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.

```html
<!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

```html
<!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.

```html
<!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.

```html
<!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.

```html
<!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.

```html
<!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 lis
        }
    </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

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tables</title>
    <style>
        table {
            width: 100%;
            border-collapse: collapse; /* Remove space between cells */
        }

        th, td {
            border: 1px solid black; /* Cell borders */
            padding: 10px; /* Cell padding */
            text-align: left; /* Left align text */
        }

        th {
            background-color: #f2f2f2; /* Header background */
        }
    </style>
</head>
<body>
```

```
        background-color: #f2f2f2; /*
    }
</style>
</head>
<body>
    <table>
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>City</th>
        </tr>
        <tr>
            <td>Alice</td>
            <td>25</td>
            <td>New York</td>
        </tr>
        <tr>
            <td>Bob</td>
            <td>30</td>
            <td>San Francisco</td>
        </tr>
    </table>
</body>
</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:

```html
<!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.

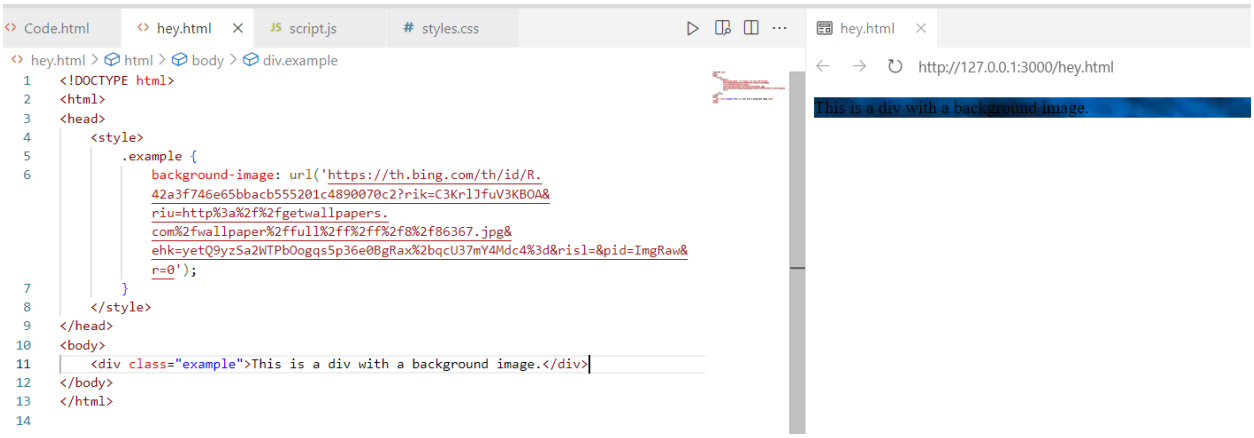
Example:

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

2. Background Image

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

Example:



3. Background Repeat

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

Example:

Html Copy

```
<!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 imag
</body>
</html>
```

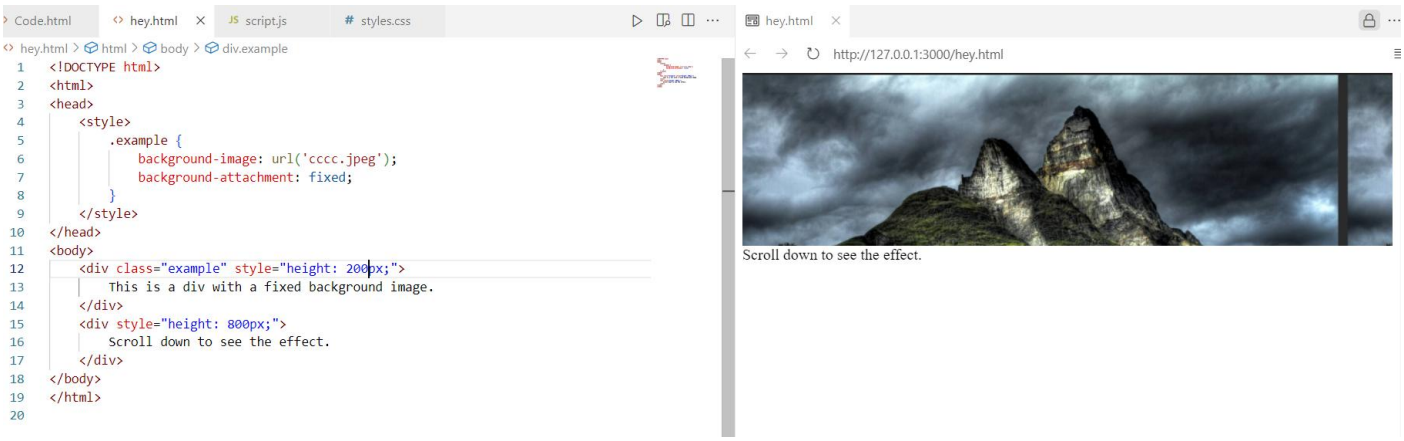
4. Background Position

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

5. Background Attachment

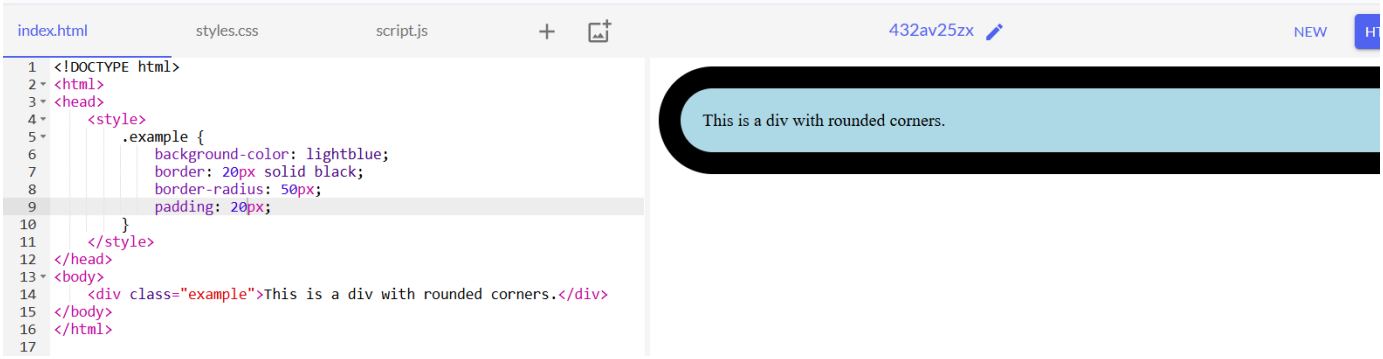
Theory: The `background-attachment` property sets whether a background image is fixed or scrolls with the rest of the page.

Scrolling Effect :



6. Border Radius

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



Box Shadow Theory

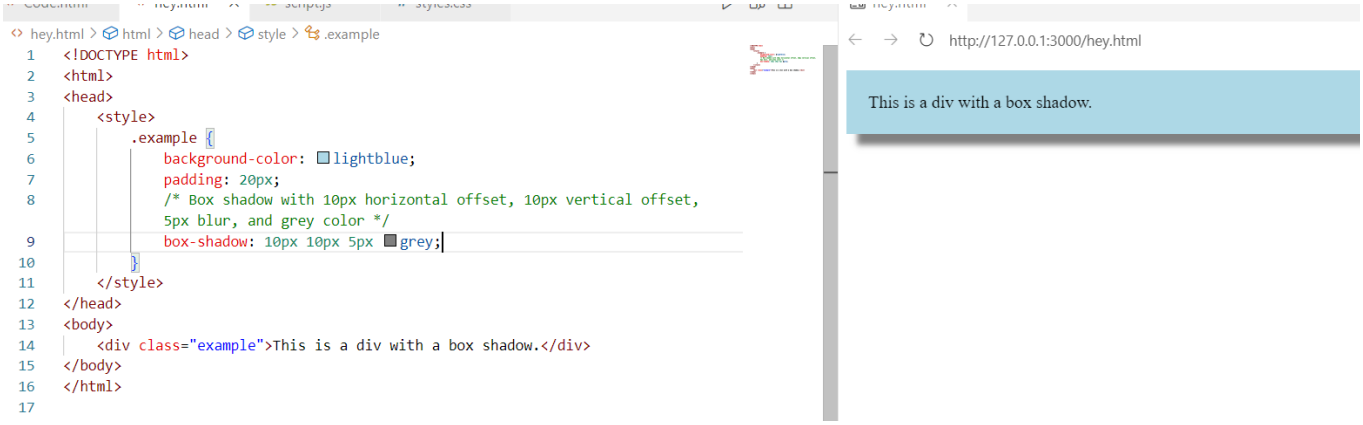
Theory: The `box-shadow` property adds shadow effects around an element's frame. You can specify the shadow's color, size, blur, and spread.

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.



- --> 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:

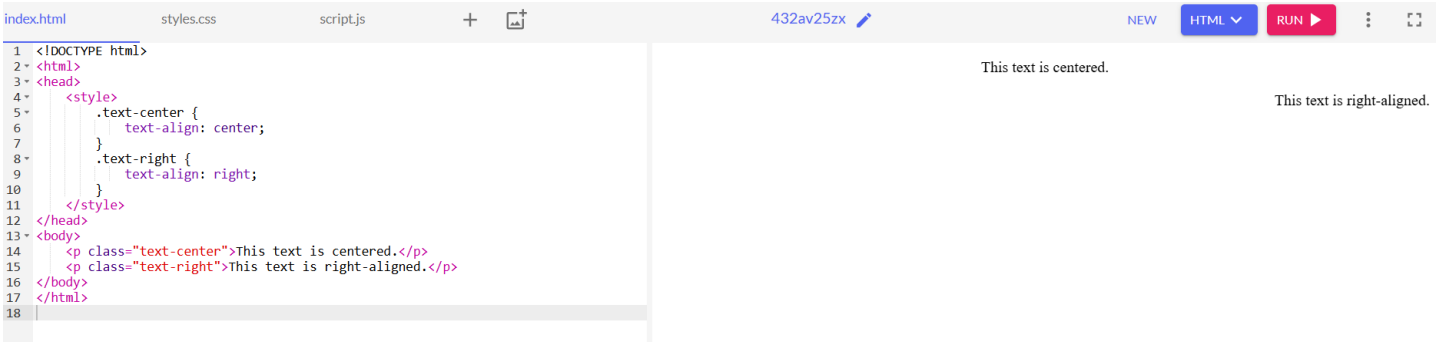
index.html styles.css script.js

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .text-color {
            color: blue;
        }
    </style>
</head>
<body>
    <p class="text-color">This text is blue.</p>
</body>
</html>
```

This text is blue.

