Filter**

663. **Transform**

664. **Transition**

665. **Animation**

666. **Keyframes**

667. **Font**

668. **Color**

669. **Background**

670. **Border**

671. **Outline**

672. **Box-Shadow**

673. **Filter**

674. **Transform**

675. **Transition**

676. **Animation**

677. **Keyframes**

678. **Font**

679. **Color**

680. **Background**

681. **Border**

682. **Outline**

683. **Box-Shadow**

684. **Filter**

685. **Transform**

686. **Transition**

687. **Animation**

688. **Keyframes**

689. **Font**

690. **Color**

691. **Background**

692. **Border**

693. **Outline**

694. **Box-Shadow**

695. **Filter**

696. **Transform**

697. **Transition**

698. **Animation**

699. **Keyframes**

700. **Font**

701. **Color**

702. **Background**

703. **Border**

704. **Outline**

705. **Box-Shadow**

706. **Filter**

707. **Transform**

708. **Transition**

709. **Animation**

710. **Keyframes**

711. **Font**

712. **Color**

713. **Background**

714. **Border**

715. **Outline**

716. **Box-Shadow**

717. **Filter**

718. **Transform**

719. **Transition**

720. **Animation**

721. **Keyframes**

722. **Font**

723. **Color**

724. **Background**

725. **Border**

726. **Outline**

727. **Box-Shadow**

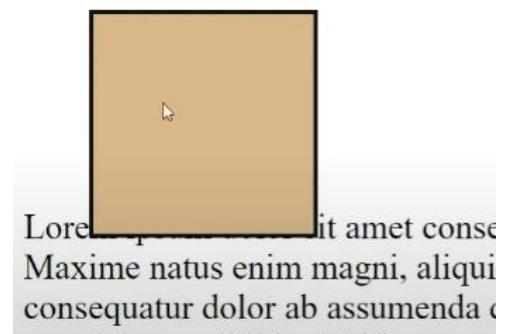
```
index.html    # style.css  X
css > # style.css > .relative
1   .relative
2   {
3     width: 100px;
4     height: 100px;
5     border: 2px solid □rgb(14, 14, 15);
6     background-color: ■burlywood;
7     position: relative;
8
9 }
```

NO : Change for HTML PAGE We Need top ,bottom, left, right

```
style.css > .relative
.relative
{
    width: 100px;
    height: 100px;
    border: 2px solid #rgb(14, 14, 15);
    background-color: #burlywood;
    position: relative;
    top: 30px;
    left: 30px;
}
```

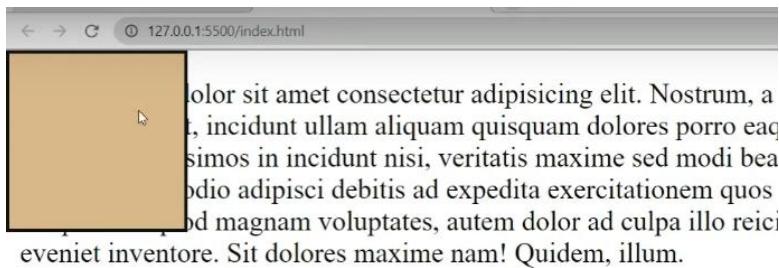
temporibus quia magnam voluptate
eveniet inventore. Sit dolores maxime
natus enim magni, aliquid nulla quaerat nisi!
Facere officia consequatur dolor ab assumenda quis optio recusandae distinctio ut

Change for normal position :→



2. ABSOLUTE:

```
style.css > .relative
1 .relative
2 {
3     width: 100px;
4     height: 100px;
5     border: 2px solid #rgb(14, 14, 15);
6     background-color: #burlywood;
7     position: absolute;
8     top: 0px;
9     left: 0px;
10 }
```



Latin placeholder text (Lorem ipsum) followed by a long string of random characters.

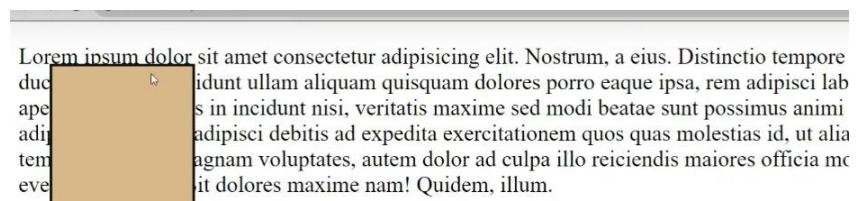
```
css > # style.css > .relative
1  .relative
2  {
3    width: 100px;
4    height: 100px;
5    border: 2px solid #rgb(14, 14, 15);
6    background-color: #burlywood;
7    position: absolute;
8    top: 70px;
9    left: 70px;
10 }
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Nostrum, a eius. Distinctio tempore officiis del
 ducimus sed sit, incidunt ullam aliquam quisquam dolores porro eaque ipsa, rem adipisci laboriosam ra
 aperiam dignissimos in incidunt nisi, veritatis maxime sed modi beatae sunt possimus animi consequat
 adipisci! Sed debitis ad expedita exercitationem quos quas molestias id, ut alias ratione?
 temporibus magnam voluptates, autem dolor ad culpa illo reiciendis maiores officia molestiae ver
 eveniet in nobis. Quod enim dolores maxime nam! Quidem, illum.
 Lorem ipsum dolor sit amet consectetur adipisicing elit. Iure tempora totam quibusdam, quaerat earum
 Maxime natus enim magni, aliquid nulla quaerat nisi! Facere officia fugiat expedita ex est facilis reicie
 consequatur dolor ab assumenda quis optio recusandae distinctio ut animi perspiciatis beatae mollitia!
 vero labore, nihil in debitis atque rem, neque fuga aliquam preferendis explicabo eveniet voluptates at,
 magnam laboriosam exercitationem numquam laborum. Onisnam dolore laudantium aneriam eius est

3. FIXIT

