Ways to add css :

Inline css : Inline CSS is used to add custom properties to specific elements. The added style will only reflect on that particular element only.To use inline CSS, insert the `style` attribute within the HTML element's opening tag.

Consider the code snippet:

```
<h1 style="color: purple;">I'm harry</h1><h2>I'm CodeWithHarry</h2>
```

Internal CSS

Internal CSS is used to apply custom style to multiple elements on the same page. The style can be used throughout the HTML page.

Internal CSS is defined in a **style block**, which will be inside the **head section**.

External CSS

External CSS works similarly to internal CSS but with a twist. Instead of adding the styles within the HTML file, we create a separate file with a **.css** extension. This file will hold all the styling details. Then, we link this file to the HTML page, giving it the instructions on how to look.

What are CSS Selectors?

CSS selectors allow us to choose specific elements and apply styles to them. Suppose we want to add a custom style to only a specific tag(s). There, we can make use of CSS selectors.

There are different types of CSS selectors, which are as follows:

1. Universal Selector

2. Element Selector
3. Id Selector
4. Class Selector
5. Group Selector

Universal Selector

Universal selector represented by * targets all the HTML elements on the page.

The syntax of Universal Selector is as follows:

```
* {

   property: value;

}
```

```
<html><head> <style> * { color: purple; text-align: center; }
</style></head><body> <p>Welcome to </p> <h1>CodeWithsri</h1></body></html>
```

Element Selector (Type Selector)

The element selector selects the target element based on the specific type. Suppose you want to underline all the <p> tags; in this case, the element selector will be the best choice.

The syntax of Element Selector is as follows:

```
p {    property: value;}

<html>

<head>
```

```
    <title>CSS</title>

    <style>

        p {

            text-decoration: underline;

        }

    </style>

</head>

<body>

    <h1>CodeWithsri</h1>

    <h2>we offer: </h2>

    <p>Python Tutorials - 100 Days of Code</p>

    <p>Ultimate JavaScript Course</p>

    <p>React JS Tutorials For Beginners</p>

</body>

</html>
```

ID Selector

The ID selector targets the elements based on the specific ID. It is written with the hash # character followed by the ID name in the style sheet.

The syntax of ID Selector is as follows:

```
#ID {    property: value;}
```

```
<html>
<head>
    <style>
        #title {
            text-align: center;
            color: red;
        }
    </style>
</head>
<body>
    <h1 id="title">CodeWithsri</h1>
    <p>I'm a Developer and the founder of CodeWithHarry.com</p>
</body>
</html>
```

Class Selector

The class selector does the same job as the id selector, a class selector helps group various types of elements. Suppose, we want to give a custom style to a specific group of elements. In this case, the class selector is the best option.

It is written with the period `.` character followed by the class name in the style sheet.

The syntax of Class Selector is as follows:

```
.class {    property: value;}
<html>
<head>
    <title>CSS</title>
    <style>
        .red {
            color: red;
        }
    </style>
</head>
<body>
    <p>This is simple p tag</p>
    <p class="red">This p tag has class red</p>
    <p>This is simple p tag</p>
    <p class="red">This p tag has class red</p>
</body>
</html>
```

Group Selector

The group selector is used to minimize the code. Commas , are used to separate each selector in a grouping. This reduces the number of lines of code. The code also looks clean.

The syntax of Group Selector is as follows:

```
div, p, a {    property: value;}
<html>
<head>
    <title>CSS</title>
    <style>
        h1 {
            color: red;
        }
        p, a {
            color: purple;
        }
    </style>
</head>
<body>
    <h1>CodeWithHarry</h1>
    <p>This is the p tag</p>
    <a href="#">This is the anchor (a) tag</a>
</body>
</html>
```

- Universal Selector (*): Target the entire page.
- Element Selector: Target a specific element.
- ID Selector (#): Target element with a specific ID.
- Class Selector (.): Target element(s) with the same class.
- Group Selector: Group elements and target them.

CSS Box Model

The CSS Box model defines how elements are rendered and how their dimensions are calculated. It describes the structure of an element as a rectangular box that has content, padding, a border, and a margin. The box model consists of four main components, which are:

1. Content

- The innermost component of the box model is the actual content of the element. It can be text, image, video, etc.
- The content area is defined by the **width** and **height** properties.