2. Text Alignment

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



3. Text Decoration

Properties:

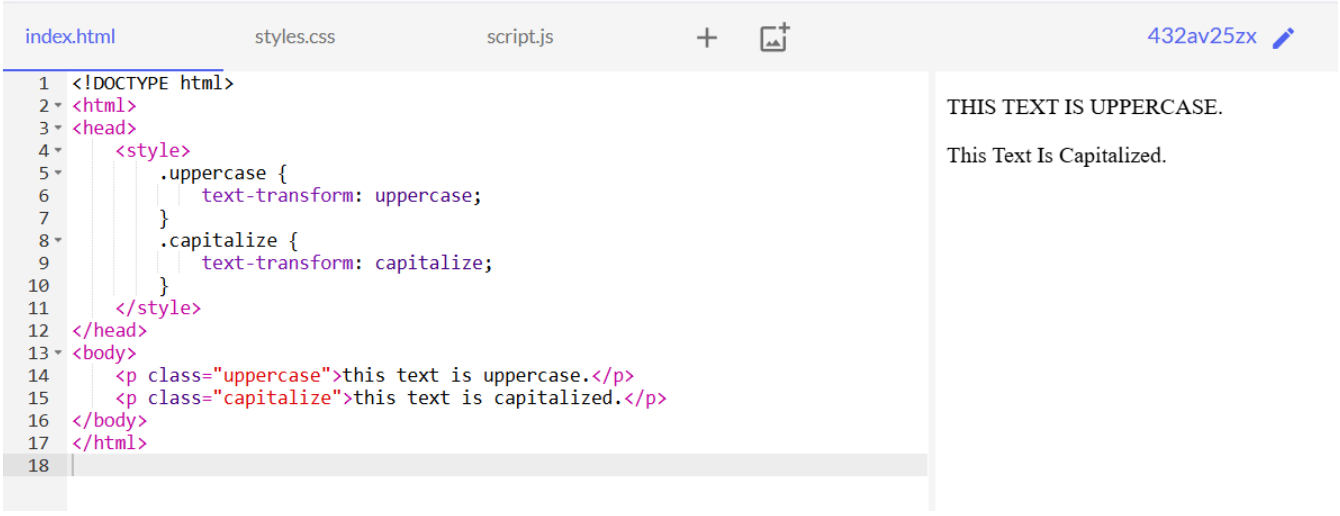
- `text-decoration`: Adds decorations to the text (e.g., `underline`, `overline`, `line-through`, `none`).

```html
<!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`.**

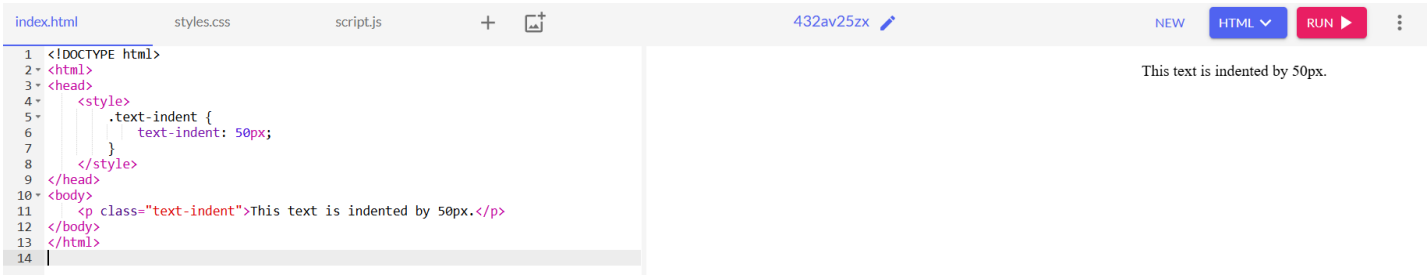


5. Text Indent

Properties:

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

Example:



6. Line Height

Properties:

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

7. Letter Spacing

Properties:

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

index.html styles.css script.js 432av25zx

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .letter-spacing {
            letter-spacing: 2px;
        }
    </style>
</head>
<body>
    <p class="letter-spacing">This text has 2px letter spacing.</p>
</body>
</html>
```

This text has 2px letter spacing.

8. Font Style

Properties:

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

index.html styles.css script.js

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .italic {
            font-style: italic;
        }
        .normal {
            font-style: normal;
        }
    </style>
</head>
<body>
    <p class="italic">This text is italic.</p>
    <p class="normal">This text is normal.</p>
</body>
</html>
```

This text is italic.

This text is normal.

9. Font Size

Properties:

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

index.html styles.css script.js

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .font-small {
            font-size: 12px;
        }
        .font-large {
            font-size: 24px;
        }
    </style>
</head>
<body>
    <p class="font-small">This text is small.</p>
    <p class="font-large">This text is large.</p>
</body>
</html>
```

This text is small.

This text is large.

10. Font Family

Properties:

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

index.html styles.css script.js

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .font-family {
            font-family: 'Arial', sans-serif;
        }
    </style>
</head>
<body>
    <p class="font-family">This text is in Arial font.</p>
</body>
</html>
```

This text is in Arial font.

- --> 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.

2. Type Selector

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

index.html styles.css script.js

```
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
```

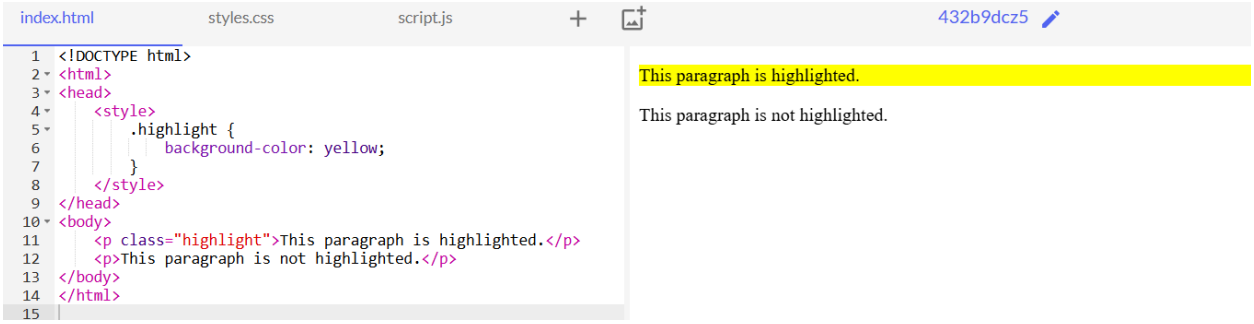
This text is green.

This text is also green.

This text is not green.

3. Class Selector

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

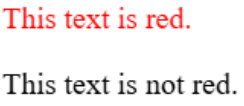


4. ID Selector

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

index.html styles.css script.js

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        #unique {
            color: red;
        }
    </style>
</head>
<body>
    <p id="unique">This text is red.</p>
    <p>This text is not red.</p>
</body>
</html>
```



This text is not red.

5. Attribute Selector

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

index.html styles.css script.js 432bbbfk2

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        [type="text"] {
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <input type="text" value="This has a border.">
    <input type="button" value="This does not have a border.">
</body>
</html>
```

This has a border. This does not have a border.

6. Pseudo-classes

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

index.html styles.css script.js

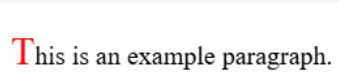
```html
<!DOCTYPE html>
<html>
<head>
    <style>
        a:hover {
            color: orange;
        }
    </style>
</head>
<body>
    <a href="#">Hover over this link.</a>
</body>
</html>
```

7. Pseudo-elements

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

index.html styles.css script.js

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        p::first-letter {
            font-size: 24px;
            color: red;
        }
    </style>
</head>
<body>
    <p>This is an example paragraph.</p>
</body>
</html>
```