```
index.html      # style.css  ×
css > # style.css > .relative
1  .relative
2  {
3    width: 100px;
4    height: 100px;
5    border: 2px solid #rgb(14, 14, 15);
6    background-color: #burlywood;
7    position: fixed;
8    top: 30px;
9    left: 30px;
10 }
```

Container (Not MOVE) for scrolling down and up



Lorem ipsum dolor sit amet consectetur adipisicing elit. Iure tempora totam quibusdam, qua
 Maxime natus enim magni, aliquid nulla quaerat nisi! Facere officia fugiat expedita ex est fa
 consequatur dolor ab assumenda quis optio recusandae distinctio ut animi perspiciatis beatae

4. Sticky

aperiam dignissimos in incident nisi, veritatis maxime sed modi beatae sunt possimus animi cc adipisci! Sunt odio adipisci debitis ad expedita exercitationem quos quas molestias id, ut alias temporibus quod magnam voluptates, autem dolor ad culpa illo reiciendis maiores officia mole eveniet inventore. Sit dolores maxime nam! Quidem, illum.



Lorem ipsum dolor sit amet consectetur adipisicing elit. Iure tempora totam quibusdam, quaerat Maxime natus enim magni, aliquid nulla quaerat nisi! Facere officia fugiat expedita ex est facilis consequatur dolor ab assumenda quis optio recusandae distinctio ut animi perspiciatibus beatae non vero labore, nihil in debitis atque rem, neque fuga aliquam perferendis explicabo eveniet voluptatum magnam laboriosam exercitationem numquam laborum. Quisquam dolore laudantium aperiam

Orginal is

```
index.html # style.css X
css > # style.css > .relative
1   .relative
2   {
3     width: 100px;
4     height: 100px;
5     border: 2px solid #rgb(14, 14, 15);
6     background-color: #burlywood;
7     position: sticky;
8     top: 30px;
9     left: 30px;
10 }
```

page scrolling but container fix top and left.(automatically space)

Lor
Ma
con
ver
mag

sit amet consectetur adipisicing elit. Iure tempora totam quibusdam, quaerat Maxime natus enim magni, aliquid nulla quaerat nisi! Facere officia fugiat expedita ex est facilis consequatur dolor ab assumenda quis optio recusandae distinctio ut animi perspiciatibus beatae non vero labore, nihil in debitis atque rem, neque fuga aliquam perferendis explicabo eveniet voluptatum magnam laboriosam exercitationem numquam laborum. Quisquam dolore laudantium aperiam

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Illo libero ab dicta quae eos illum dolore praesentium, alias nulla esse repudiandae vel nemo ipsam enim, tenetur non accusantium temporibus exercitationem

- Display

The screenshot shows a code editor with an open file named 'hello.html'. The code contains an HTML structure with an

element and a element, both with inline styles applied via a CSS block. The browser preview on the right shows the rendered output where the element has a blue background and the element has a red background.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<style>
    h1 {
        background-color: #aqua;
    }
    span {
        background-color: #red;
    }
</style>
<body>
    <h1>234567</h1>
    <span>45678</span>
</body>
</html>
```

Use display inline in h1

The screenshot shows a code editor with files 'index.html', 'styles.css', and 'script.js'. The 'styles.css' file contains CSS rules for '.block' and '.inline' classes. The '.block' rule sets 'display: block', while the '.inline' rule sets 'display: inline'. A browser preview on the right shows the rendered output where the 'Block element' is a full-width blue box and the 'Inline element' is a smaller red box.

```
.block {
    display: block; /* Block elements occupy full width */
    background-color: lightblue;
    margin-bottom: 10px;
}
.inline {
    background-color: lightcoral;
}
```

Use span tag full fill color (like div)

index.html styles.css script.js + 43235ca68 NEW HTML RUN

```
1 <div class="block">Block element</div>
2 <span class="inline">Inline element</span>
3
4 <style>
5   .block {
6     background-color: lightblue;
7     margin-bottom: 10px;
8
9   }
10  .inline {
11    background-color: lightcoral;
12    display: block;
13  }
14 </style>
15
```

The screenshot shows a code editor with three tabs: index.html, styles.css, and script.js. The styles.css tab is active, displaying the following CSS code:

```
1 <div class="block">Block element</div>
2 <span class="inline">Inline element</span>
3
4 <style>
5   .block {
6     background-color: lightblue;
7     margin-bottom: 10px;
8     display: inline;
9   }
10  .inline {
11    background-color: lightcoral;
12    display: block;
13  }
14 </style>
15
```

To the right of the code editor, there are two colored boxes: a blue box labeled "Block element" and a red box labeled "Inline element".

index.html styles.css script.js + 43235ca68

```
1 <div class="block">Block element</div>
2 <span class="inline">Inline element</span>
3
4 <style>
5   .block {
6     background-color: lightblue;
7     margin-bottom: 10px;
8     display: inline;
9   }
10  .inline {
11    background-color: lightcoral;
12    display: block;
13  }
14 </style>
15
```

The screenshot shows a code editor with three tabs: index.html, styles.css, and script.js. The styles.css tab is active, displaying the same CSS code as the previous screenshot, but with a different styling result.

To the right of the code editor, there are two colored boxes: a blue box labeled "Block element" and a red box labeled "Inline element".

DISPLAY NONE ->

index.html styles.css script.js + 43235ca68

```
1 <div class="block">Block element</div>
2 <span class="inline">Inline element</span>
3
4 <style>
5   .block {
6     background-color: lightblue;
7     margin-bottom: 10px;
8     display: inline;
9   }
10  .inline {
11    background-color: lightcoral;
12    /*display: block;*/
13    display: none;
14  }
15 </style>
16
```

The screenshot shows a code editor with three tabs: index.html, styles.css, and script.js. The styles.css tab is active, displaying the following CSS code:

```
1 <div class="block">Block element</div>
2 <span class="inline">Inline element</span>
3
4 <style>
5   .block {
6     background-color: lightblue;
7     margin-bottom: 10px;
8     display: inline;
9   }
10  .inline {
11    background-color: lightcoral;
12    /*display: block;*/
13    display: none;
14  }
15 </style>
16
```

To the right of the code editor, there is one colored box: a blue box labeled "Block element". The "Inline element" box is missing, indicating that the element is not visible due to the "display: none" rule.



```
1 <html>
2 <head>
3   <style>
4     ul{
5       background-color:yellow;
6     }
7   </style>
8 </head>
9 <body>
10 <ul>
11   <li>Home</li>
12   <li>About Us</li>
13   <li>Contact Us</li>
14 </ul>
15
16
17 </body>
18 </html>
```