2. Padding

- The space between the actual content and the border of the element is the padding.
- The padding area is defined by the property **padding**. For more details, follow the CSS Padding tutorial.

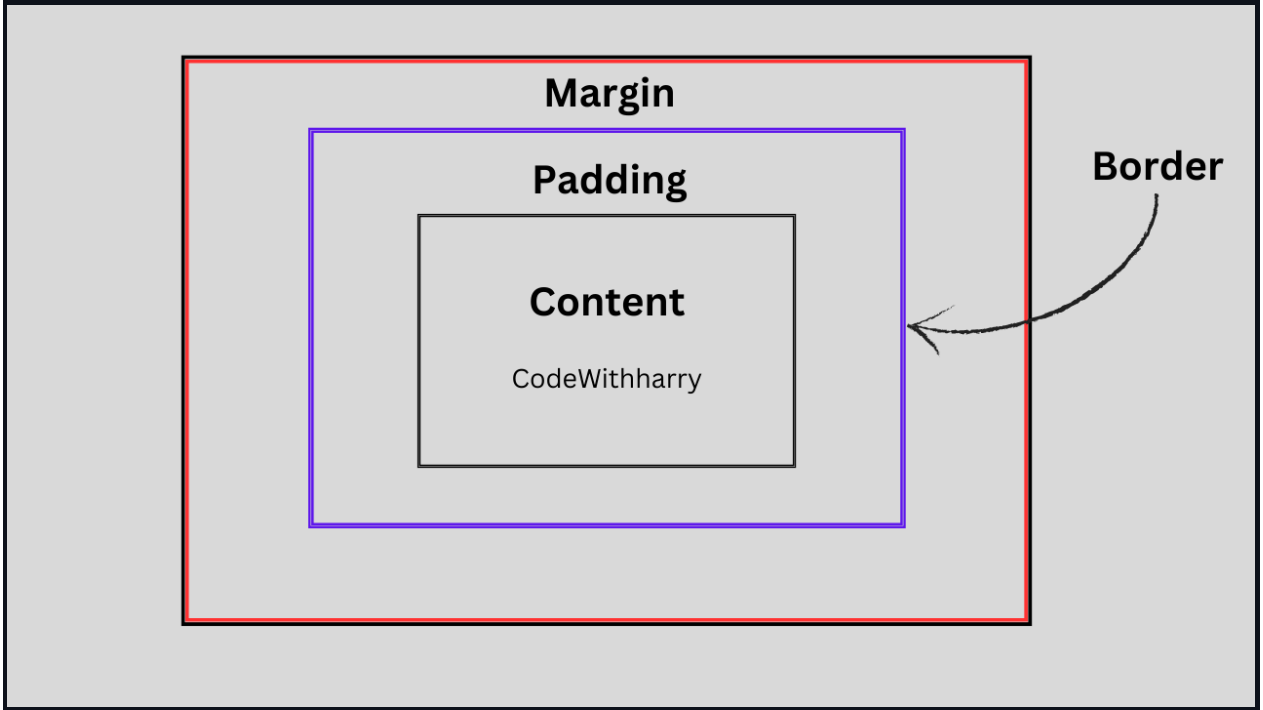
3. Border

- The border surrounds the content and padding and gives the visual border of the element.
- The border properties can be controlled using the **border** keyword. For more details, follow the CSS Borders tutorial.

4. Margin

- The margin is the space outside the element that separates it from other elements in the layout.
- The margin of the element is controlled by the **margin** property. For more details, follow the CSS Margin tutorial.

The CSS Box model can be visually represented as:



CSS Text Styling

In this tutorial, we will cover some of the important text styling properties:

Text Decoration

The `text-decoration` property adds various types of text decorations.