8. Child Selector

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

index.html styles.css script.js

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        div > p {
            color: blue;
        }
    </style>
</head>
<body>
    <div>
        <p>This text is blue.</p>
        <span>This text is not blue.</span>
    </div>
    <p>This text is not blue.</p>
</body>
</html>
```

This text is blue.

This text is not blue.

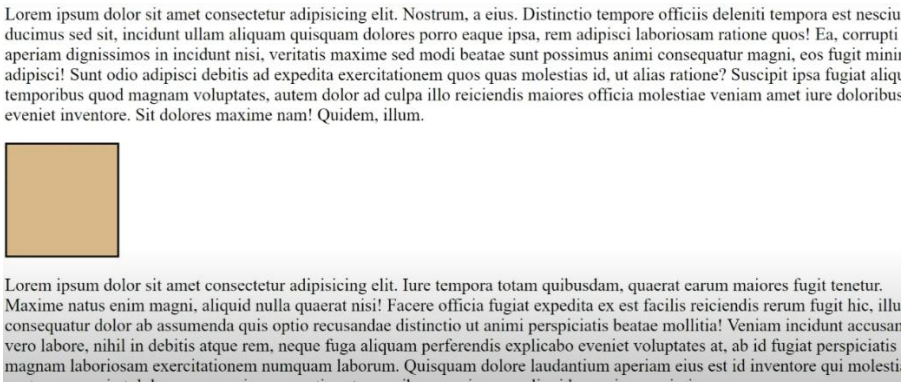
This text is not blue.

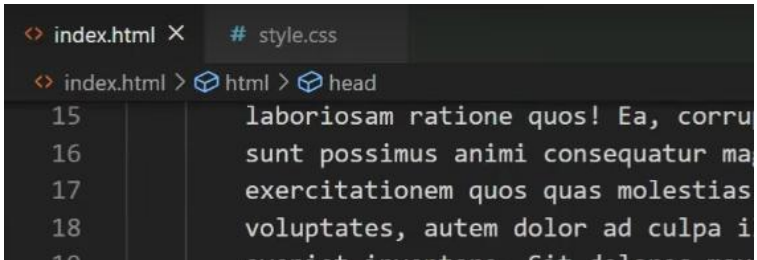
9. Adjacent Sibling Selector

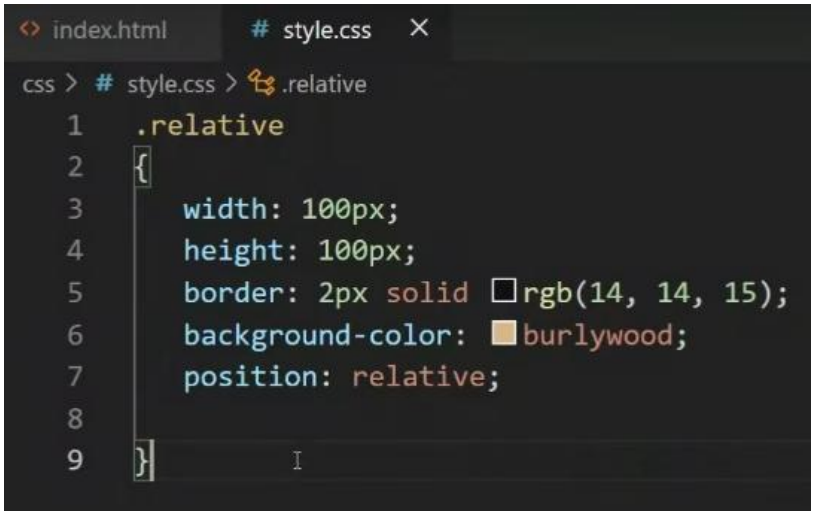
Theory: The adjacent sibling selector (+) selects an element that is immediately preceded by a specified element.

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

1. RELATIVE

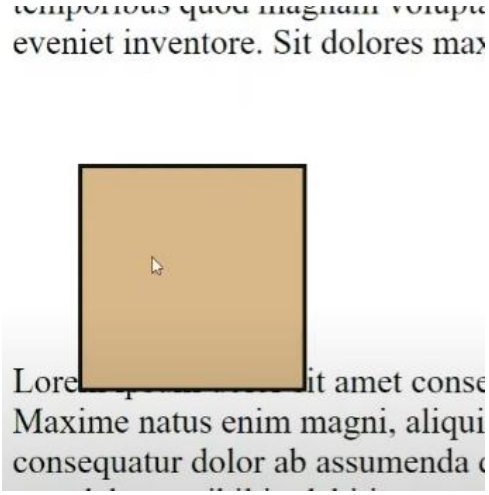






NO : Change for HTML PAGE We Need top ,bottom, left, right

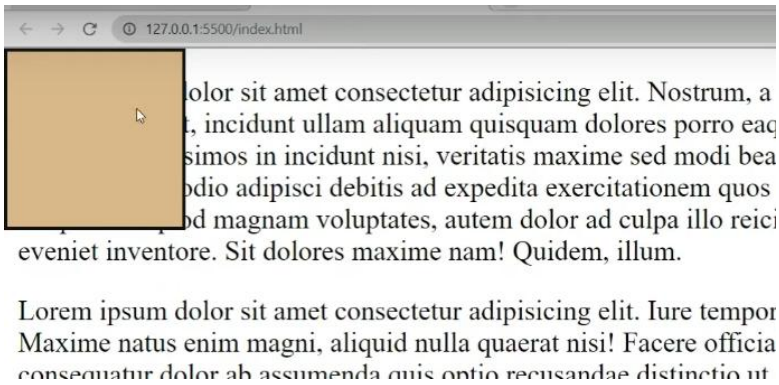
```css
.relative
{
    width: 100px;
    height: 100px;
    border: 2px solid rgb(14, 14, 15);
    background-color: burlywood;
    position: relative;
    top: 30px;
    left: 30px;
}
```



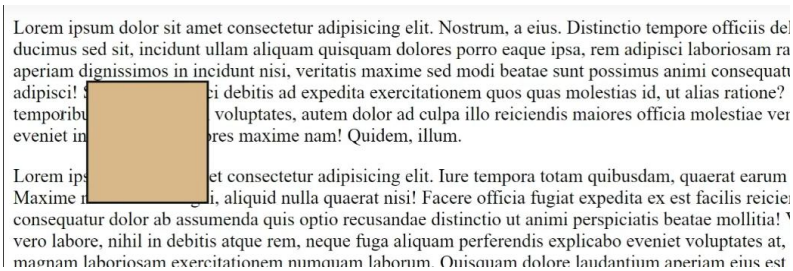
Change for normal position :→

2. **ABSOLUTE:**

```css
.relative
{
    width: 100px;
    height: 100px;
    border: 2px solid rgb(14, 14, 15);
    background-color: burlywood;
    position: absolute;
    top: 0px;
    left: 0px;
}
```



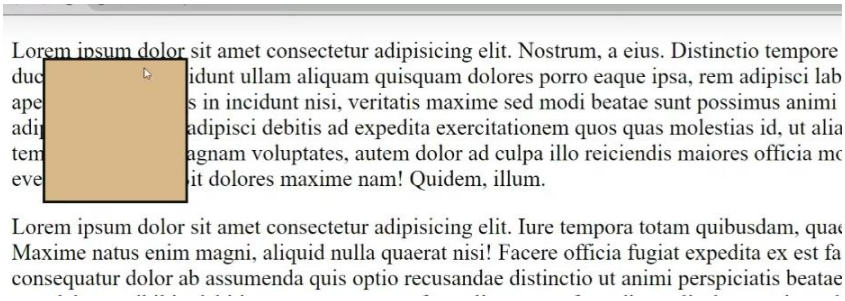
```
css > # style.css > .relative
1   .relative
2   {
3       width: 100px;
4       height: 100px;
5       border: 2px solid rgb(14, 14, 15);
6       background-color: burlywood;
7       position: absolute;
8       top: 70px;
9       left: 70px;
10  }
```



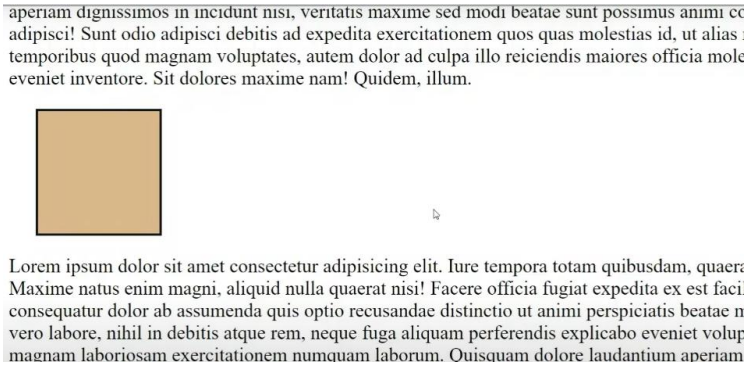
3. FIXIT