- Home
- About Us
- Contact Us

```
1 <html>
2 <head>
3   <style>
4     ul{
5       background-color:yellow;
6       list-style-type:none;
7     }
8   </style>
9 </head>
10 <body>
11 <ul>
12   <li>Home</li>
13   <li>About Us</li>
14   <li>Contact Us</li>
15 </ul>
16
17
18
```

- Home
- About Us
- Contact Us

```
1 <html>
2 <head>
3   <style>
4     ul{
5       background-color:yellow;
6       list-style-type:none;
7     }
8
9
10 </style>
11 </head>
12 <body>
13
14 <ul>
15   <li>Home</li>
16   <li>About Us</li>
17   <li>Contact Us</li>
18 </ul>
```

Home
About Us
Contact Us

The screenshot shows a Notepad++ window titled "css layout display property". The code editor contains the following CSS and HTML:

```
1 <html>
2 <head>
3   <style>
4     ul{
5       background-color:yellow;
6       list-style-type:none;
7     }
8
9     ul li{
10       background-color:red;
11     }
12
13   </style>
14 </head>
15 <body>
16
17   <ul>
18     <li>Home</li>
```

To the right, a browser window titled "YouTube" shows a red header bar with white text: "Home", "About Us", and "Contact Us".

UI->li Color change

The screenshot shows a code editor with the URL "http://127.0.0.1:3000/hello" in the address bar. The code is as follows:

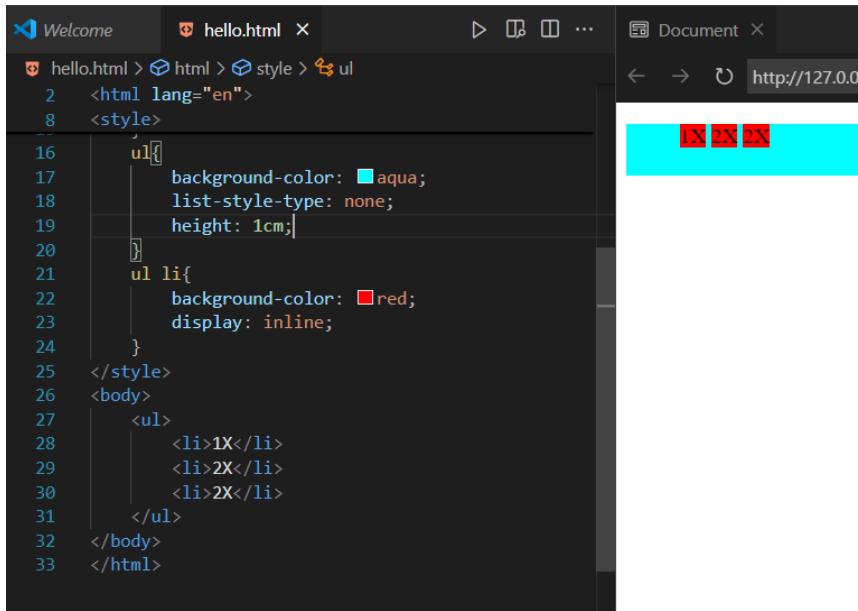
```
1 <html lang="en">
2   <style>
3     ul{
4       background-color:aqua;
5       list-style-type: none;
6     }
7     ul li{
8       background-color:red;
9     }
10    </style>
11  <body>
12    <ul>
13      <li>1X</li>
14      <li>2X</li>
15      <li>2X</li>
16    </ul>
17  </body>
18 </html>
```

The browser preview shows a red background with white text: "1X", "2X", and "2X".

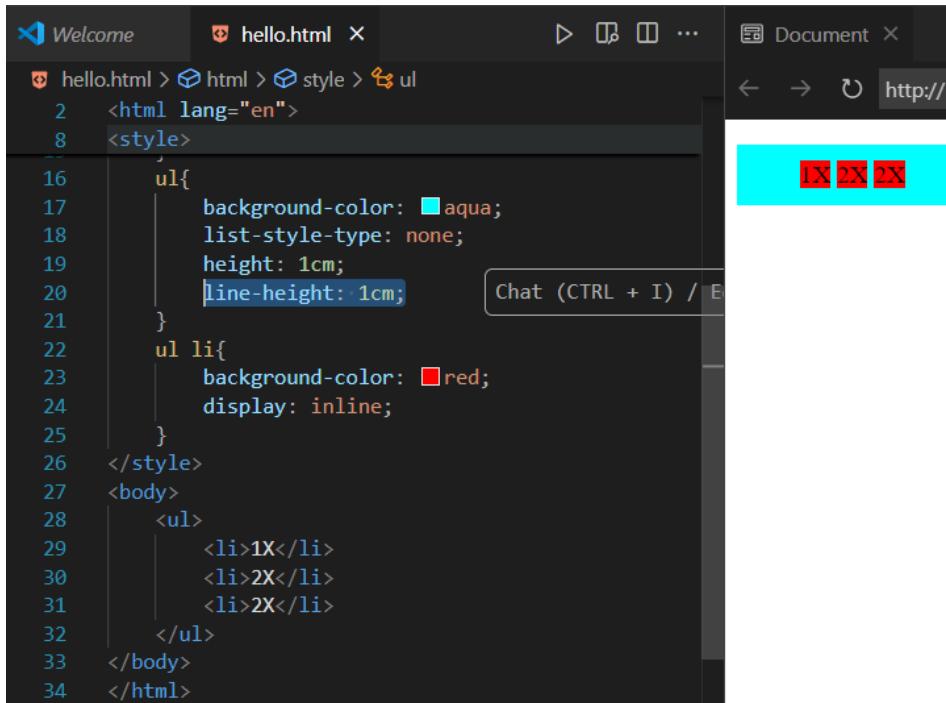
The screenshot shows a code editor with the URL "http://127.0.0.1:3000/hello" in the address bar. The code is identical to the previous one, but the "display: inline;" part of the CSS rule is highlighted with a tooltip: "Chat (CTRL + I) / Ed".

```
1 <html lang="en">
2   <style>
3     ul{
4       background-color:aqua;
5       list-style-type: none;
6     }
7     ul li{
8       background-color:red;
9       display: inline;
10    }
11    </style>
12  <body>
13    <ul>
14      <li>1X</li>
15      <li>2X</li>
16      <li>2X</li>
17    </ul>
18  </body>
19 </html>
```