Syntax:

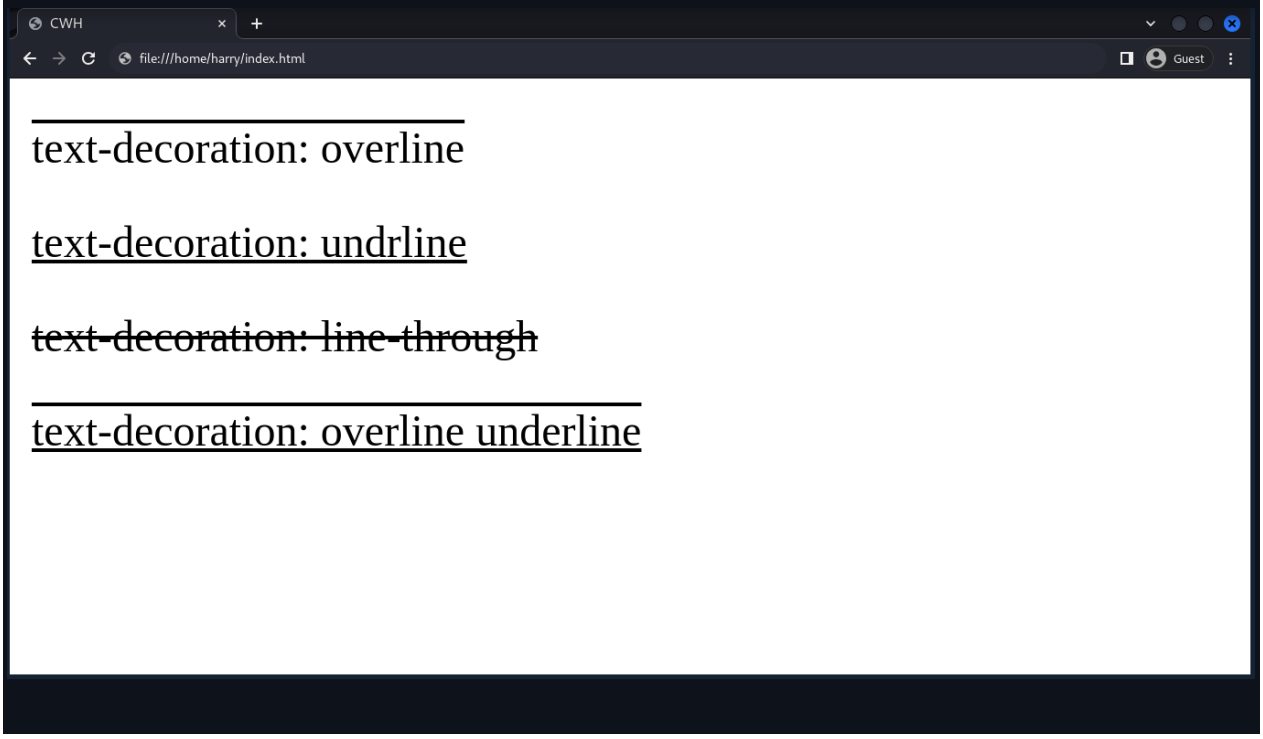
```
selector {    text-decoration: value;}
```

There are four values for `text-decoration`:

- **overline**: adds a line above the text
- **underline**: adds a line below the text
- **line-through**: adds a line through the text.
- **none**: to remove decoration.

Example:

```
<html lang="en"><head>    <style>        #p1 {text-decoration: overline;}        #p2 {text-decoration: underline;}        #p3 {text-decoration: line-through;}        #p4 {text-decoration: overline underline;}    </style></head><body>    <p id="p1">text-decoration: overline</p>    <p id="p2">text-decoration: underline</p>    <p id="p3">text-decoration: line-through</p>    <p id="p4">text-decoration: overline underline</p></body></html>
```



Text Alignment

The `text-align` property is used to align the text within the container. The container can be a div, section, etc.

Syntax:

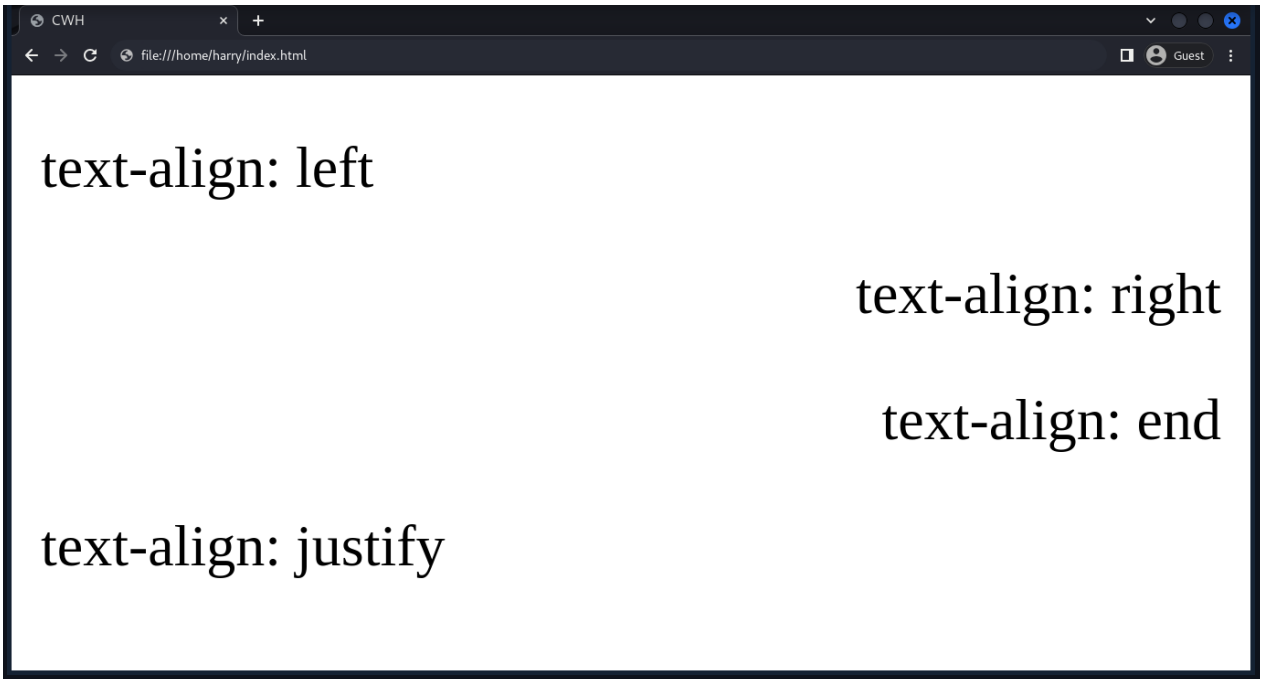
```
selector {    text-align: value;}
```

There are four values for `text-align`:

- **left**: align the text to the left.
- **center**: align the text to the center.
- **right**: align the text to the right.
- **justify**: spread the text evenly between the left and right margins.

Example:

```
<html lang="en"><head>    <style>        #p1 {text-align: left;}
#p2 {text-align: right;}        #p3 {text-align: end;}        #p4
{text-align: justify;}    </style></head><body>    <p id="p1">text-
align: left</p>    <p id="p2">text-align: right</p>    <p
id="p3">text-align: end</p>    <p id="p4">text-align:
justify</p></body></html>
```



Text Transformation

It transforms the text case.