```
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

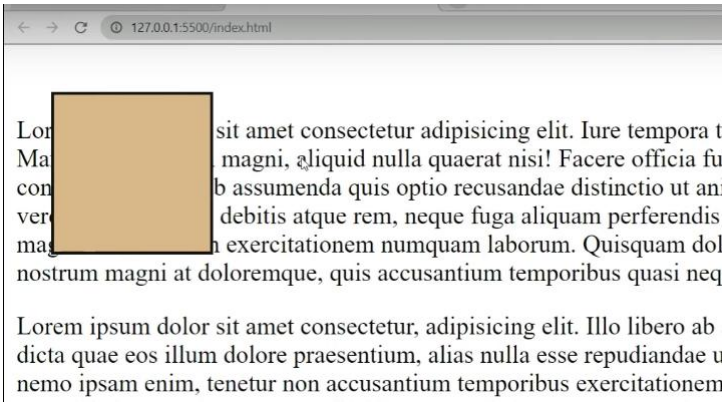


4. Sticky

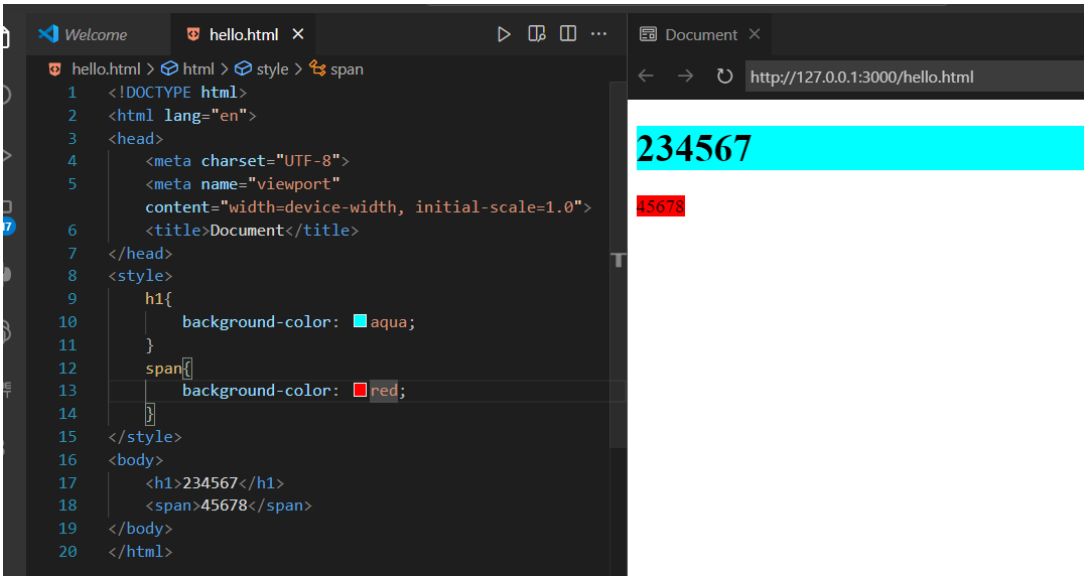


Orginal is

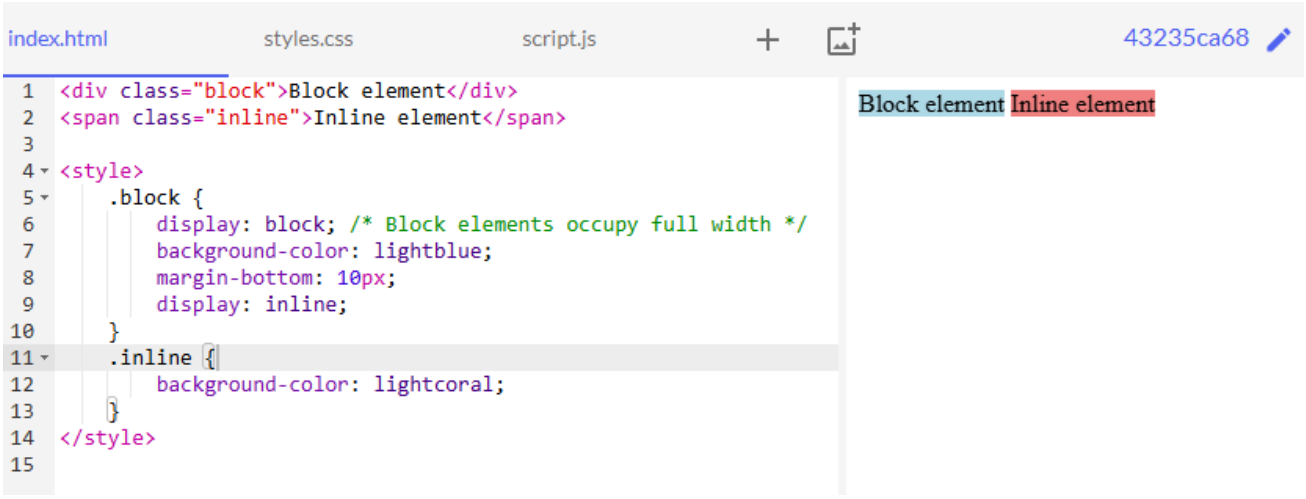
page scrolling but container fix top and left.(automatically space)



- Display



Use display inline in h1



Use span tag full fill color (like div)

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

```
<div class="block">Block element</div>
<span class="inline">Inline element</span>

<style>
    .block {
        background-color: lightblue;
        margin-bottom: 10px;

    }
    .inline {
        background-color: lightcoral;
        display: block;
    }
</style>
```

Block element

Inline element

index.html styles.css script.js 43235ca68

```
<div class="block">Block element</div>
<span class="inline">Inline element</span>

<style>
    .block {
        background-color: lightblue;
        margin-bottom: 10px;
        display: inline;

    }
    .inline {
        background-color: lightcoral;
        display: block;
    }
</style>
```

Block element

Inline element

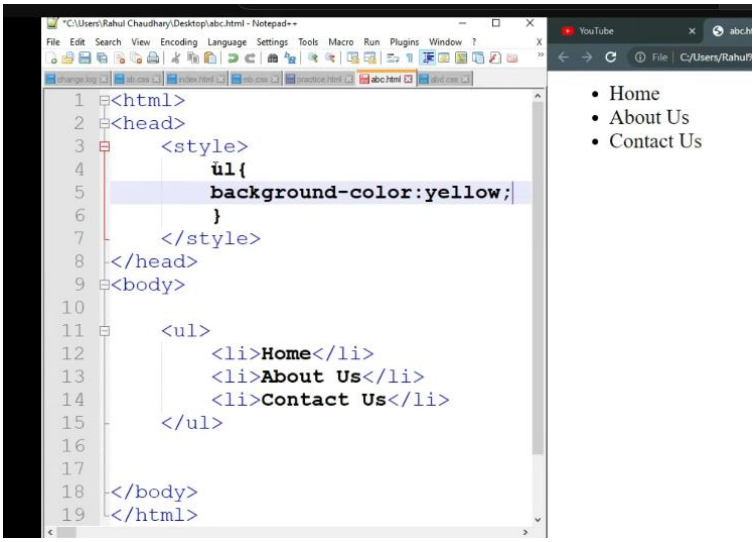
DISPLAY NONE ->

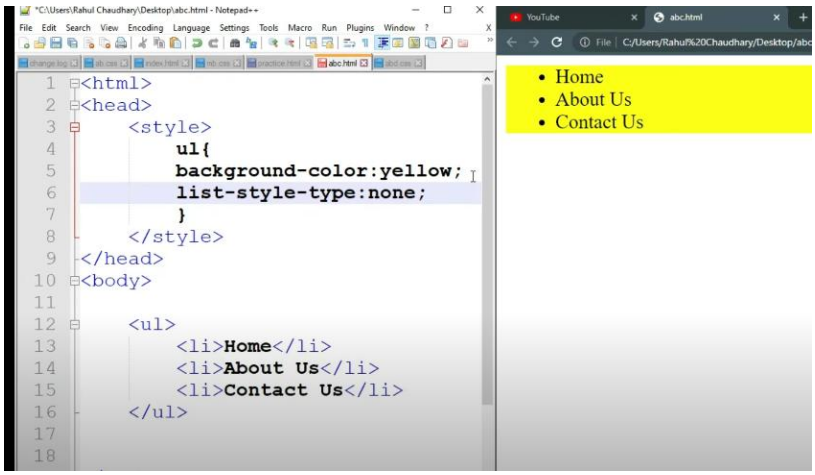
index.html styles.css script.js 43235c

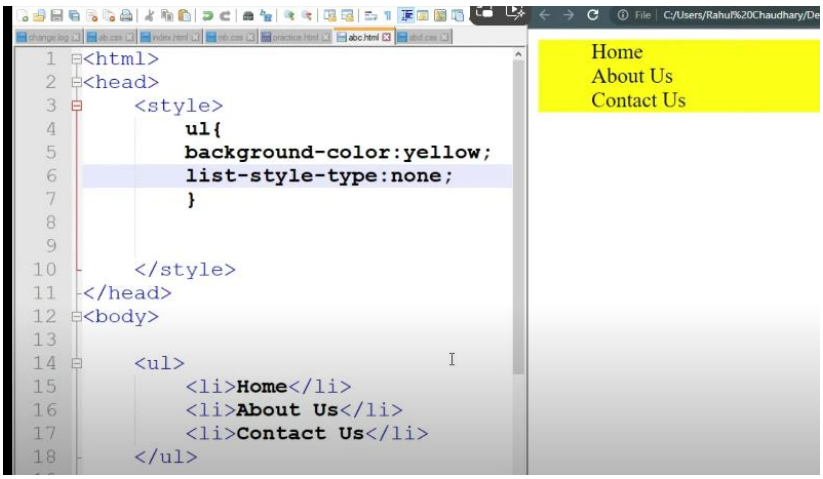
```
<div class="block">Block element</div>
<span class="inline">Inline element</span>

<style>
    .block {
        background-color: lightblue;
        margin-bottom: 10px;
        display: inline;

    }
    .inline {
        background-color: lightcoral;
        /*display: block;*/
        display: none;
    }
</style>
```