Height & Line height ->



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <ul>
        <li>1X</li>
        <li>2X</li>
        <li>2X</li>
    </ul>
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <ul>
        <li>1X</li>
        <li>2X</li>
        <li>2X</li>
    </ul>
</body>
</html>
```

Use inline block ->

The screenshot shows a code editor on the left and a browser window on the right. The code editor displays the following HTML and CSS:

```
1<html lang="en">
2 <head>
3   <style>
4     ul{
5       background-color: #aqua;
6       list-style-type: none;
7       height: 1cm;
8       line-height: 1cm;
9     }
10    ul li{
11      background-color: #red;
12      display: inline-block;
13    }
14  </style>
15  <body>
16    <ul>
17      <li>1X</li>
18      <li>2X</li>
19      <li>2X</li>
20    </ul>
21  </body>
22</html>
```

The browser window shows a horizontal list of three items: "1X", "2X", and "2X". Each item is a red square with white text, and they are separated by thin vertical lines, indicating they are inline-block elements.

Add padding left& Right

The screenshot shows a code editor on the left and a browser window on the right. The code editor displays the same HTML and CSS as the previous screenshot, but with additional padding:

```
1<html lang="en">
2 <head>
3   <style>
4     ul{
5       height: 1cm;
6       line-height: 1cm;
7     }
8     ul li{
9       background-color: #red;
10      display: inline-block;
11      padding-left: 1cm;
12      padding-right: 1cm;
13    }
14  </style>
15  <body>
16    <ul>
17      <li>1X</li>
18      <li>2X</li>
19      <li>2X</li>
20    </ul>
21  </body>
22</html>
```

The browser window shows the same list of items, but now each item has a 1cm padding on both the left and right sides, resulting in a wider total width for each item.

index.html styles.css script.js + 43235ca68 NEW HTML RUN

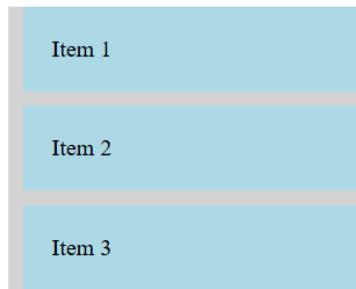
```
1 <span class="inline">Inline element</span>
2 <div class="inline1">Inline element</div>
3 <div class="inline-block">Inline-block element</div>
4 <style>
5   .inline {
6     background-color: lightcoral;
7   }
8   .inline1 {
9     display: inline-block;
10    background-color: lightcoral;
11  }
12  .inline-block {
13    display: inline-block;
14    width: 200px;
15    height: 50px;
16    background-color: lightgreen;
17  }
18 </style>
```



Display flex->

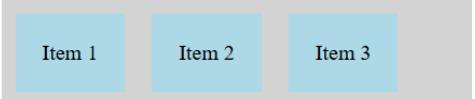
index.html styles.css script.js + 43235ca68

```
1 <div class="flex-container">
2   <div class="flex-item">Item 1</div>
3   <div class="flex-item">Item 2</div>
4   <div class="flex-item">Item 3</div>
5 </div>
6
7 <style>
8   .flex-container {
9     background-color: lightgrey;
10  }
11
12  .flex-item {
13    background-color: lightblue;
14    padding: 20px;
15    margin: 10px;
16  }
17 </style>
18
19
```



index.html styles.css script.js + 43235ca68 NEW

```
1 <div class="flex-container">
2   <div class="flex-item">Item 1</div>
3   <div class="flex-item">Item 2</div>
4   <div class="flex-item">Item 3</div>
5 </div>
6
7 <style>
8   .flex-container {
9     display: flex;
10    background-color: lightgrey;
11  }
12
13  .flex-item {
14    background-color: lightblue;
15    padding: 20px;
16    margin: 10px;
17  }
18 </style>
19
```



Inline display flex

A screenshot of a code editor interface. On the left, there are three tabs: 'index.html', 'styles.css', and 'script.js'. The 'index.html' tab shows the following HTML code:

```
1 <div class="inline-flex-container">
2   <div class="flex-item">Item 1</div>
3   <div class="flex-item">Item 2</div>
4   <div class="flex-item">Item 3</div>
5 </div>
6
7 <style>
8   .inline-flex-container {
9     display: inline-flex;
10    background-color: lightgrey;
11  }
12
13  .flex-item {
14    background-color: lightblue;
15    padding: 50px;
16    margin: 10px;
17  }
18 </style>
19
```

The 'styles.css' tab contains the following CSS code:

```
.inline-flex-container {
  display: inline-flex;
  background-color: lightgrey;
}