Syntax:

```
selector {    text-transform: value;}
```

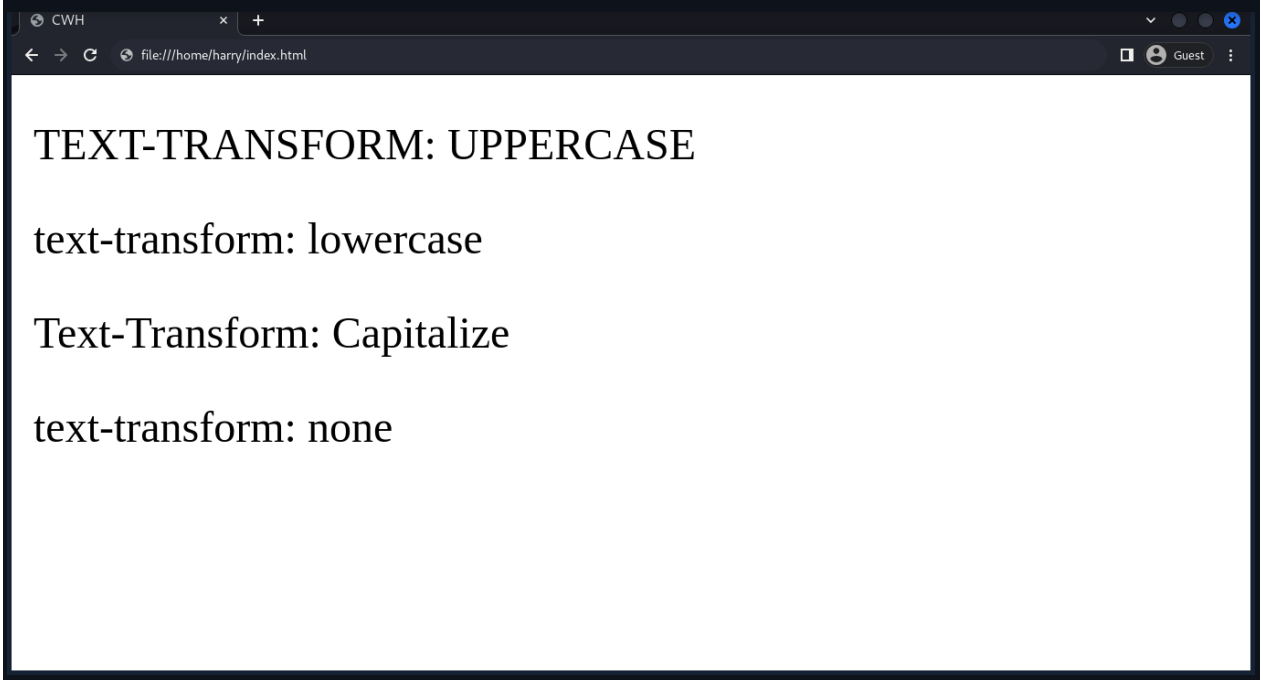
There are four values for `text-transform`:

- **uppercase**: transform text to uppercase (all capital letters).
- **lowercase**: transform text to lowercase (all small letters).
- **capitalize**: capitalize the first character of each word.
- **none**: to remove text transformation.

Example:

```
<html lang="en"><head>    <style>        #p1 {text-transform:
uppercase;}        #p2 {text-transform: lowercase;}        #p3 {text-
```

```
transform: capitalize;}        #p4 {text-transform: none;}
</style></head><body>    <p id="p1">text-transform: uppercase</p>
<p id="p2">text-transform: lowercase</p>    <p id="p3">text-transform:
capitalize</p>    <p id="p4">text-transform: none</p></body></html>
```



Letter Spacing

The `letter-spacing` property allows you to specify the spacing between each character in the text. The property values can be in pixels (px), em, rem, percentage (%), etc.

Example:

```
<html lang="en"><head>    <style>        h1 {            letter-
spacing: 5px;        }    </style></head><body>
<h1>CodeWithsri</h1></body></html>
```

Line Height

The `line-height` property controls the spacing between two lines of text.

Example:

```
<html lang="en"><head>    <style>        h1 {            line-height: 3.5;        }    </style></head><body>    <h1>CodeWithsri</h1> <h1>Developer and CodeWithHarry.com founder</h1></body></html>
```

The value of 3.5 means that the space between lines will be 3.5 times the height of the font size.

Text Shadow

The `text-shadow` property adds a shadow to the text.

Syntax:

```
selector {    text-shadow: horizontal offset, vertical offset, blur radius, color;}
```

Example:

```
<html lang="en"><head>    <style>        h1 {            text-shadow: 2px 3px 4px red;        }    </style></head><body> <h1>CodeWithsri</h1></body></html>
```

Text Overflow

The `text-overflow` property handles text that overflows its container. There are two values we can use with `text-overflow`:

- **ellipsis**: truncates overflowing text with an ellipsis.
- **clip**: clips the text without any visual indication.

Font Color

Font color defines the colour of the text or typography.

Syntax:

```
selector {    color: red;}
```

CSS Colors

The color property of CSS helps to set the color of the HTML element(s). This helps to set the foreground color of text, text decorations, and borders.