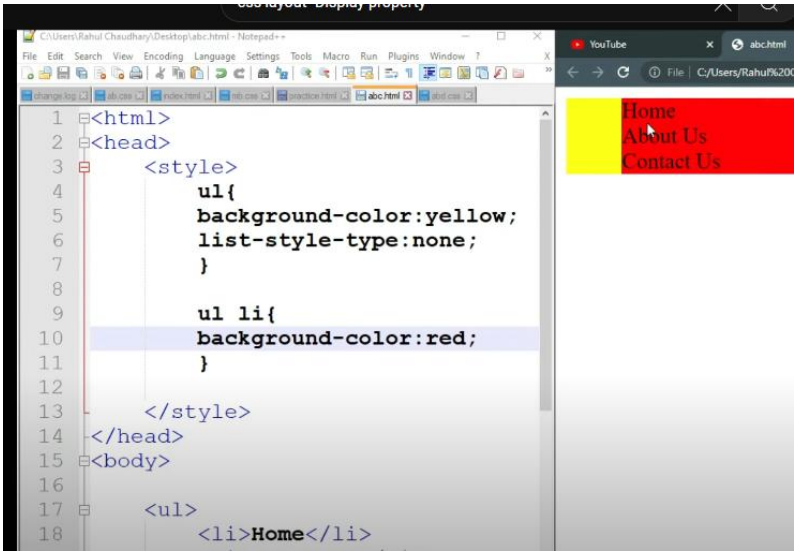
Block element

→

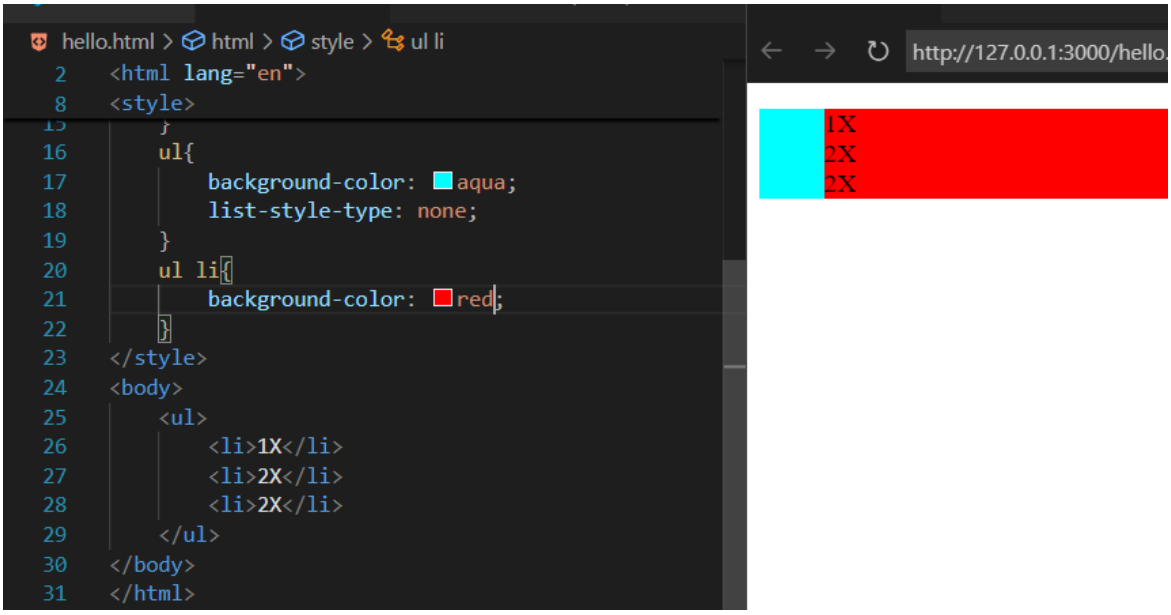

*C:\Users\Rahul Chaudhary\Desktop\abc.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
abc.html
1 <html>
2 <head>
3 <style>
4 ul{
5 background-color:yellow;
6 }
7 </style>
8 </head>
9 <body>
10
11
12 Home
13 About Us
14 Contact Us
15
16
17
18 </body>
19 </html>
YouTube
abc.h
File | C:/Users/Rahul%
• Home
• About Us
• Contact Us

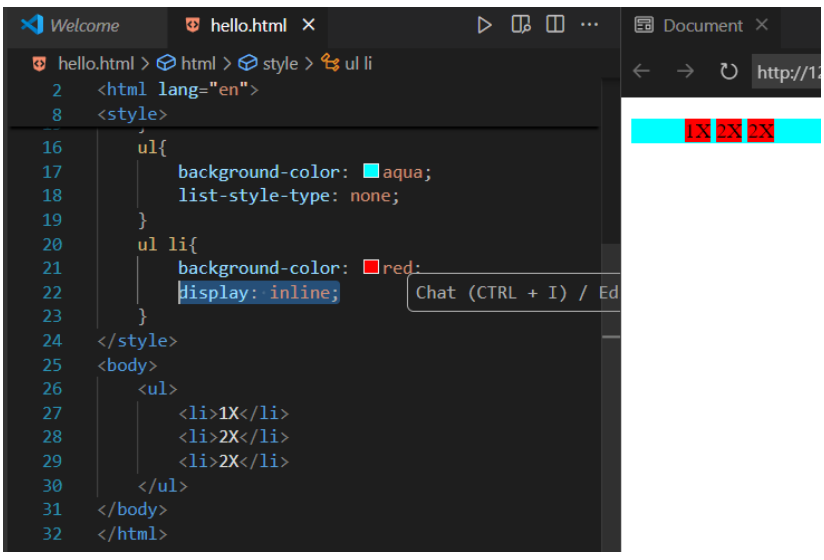

*C:\Users\Rahul Chaudhary\Desktop\abc.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
abc.html
1 <html>
2 <head>
3 <style>
4 ul{
5 background-color:yellow;
6 list-style-type:none;
7 }
8 </style>
9 </head>
10 <body>
11
12
13 Home
14 About Us
15 Contact Us
16
17
18
YouTube
abc.html
File | C:/Users/Rahul%20Chaudhary/Desktop/abc
• 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
15 Home
16 About Us
17 Contact Us
18
File | C:/Users/Rahul%20Chaudhary/De
Home
About Us
Contact Us