.flex-item {
  background-color: lightblue;
  padding: 50px;
  margin: 10px;
}
```

On the right side of the editor, there is a preview window showing three lightblue rectangular boxes labeled 'Item 1', 'Item 2', and 'Item 3' arranged horizontally. The preview window has a light grey border and is set against a white background.

FLOAT →

A screenshot of a code editor interface. On the left, there are three tabs: 'index.html', 'styles.css', and 'script.js'. The 'index.html' tab shows the following HTML code:

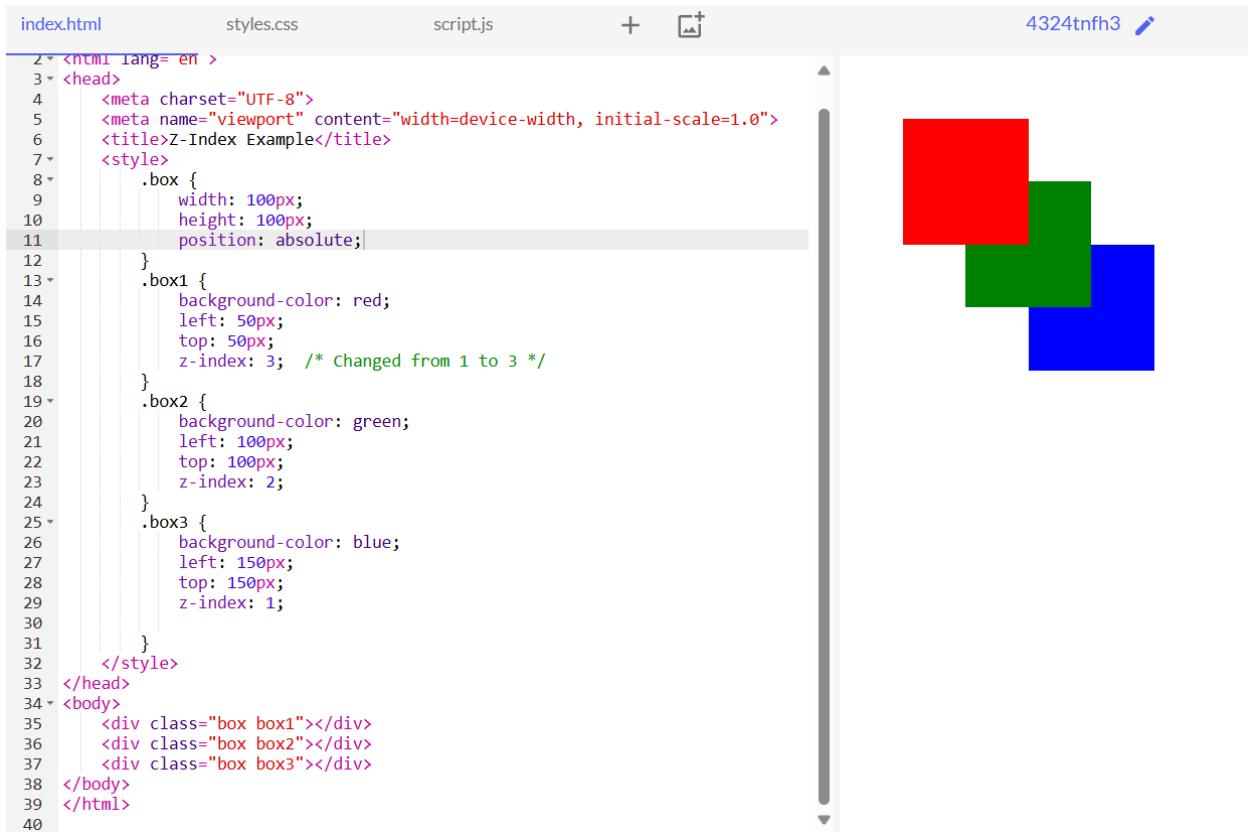
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>Float Example</title>
5   <style>
6     .float-left {
7       float: left;
8       margin: 100px;
9     }
10  </style>
11 </head>
12 <body>
13   
14   <p>
15     Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus lacin
16     Cras venenatis euismod malesuada. Nullam ac erat ante. Phasellus digni
17     bibendum ac. Fusce tincidunt, ligula et facilisis facilisis, purus sap
18     id lectus.
19   </p>
20 </body>
21 </html>
22
```

The 'styles.css' tab contains the following CSS code:

```
.float-left {
  float: left;
  margin: 100px;
}
```

On the right side of the editor, there is a preview window showing a lightblue rectangular box labeled 'Example Image' floating to the left of a block of text. The preview window has a light grey border and is set against a white background.

Z-indexing



```
index.html          styles.css          script.js      +  4324tnfh3  ↗
2 <!DOCTYPE html>
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Z-Index Example</title>
7   <style>
8     .box {
9       width: 100px;
10      height: 100px;
11      position: absolute;
12    }
13    .box1 {
14      background-color: red;
15      left: 50px;
16      top: 50px;
17      z-index: 3; /* Changed from 1 to 3 */
18    }
19    .box2 {
20      background-color: green;
21      left: 100px;
22      top: 100px;
23      z-index: 2;
24    }
25    .box3 {
26      background-color: blue;
27      left: 150px;
28      top: 150px;
29      z-index: 1;
30    }
31  </style>
32 </head>
33 <body>
34   <div class="box box1"></div>
35   <div class="box box2"></div>
36   <div class="box box3"></div>
37 </body>
38 </html>
39
40
```

Example: Basic Flexbox Layout

This example will show a container with three items, centered both horizontally and vertically using Flexbox.

1. Simplified Layouts

Flexbox simplifies the process of creating both simple and complex layouts. It provides a set of properties that allow you to align and distribute space among items in a container, making it easier to build layouts without using floats or positioning.

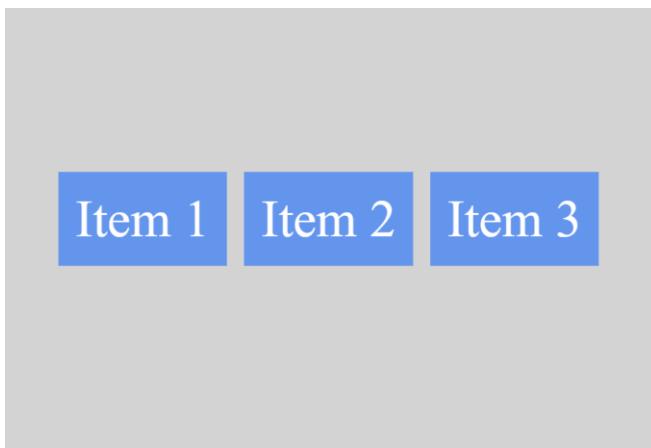
2. Responsive Design

Flexbox makes it easy to create responsive designs that adapt to different screen sizes and orientations. The flexible nature of Flexbox allows items to adjust and distribute space dynamically based on the container size.

3. Centralized Alignment

One of the most powerful features of Flexbox is its ability to align items both horizontally and vertically with ease. Properties like `justify-content` and `align-items` provide simple solutions for centering elements.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Basic Flexbox Layout</title>
7  <style>
8      .container {
9          display: flex; /* Establishes a flex container */
10         justify-content: center; /* Centers items horizontally */
11         align-items: center; /* Centers items vertically */
12         height: 100vh; /* Full viewport height */
13         background-color: lightgray; /* Background color for the container */
14     }
15     .item {
16         background-color: cornflowerblue; /* Background color for items */
17         padding: 20px; /* Padding inside items */
18         margin: 10px; /* Margin around items */
19         color: white; /* Text color */
20         font-size: 3.8rem; /* Font size */
21     }
22 </style>
23 </head>
24 <body>
25     <div class="container">
26         <div class="item">Item 1</div>
27         <div class="item">Item 2</div>
28         <div class="item">Item 3</div>
29     </div>
30 </body>
31 </html>
32
```



- --> Responsive Design
- Media Queries
- Viewport
- Responsive Units (vw, vh, %, etc.)

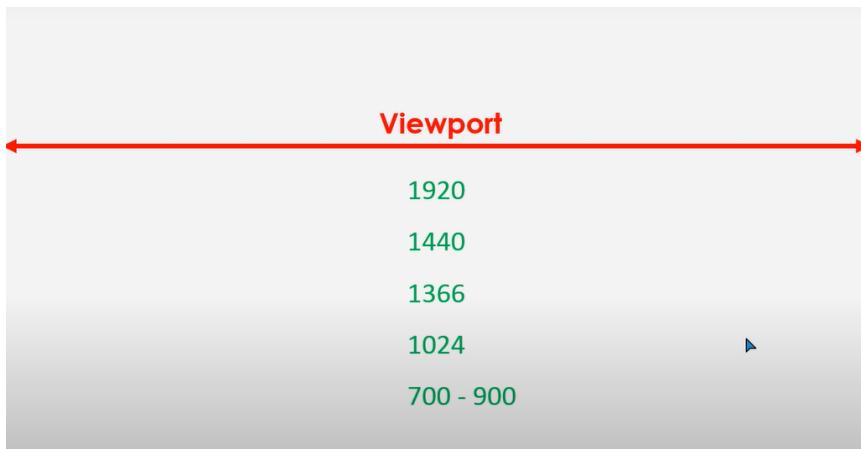
Media Queries

CSS Media Types

▶▶▶

- All
- Print
- `<Screen>`
- Speech

View port -> width for Screen ,(tab,phone,laptop,PC)



```
@media screen and (max-width: 900px) {
```

```
}
```

```
@media screen and (max-width: 900px) {
```

```
}
```

```
@media screen and (max-width: 900px) {  
}
```

Break Point

```
@media screen and (max-width: 900px) {  
  .container {  
    width: 50%;  
  }  
}
```

- any-hover
- inverted-colors
- min-aspect-ratio
- overflow-block
- any-pointer
- light-level
- min-color
- overflow-inline
- aspect-ratio
- max-aspect-ratio
- min-color-index
- pointer
- color
- max-color
- min-height
- resolution
- color-gamut
- max-color-index
- min-monochrome
- scan
- color-index
- max-height
- min-resolution
- scripting
- grid
- max-monochrome
- min-width
- update
- height
- max-resolution
- monochrome
- width
- hover
- max-width
- orientation

```
<html>
  <head>
    <title>@media Query</title>
    <style>
      body{
        background: white;
        font-family:arial;
      }
      @media screen and (max-width:1200px){
        body{
          background: pink;
        }
      }
    </style>
  </head>

  <body>
    <h1>Yahoo Baba : CSS @media Query</h1>
  </body>
</html>
```



Multiple Condition apply for

```
<html>
  <head>
    <title>@media Query</title>
    <style>
      body{
        background: white;
        font-family: arial;
      }
      @media screen and (max-width:1200px) {
        body{
          background: pink;
        }
      }
      @media screen and (max-width:1000px) {
        body{
          background: lightblue;
        }
      }
      @media screen and (max-width:700px) {
        body{
          background: orange;
        }
      }
    </style>
  </head>

  <body>
    <h1>Yahoo Baba : CSS @media Query</h1>
```



ADD Font Size →

```
<html>
  <head>
    <title>@media Query</title>
    <style>
      body{
        background: white;
        font-family:arial;
        font-size:100px;
      }
      @media screen and (max-width:1200px){
        body{
          background: pink;
          font-size:100px;
        }
      }
      @media screen and (max-width:1000px){
        body{
          background: lightblue;
        }
      }
      @media screen and (max-width:700px){
        body{
          background: orange;
        }
      }
      @media screen and (max-width:480px){
        body{
          background: red;
          font-family:arial;
          font-size:100px;
        }
      }
      @media screen and (max-width:1200px){
        body{
          background: pink;
          font-size:70px;
        }
      }
      @media screen and (max-width:1000px){
        body{
          background: lightblue;
          font-size:50px;
        }
      }
      @media screen and (max-width:700px){
        body{
          background: orange;
          font-size:100px;
        }
      }
      @media screen and (max-width:480px){
        body{
          background: red;
        }
      }
    </style>
```

The screenshot shows a code editor with two tabs: "hello1.html" and "Document". The "hello1.html" tab displays the following HTML and CSS code:

```
me      hello.html      hello1.html X  ▶  ⌂  ⌂  ...  Document X
      http://127.0.0.1:3000

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
6       initial-scale=1.0">
7     <title>Document</title>
8   </head>
9   <style>
10    body{
11      background-color: #rgb(187, 0, 0);
12      font-size: 50px;
13    }
14    @media screen and (min-width: 468px) {
15      body{
16        background-color: #blue;
17        font-size: 40px;
18      }
19    }
20    @media screen and (min-width: 668px) {
21      body{
22        background-color: #rgb(3, 255, 112);
23        font-size: 10px;
24      }
25    }
26  </style>
27  <body>
28    <h1>
29      hello1234567890
30    </h1>
31
32  </body>
33  </html>
```

The "Document" tab shows the browser output with the text "hello1234567890" displayed in a large, red font.

O/p ?

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Responsive Line Height</title>
7      <style>
8          /* Base Styles */
9          body {
10              font-family: Arial, sans-serif;
11              margin: 0;
12              padding: 0;
13          }
14          .text-container {
15              width: 80%;
16              padding: 20px;
17          }
18          p {
19              line-height: 1.5;
20          }
21          @media only screen and (max-width: 600px) {
22              p {
23                  background-color: red ;
24                  line-height: 1.2;
25              }
26          }
27
28          @media only screen and (min-width: 601px) and (max-width: 900px) {
29              p {
30                  background-color: yellow ;
31                  line-height: 1.6;
32              }
33          }
34
35          @media only screen and (min-width: 901px) {
36              p {
37                  line-height: 2;
38              }
39          }
40      </style>
41  </head>
42  <body>
43      <div class="text-container">
44          <p>This is a paragraph with responsive line height. Resize the browser window to see how the line height changes based on the screen size.</p>
45      </div>
46  </body>
47 </html>
48
```

• Viewport

Using Viewport Units:

CSS viewport units jaise `vw` (**viewport width**), `vh` (**viewport height**), `vmin` aur `vmax` ko use karke tum layout ko responsive aur flexible bana sakte ho. Yeh units viewport ke size ke relative hote hain, jo responsive design mein kaam aate hain.