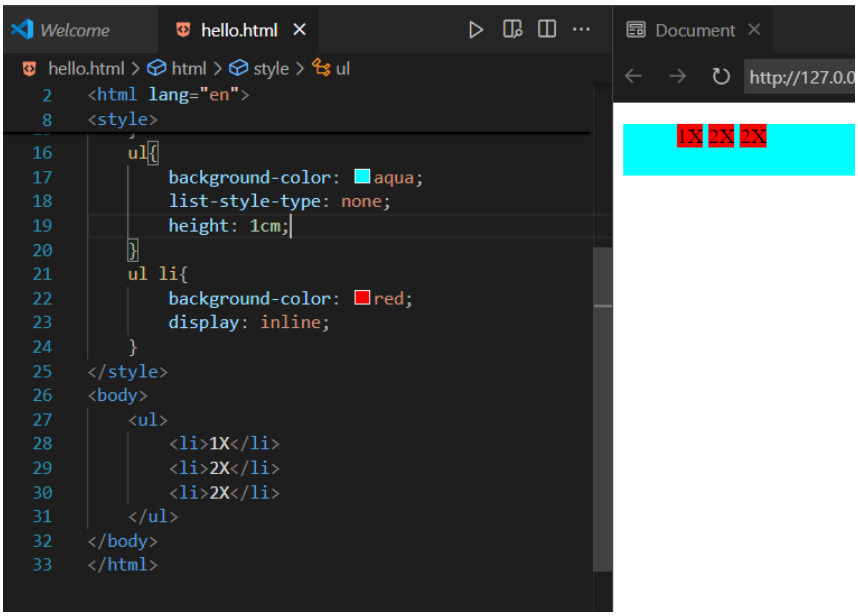


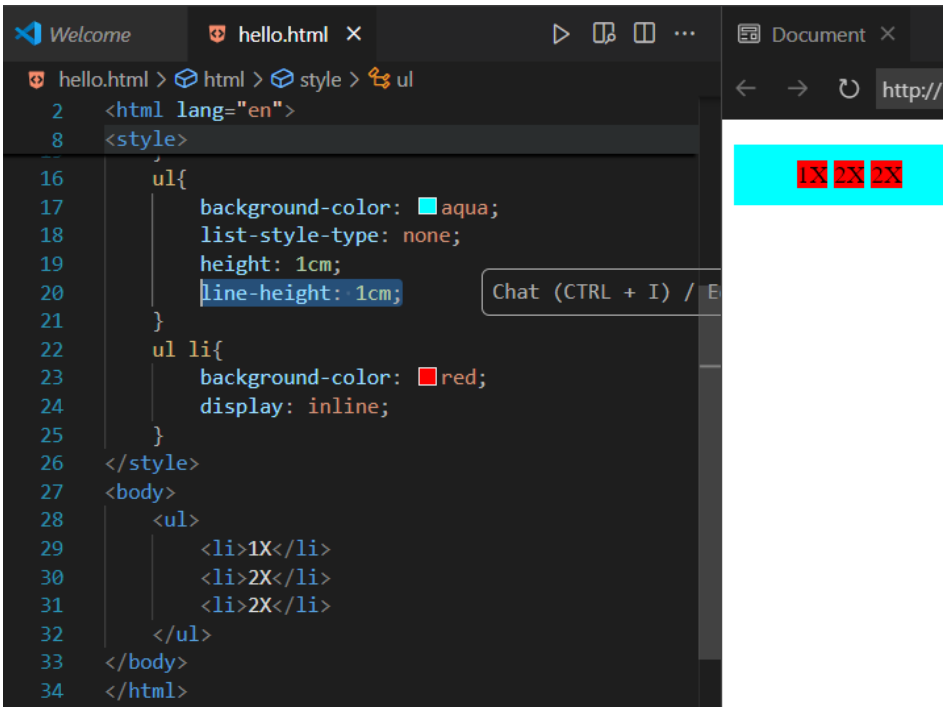
Ul->li Color change



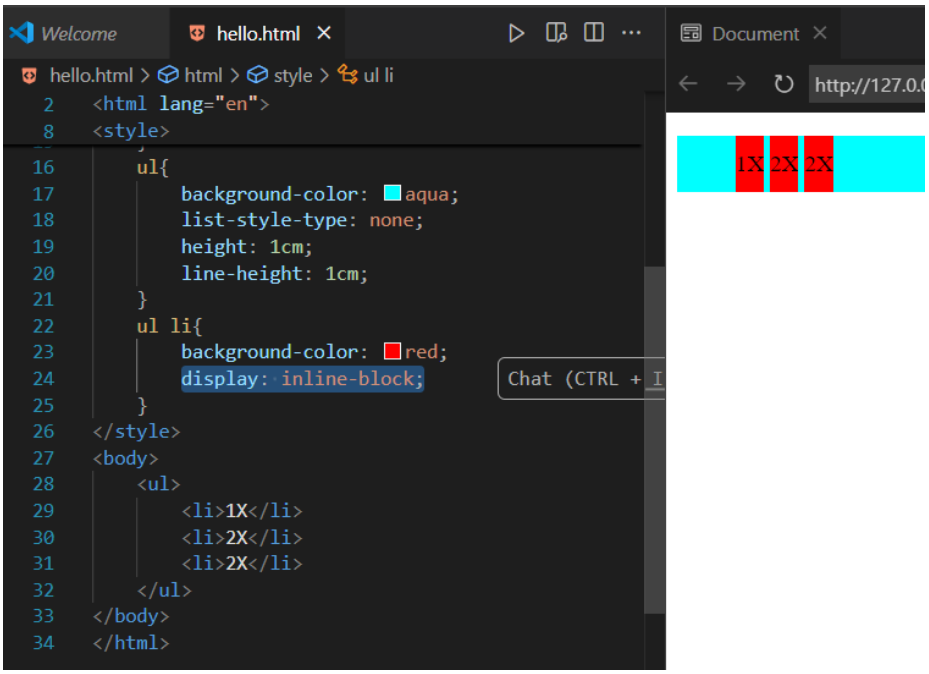


Height & Line height ->

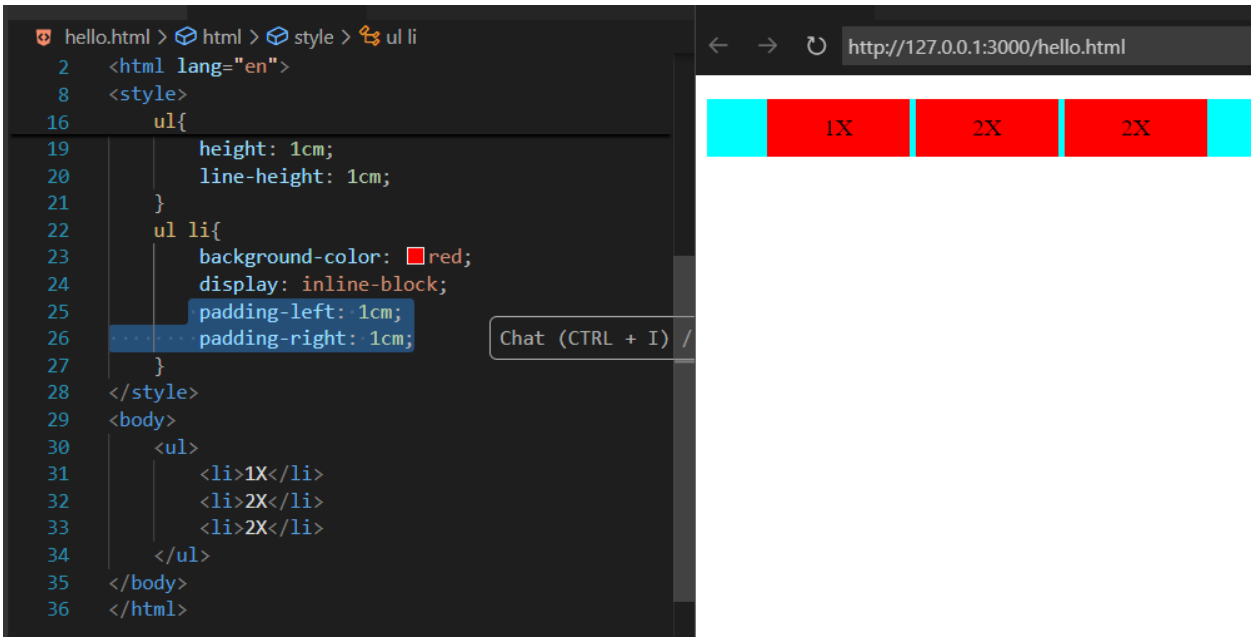




Use inline block ->



Add padding left& Right



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

```
<span class="inline">Inline element</span>
<div class="inline1">Inline element</div>
<div class="inline-block">Inline-block element</div>
<style>
    .inline {
        background-color: lightcoral;
    }
    .inline1 {
      display: inline-block;
        background-color: lightcoral;
    }
    .inline-block {
        display: inline-block;
        width: 200px;
        height: 50px;
        background-color: lightgreen;
    }
</style>
```

Inline element Inline element Inline-block element

Display flex->

index.html styles.css script.js 43235ca68

```
<div class="flex-container">
    <div class="flex-item">Item 1</div>
    <div class="flex-item">Item 2</div>
    <div class="flex-item">Item 3</div>
</div>

<style>
    .flex-container {

        background-color: lightgrey;
    }

    .flex-item {
        background-color: lightblue;
        padding: 20px;
        margin: 10px;
    }
</style>

```

Item 1

Item 2

Item 3

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

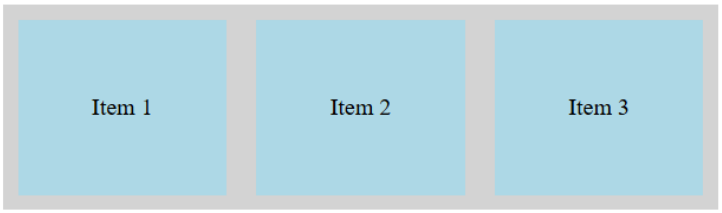
```
<div class="flex-container">
    <div class="flex-item">Item 1</div>
    <div class="flex-item">Item 2</div>
    <div class="flex-item">Item 3</div>
</div>

<style>
    .flex-container {
        display: flex;
        background-color: lightgrey;
    }

    .flex-item {
        background-color: lightblue;
        padding: 20px;
        margin: 10px;
    }
</style>

```

Item 1 Item 2 Item 3

Inline display flex

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

```
<div class="inline-flex-container">
    <div class="flex-item">Item 1</div>
    <div class="flex-item">Item 2</div>
    <div class="flex-item">Item 3</div>
</div>

<style>
    .inline-flex-container {
        display: inline-flex;
        background-color: lightgrey;
    }

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



FLOAT →

index.html styles.css script.js 4324tnfh3

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Float Example</title>
    <style>
        .float-left {
            float: left;
            margin: 100px;
        }
    </style>
</head>
<body>
    <img src="example.jpg" alt="Example Image" class="float-left">
    <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus lacin
        Cras venenatis euismod malesuada. Nullam ac erat ante. Phasellus digni
        bibendum ac. Fusce tincidunt, ligula et facilisis facilisis, purus sap
        id lectus.
    </p>
</body>
</html>
```

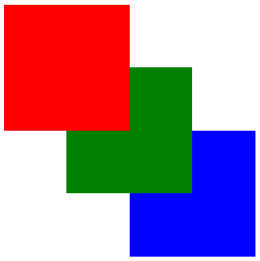
Example Image

Lorem ipsum dolor sit amet, con
vitae vestibulum vestibulum. Cra
ante. Phasellus dignissim sollicit
tincidunt, ligula et facilisis facili
risus dui id lectus.

Z-indexing

index.html styles.css script.js 4324tnfh3

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Z-Index Example</title>
    <style>
        .box {
            width: 100px;
            height: 100px;
            position: absolute;
        }
        .box1 {
            background-color: red;
            left: 50px;
            top: 50px;
            z-index: 3;  /* Changed from 1 to 3 */
        }
        .box2 {
            background-color: green;
            left: 100px;
            top: 100px;
            z-index: 2;
        }
        .box3 {
            background-color: blue;
            left: 150px;
            top: 150px;
            z-index: 1;

        }
    </style>
</head>
<body>
    <div class="box box1"></div>
    <div class="box box2"></div>
    <div class="box box3"></div>
</body>
</html>

```



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.

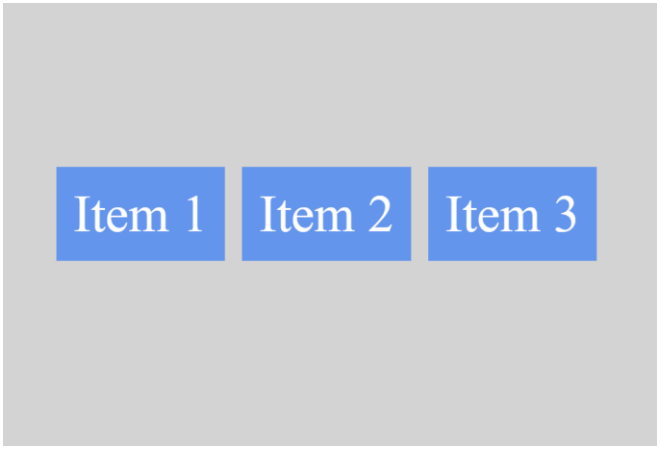
2. Responsive Design

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

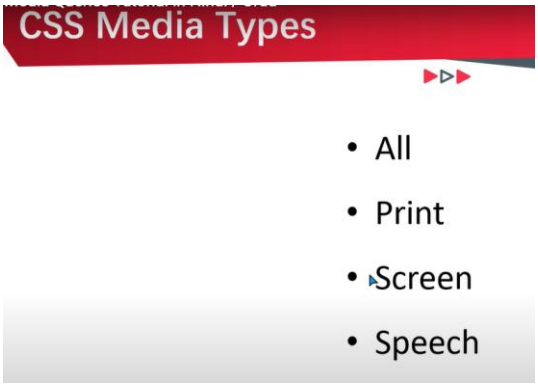
One of the most powerful features of Flexbox is its ability to align items both horizontally and vertically with ease. Properties like `justify-content` and `align-items` provide simple solutions for centering elements.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Basic Flexbox Layout</title>
    <style>
        .container {
            display: flex; /* Establishes a flex container */
            justify-content: center; /* Centers items horizontally */
            align-items: center; /* Centers items vertically */
            height: 100vh; /* Full viewport height */
            background-color: lightgray; /* Background color for the container */
        }
        .item {
            background-color: cornflowerblue; /* Background color for items */
            padding: 20px; /* Padding inside items */
            margin: 10px; /* Margin around items */
            color: white; /* Text color */
            font-size: 3.8rem; /* Font size */
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
    </div>
</body>
</html>

```

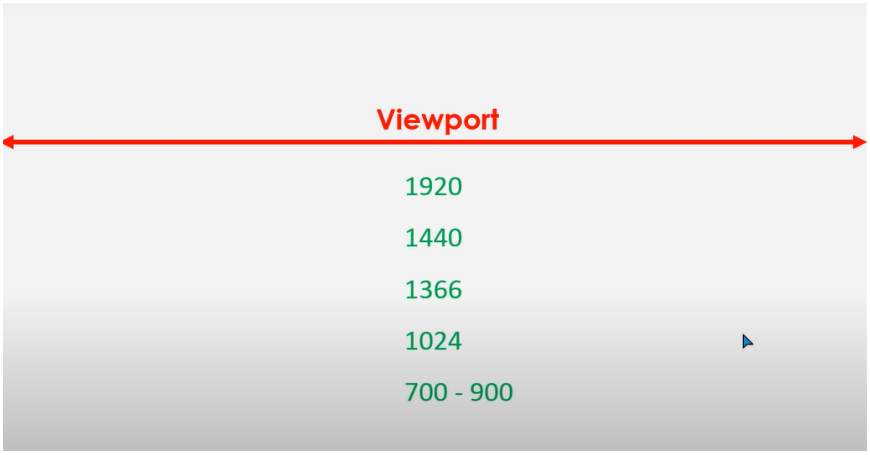


- --> Responsive Design
- Media Queries
- Viewport
- Responsive Units (vw, vh, %, etc.)