Example:

Css

```
.box {  
    width: 50vw; /* 50% of the viewport width */  
    height: 30vh; /* 30% of the viewport height */  
    background-color: lightcoral;  
    padding: 5vh;  
    margin: 5vmin;  
    font-size: 2vmax; /* 2% of the larger viewport dimension */  
    text-align: center;  
}
```

index.html

styles.css

script.js

+



```
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4      <meta charset="UTF-8">  
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">  
6      <title>Viewport Units Example</title>  
7      <style>  
8          body {  
9              font-family: Arial, sans-serif;  
10             margin: 0;  
11             padding: 0;  
12         }  
13  
14         .box {  
15             width: 50vw; /* 50% of the viewport width */  
16             height: 30vh; /* 30% of the viewport height */  
17             background-color: lightcoral;  
18             padding: 5vh;  
19             margin: 5vmin;  
20             font-size: 2vmax; /* 2% of the larger viewport dimension */  
21             text-align: center;  
22             display: flex;  
23             align-items: center;  
24             justify-content: center;  
25         }  
26     </style>  
27 </head>  
28 <body>  
29     <div class="box">  
30         <p>This box uses vw, vh, vmin, and vmax units to adapt to the viewport size.</p>  
31     </div>  
32 </body>  
33 </html>
```

CSS Transitions aur Animations ke differences ko table ke format mein diya gaya hai:

| Aspect | Transitions | Animations |
|-----------------------|--|--|
| Purpose | State changes se simple effects create karna | Complex sequences aur multiple stages ko define karna |
| Trigger | User interactions (hover, click, focus) se trigger hoti hain | Automatically start ho sakti hain ya state changes se trigger ho sakti hain |
| Keyframes | Keyframes use nahi karte | Keyframes use karte hain |
| Control Over Timeline | Sirf start aur end properties ko duration ke upar specify kar sakte hain | Granular control offer karte hain animation stages par keyframes ke through |
| Flexibility | Less flexible, simple effects ke liye suitable | Highly flexible, complex motion aur transformation sequences ke liye suitable |
| Syntax | <code>transition</code> property use karte hain | <code>animation</code> property aur <code>@keyframes</code> use karte hain |
| Iteration | Typically ek baar run karte hain per state change | Infinite loop ya specific number of times run ho sakte hain |
| Complexity | Basic effects ke liye best (fading, scaling, position changes) | Intricate sequences (moving objects, complex interactions) create karne ke liye best |

| | | |
|--------------------|--|--|
| Syntax | <code>transition</code> property use karte hain | <code>animation</code> property aur <code>@keyframes</code> use karte hain |
| Iteration | Typically ek baar run karte hain per state change | Infinite loop ya specific number of times run ho sakte hain |
| Complexity | Basic effects ke liye best (fading, scaling, position changes) | Intricate sequences (moving objects, complex interactions) create karne ke liye best |
| Performance | Often more performant due to simplicity aur limited scope | Resource-intensive ho sakti hain, especially with complex keyframe animations |
| Ease of Use | Basic effects ke liye implement karna easy, fewer lines of code required | Keyframes aur complex effects ke liye more code aur understanding required |

- calc()
- var()
- url()

→Calc ()

Concept: `calc()` function dynamically calculate karne ka feature deta hai. Iska matlab, hum different units ko combine karke ek value bana sakte hain jo container aur viewport ke size ke according adjust hoti hai.

`calc()` function tumhe CSS properties ke values ko calculate karne ki facility deta hai. Iska use karke tum different units jaise pixels (px), percentages (%), viewport units (vw, vh), etc., ko combine karke ek dynamic value bana sakte ho.

Without calc() -> page

```

cc.html > html > head > style
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
6     initial-scale=1.0">
6   <title>Without calc() Example</title>
7   <style>
8     .box {
9       width: 200px; /* Fixed width */
10    height: 150px; /* Fixed height */
11    background-color: lightcoral;
12  }
13 </style>
14 </head>
15 <body>
16   <div class="box"></div>
17 </body>
18 </html>
19

```

With CALC()->

```

cc.html > html > head > style > .box
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>With calc() Example</title>
7   <style>
8     .box {
9       width: calc(100% - 40px); /* 100% width of container minus 40px */
10    height: calc(100px + 20vh); /* 100px plus 20% of the viewport height */
11    background-color: lightcoral;
12  }
13 </style>
14 </head>
15 <body>
16   <div class="box"></div>
17 </body>
18 </html>
19

```

Var Example :

```

index.html      styles.css      script.js      +      4329bn5pc
This box uses the var() and calc() functions.
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Using var() and calc()</title>
7   <style>
8     :root {
9       --box-width-offset: 40px;
10    --box-height-base: 100px;
11    --box-height-vh: 20vh;
12    --box-bg-color: lightcoral;
13  }
14  .box {
15    width: calc(100% - var(--box-width-offset)); /* 100% width minus variable offset */
16    height: calc(var(--box-height-base) + var(--box-height-vh));
17    /* Variable base
18    height plus variable viewport height */
19    background-color: var(--box-bg-color);
20  }
21 </style>
22 </head>
23 <body>
24   <div class="box">This box uses the var() and calc() functions.</div>
25 </body>
26 </html>
27