Media Queries



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) {

}
```


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

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

```html
<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>
```

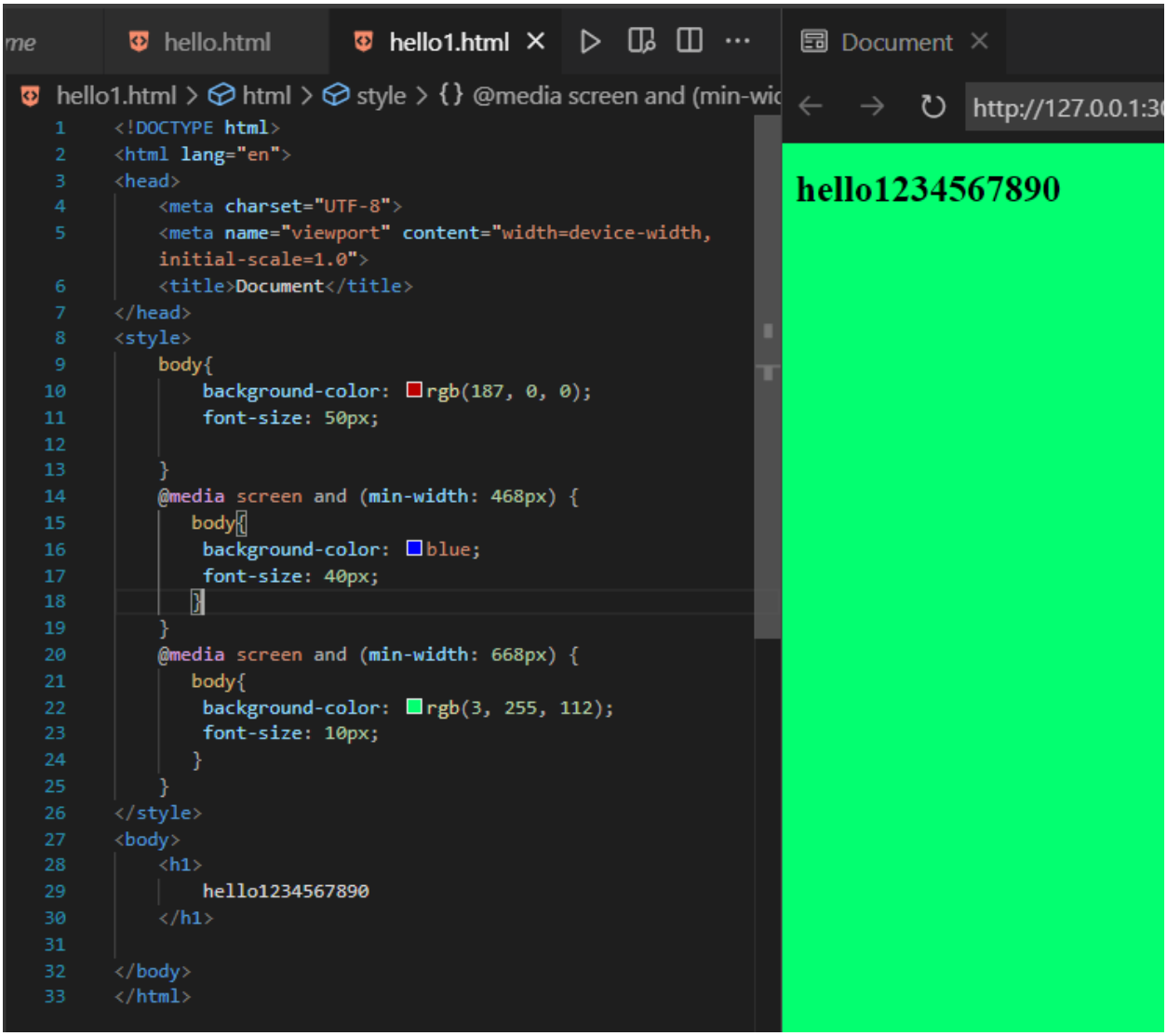
Multiple Condition apply for

```html
<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>
```

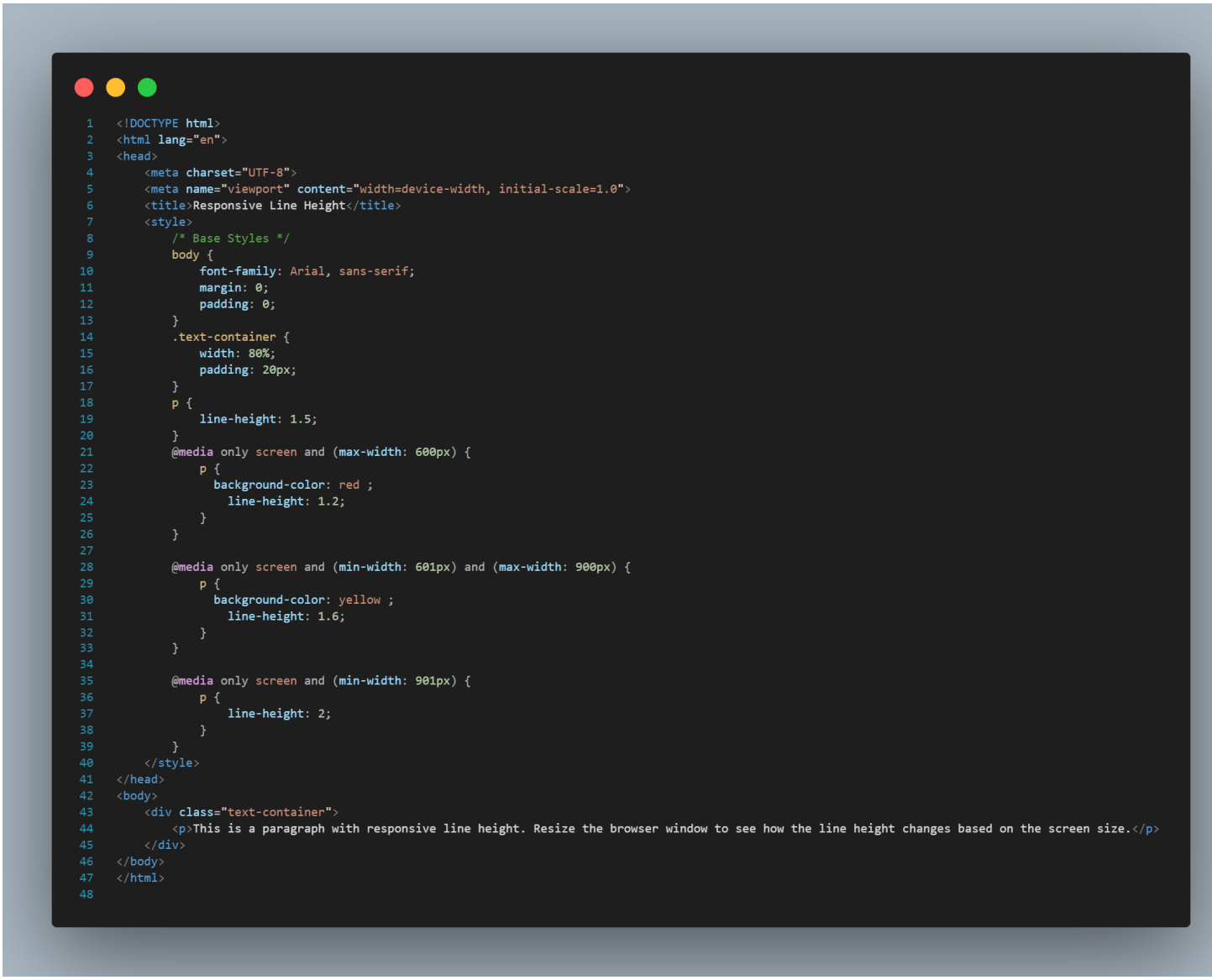
ADD Font Size →

```
<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>
```


hello.html
hello1.html
hello1.html > html > style > {} @media screen and (min-wid
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>Document</title>
7 </head>
8 <style>
9 body{
10 background-color: rgb(187, 0, 0);
11 font-size: 50px;
12
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>
Document
http://127.0.0.1:3
hello1234567890

O/p ?



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Line Height</title>
    <style>
        /* Base Styles */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        .text-container {
            width: 80%;
            padding: 20px;
        }
        p {
            line-height: 1.5;
        }
        @media only screen and (max-width: 600px) {
            p {
              background-color: red ;
                line-height: 1.2;
            }
        }

        @media only screen and (min-width: 601px) and (max-width: 900px) {
            p {
              background-color: yellow ;
                line-height: 1.6;
            }
        }

        @media only screen and (min-width: 901px) {
            p {
                line-height: 2;
            }
        }
    </style>
</head>
<body>
    <div class="text-container">
        <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>
    </div>
</body>
</html>

```

- # 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

```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

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Viewport Units Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }

        .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;
            display: flex;
            align-items: center;
            justify-content: center;
        }
    </style>
</head>
<body>
    <div class="box">
        <p>This box uses vw, vh, vmin, and vmax units to adapt to the viewport size.</p>
    </div>
</body>
</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	`transition` property use karte hain	`animation` property aur `@keyframes` 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

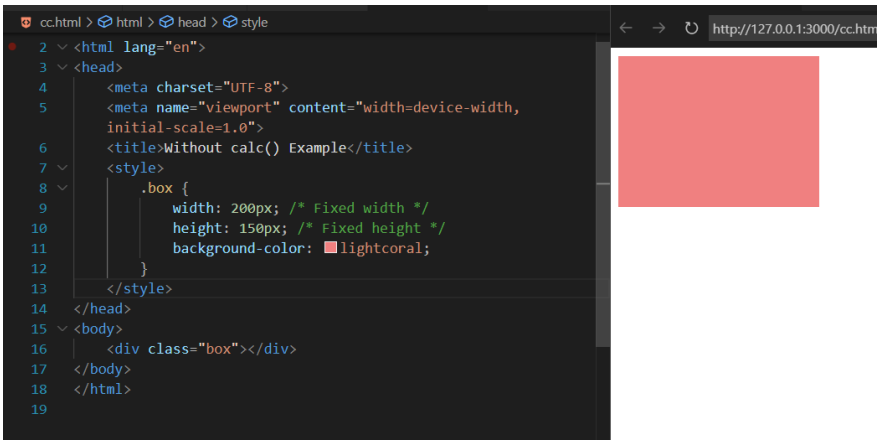
Syntax	`transition` property use karte hain	`animation` property aur `@keyframes` use karte hain
Iteration	Typically ek baar run karte hain per state change	Infinite loop ya specific number of times run ho sakte hain
Complexity	Basic effects ke liye best (fading, scaling, position changes)	Intricate 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()

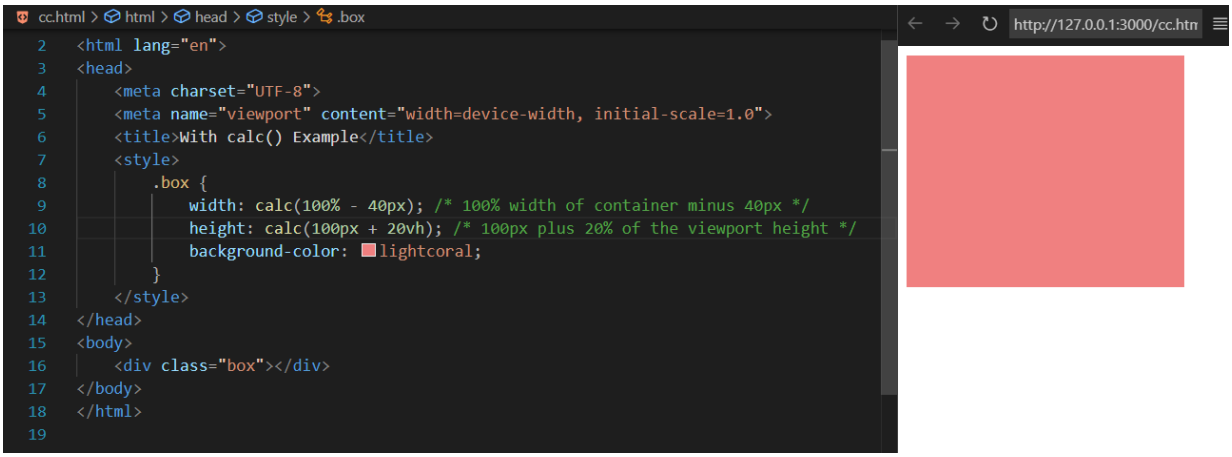
→Calc ()

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.

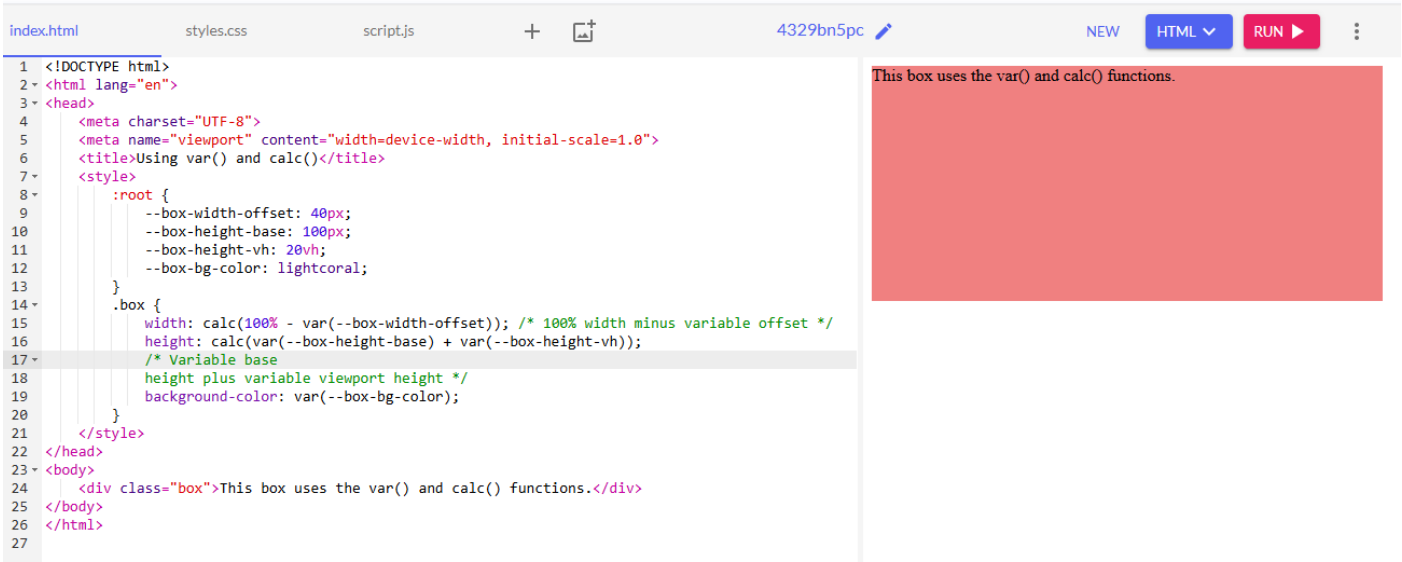
Without calc()-> page



With CALC()->



Var Example :



Another example

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

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Using var()</title>
    <style>
        :root {
            --main-color: lightblue;
            --padding-size: 20px;
        }

        .var-box {
            background-color: var(--main-color);
            padding: var(--padding-size);
            width: calc(100% - 40px);
            height: calc(50px + 15vh);
        }
    </style>
</head>
<body>
    <div class="var-box">
        This box uses the var() function to apply CSS variables.
    </div>
</body>
</html>
```

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

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Using url()</title>
    <style>
        .url-box {
            width: 100%;
            height: 300px;
            background-image: url('https://th.bing.com/th/id/R.
            ae3d2f4880eaca12be7c84e2d2c56041?rik=KeMl0VTd5YUt2g&
            riu=http%3a%2f%2fgetwallpapers.
            com%2fwallpaper%2ffull%2fb%2f1%2f9%2f355093.jpg&
            ehk=oGlwjQgO%2fFGGKmJsbxLKvu%2bk2XY18CIw6bAAeo6QVPw%3d&risl=&
            pid=ImgRaw&r=0');
            background-size: cover;
            background-position: center;
        }
    </style>
</head>
<body>
    <div class="url-box">
        This box uses the url() function to set a background image.
    </div>
</body>
</html>
```

http://127.0.0.1:3000/url_css_.html

This box uses the url() function to set a background image.

CSS.

The BOX is use Click:

index.html styles.css script.js 4329e5xwd NEW HTML RUN

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Open Google</title>
    <style>
        .link-box {
            width: 100%;
            height: 300px;
            background-image:url(https://th.bing.com
            /th/id/R.ae3d2f4880eaca12be7c84e2d2c56041
            ?rik=KeMl0VTdSYUt2g&riu=http%3a%2f%2fgetwallpape
            rs.com%2fwallpaper%2ffull%2fb%2f1%2f9%2f355093.
            jpg&ehk=oGlwjQgO%2fFGGKmJsbxLKvu%2bk2XY18CIw6bA
            Aeo6QVPw%3d&risl=&pid=ImgRaw&r=0);
            display: flex;
            justify-content: center;
            align-items: center;
            text-decoration: none;
            color: white;
            font-size: 2rem;
        }
    </style>
</head>
<body>
    <a href="https://www.google.com" target="_blank" class="link-box">
        Click Here to Open Google
    </a>
</body>
</html>

```

Feature	`calc()`	`var()`	`url()`
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	`calc(expression)` (e.g., `calc(100% - 50px)`)	`var(--custom-property, fallback)` (e.g., `var(--main-color, blue)`)	`url(path-to-resource)` (e.g., `url('image.jpg')`)
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 `--custom-property`. Provides fallback if the variable is unset.	Simply resolves the URL to fetch or load the resource.
Example	`width: calc(100% - 50px);`	`color: var(--text-color, black);`	`background-image: url('background.jpg');`

• 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.

index.html styles.css script.js 432bbbfk2 NEW

```html
<!DOCTYPE html>
<html>
<head>
    <title>Simple Task</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>Welcome to My Simple Page</h1>
    <p>This is a simple paragraph to demonstrate HTML and CSS styling.</p>
    <button>Click Me!</button>
</body>
</html>
```

index.html styles.css script.js 432bbbfk2

```css
/* Style for the header */
h1 {
    color: blue;
}

/* Style for the paragraph */
p {
    font-size: 18px;
    margin: 20px 0;
}

/* Style for the button */
button {
    background-color: coral;
    padding: 10px 20px;
    border: none;
    color: white;
    cursor: pointer;
}
```