```

Another example

index.html styles.css script.js + 4329bn5pc NEW

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Using var()</title>
7 <style>
8   :root {
9     --main-color: lightblue;
10    --padding-size: 20px;
11  }
12
13 .var-box {
14   background-color: var(--main-color);
15   padding: var(--padding-size);
16   width: calc(100% - 40px);
17   height: calc(50px + 15vh);
18 }
19 </style>
20 </head>
21 <body>
22 <div class="var-box">
23   This box uses the var() function to apply CSS variables.
24 </div>
25 </body>
26 </html>
27

```

This box uses the var() function to apply CSS variables.

Theory: url() function is used to include external resources, such as images, fonts, or other files in your

url_css_.html > html > head > style > .url-box

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Using url()</title>
7 <style>
8   .url-box {
9     width: 100%;
10    height: 300px;
11    background-image: url("https://th.bing.com/th/id/R.ae3d2f4880eca12be7c84e2d2c56041?rik=KeMl0VtDSYUt2g&rlu=http%3a%2f%2fgetwallpapers.com%2fwallpaper%2ffull%2fb%2f1%2f9%2f355093.jpg&ehk=oGlwjQg0%2fGGKmJsbxLkvu%2bk2XY18CIw6bAAeo60VPw%3d&rls=pid=ImgRaw&r=0");
12    background-size: cover;
13    background-position: center;
14  }
15 </style>
16 </head>
17 <body>
18   <div class="url-box">
19     This box uses the url() function to set a background image.
20   </div>
21 </body>
22 </html>
23

```



CSS.

The BOX is use Click:

index.html styles.css script.js + 4329e5xwd

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Open Google</title>
7   <style>
8     .link-box {
9       width: 100%;
10      height: 300px;
11      background-image:url(https://th.bing.com/th/id/R.ae3d2f4880eaca12be7c84e2d2c56041?rik=KeMleTVdSVt2g&rlv=http%3a%2f%2fgetwallpaper.com%2fwallpaper%2ffull%2fb%2f1%2f9%2f355093.jpg&hkm=G1wjqG0%2FFGGKmJsbxLkvu%2bk2XY18CIw6BA Ae66QVpw%3d&risl=&pid=ImgRaw&r=0);
12      display: flex;
13      justify-content: center;
14      align-items: center;
15      text-decoration: none;
16      color: white;
17      font-size: 2rem;
18    }
19  </style>
20 </head>
21 <body>
22   <a href="https://www.google.com" target="_blank" class="link-box">
23     Click Here to Open Google
24   </a>
25 </body>
26 </html>

```

| Feature | <code>calc()</code> | <code>var()</code> | <code>url()</code> |
|------------------|---|--|---|
| Purpose | Used to perform mathematical calculations to determine CSS property values. | Used to reference and apply custom properties (CSS variables) defined earlier in the stylesheet. | Used to specify and include external resources, such as images, fonts, or videos. |
| Syntax | <code>calc(expression)</code> (e.g., <code>calc(100% - 50px)</code>) | <code>var(--custom-property, fallback)</code> (e.g., <code>var(--main-color, blue)</code>) | <code>url(path-to-resource)</code> (e.g., <code>url('image.jpg')</code>) |
| Use Case | To dynamically calculate values like widths, margins, or paddings. | To create reusable and maintainable styles with custom variables. | To link external files like background images or fonts. |
| Dynamic Behavior | Allows combining different units (e.g., percentages, pixels). | Relies on a previously declared <code>--custom-property</code> . Provides fallback if the variable is unset. | Simply resolves the URL to fetch or load the resource. |
| Example | <code>width: calc(100% - 50px);</code> | <code>color: var(--text-color, black);</code> | <code>background-image: url('background.jpg');</code> |

- **CSS Preprocessors (LESS, SASS)**
- **CSS Frameworks (Bootstrap, Tailwind)**
- **CSS-in-JS**
- **Custom Properties (CSS Variables)**
- **CSS Preprocessors (LESS, SASS)**

CSS preprocessors like LESS and SASS extend the capabilities of regular CSS by introducing features such as variables, nesting, and mixins. These preprocessors enable more efficient and maintainable styling through enhanced syntax and functionalities before compiling into standard CSS.

CSS Frameworks (Bootstrap, Tailwind)

CSS frameworks like Bootstrap and Tailwind provide pre-designed, responsive components and utility classes, accelerating web development and ensuring consistency. Bootstrap offers a robust component library, while Tailwind focuses on utility-first styling for more customization.

CSS-in-JS

CSS-in-JS refers to writing CSS directly within JavaScript files, enabling styles to be scoped to individual components. This approach, often used in React with libraries like styled-components and Emotion, promotes modular and reusable styling by integrating tightly with JavaScript logic.

Custom Properties (CSS Variables)

CSS variables, also known as custom properties, allow developers to store values that can be reused throughout a document. Defined using `--var-name` and accessed with `var(--var-name)`, they provide easier theme management and maintainability in CSS.

task 1 :

Create →

Create a webpage with a header, a paragraph, and a button. Style the header with

a different color, the paragraph with a specific font size and margin, and the button with a background color and some padding.

Welcome to My Simple Page

This is a simple paragraph to demonstrate HTML and CSS styling.

Click Me!

The screenshot shows a code editor interface with two tabs: "index.html" and "styles.css". The "index.html" tab contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Simple Task</title>
5   <link rel="stylesheet" type="text/css" href="styles.css">
6 </head>
7 <body>
8   <h1>Welcome to My Simple Page</h1>
9   <p>This is a simple paragraph to demonstrate HTML and CSS styling.</p>
10  <button>Click Me!</button>
11 </body>
12 </html>
```

The "styles.css" tab contains the following CSS code:

```
1 /* style for the header */
2 h1 {
3   color: blue;
4 }
5 /* style for the paragraph */
6 p {
7   font-size: 18px;
8   margin: 20px 0;
9 }
10 /* style for the button */
11 button {
12   background-color: coral;
13   padding: 10px 20px;
14   border: none;
15   color: white;
16   cursor: pointer;
17 }
```

Welcome to My Simple Page

This is a simple paragraph to demonstrate HTML and CSS styling.

Click Me!

The screenshot shows a code editor interface with two tabs: "index.html" and "styles.css". The "index.html" tab contains the same HTML code as the previous screenshot.

The "styles.css" tab contains the same CSS code as the previous screenshot.

Welcome to My Simple Page

This is a simple paragraph to demonstrate HTML and CSS styling.

Click Me!