Syntax:

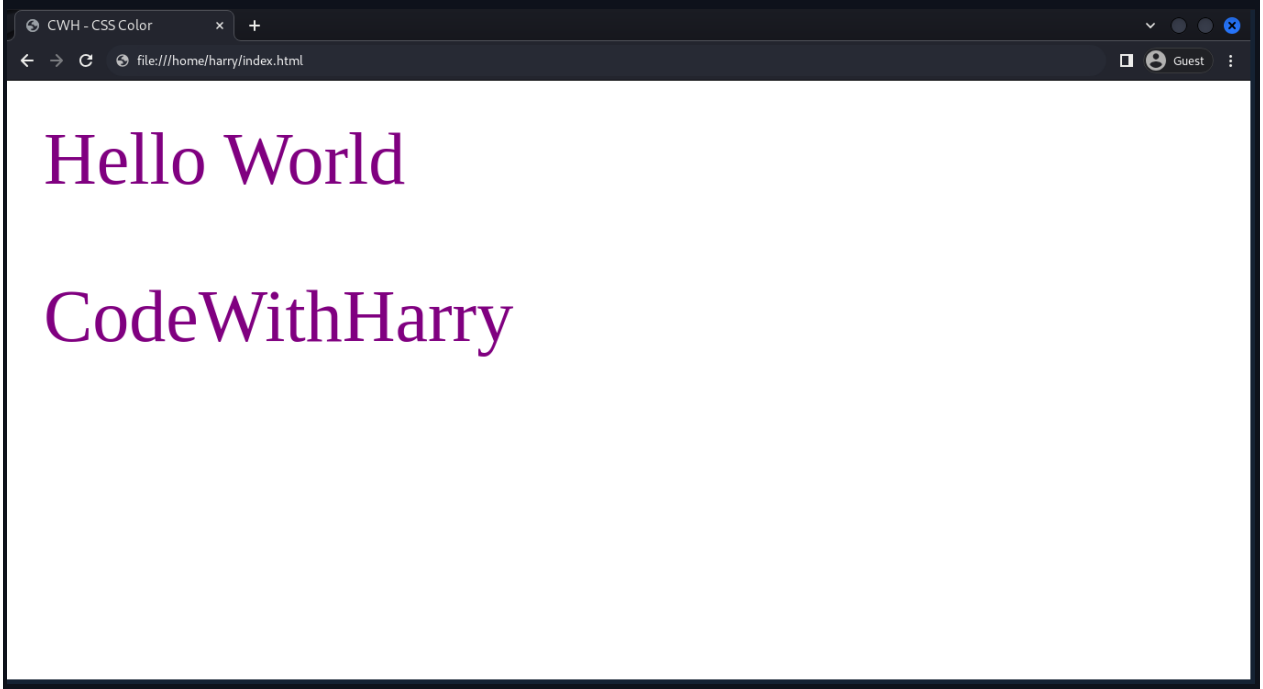
```
/* Syntaxselector {    color: value} */selector {    /*
colorname can be any colour, such as red, blue, yellow, purple,
green, etc. */    color: colorname}
```

Note: In CSS we use **color**, not colour.

Example:

```
<head>    <style>        p {            color: purple;        }
</style></head> <body>    <p>Hello World</p>
<p>CodeWithHarry</p></body></html>
```

Output:



There are many ways to set the property-value of color, with some of the most common listed below.

Hexadecimal notation:

The hex code consists of a hash(#) symbol followed by six characters. These six characters are arranged into a set of three pairs (RR, GG, and BB).

Each character pair defines the intensity level of the colour, where R stands for red, G stands for green, and B stands for blue.

The intensity value lies between 00 (no intensity) and ff (maximum intensity).

Breaking the Character Set (RRGGBB):

- **RR**: RR defines the intensity of color red, ranging from 00 (no red) to FF (maximum red).
- **GG**: GG defines the intensity of color Green, with values from 00 (no green) to FF (full green).
- **BB**: BB defines the intensity of color Blue, varying from 00 (no blue) to FF (full blue).

Syntax:

```
selector {    color: #RRGGBB;}
```

RGB

RGB stands for “Red, Green, Blue,” and it defines the colour value by taking three (red, green, blue) arguments.

Each argument value lies between 0 and 255.

Syntax:

```
selector {    color: rgb(red, green, blue);}
```

RGBA

Similar to RGB, in RGBA, **a** stands for alpha value, which defines the opacity of the color. The alpha value lies between 0 and 1.

Syntax:

```
selector {    color: rgba(red, green, blue, opacity);}
```

HSL

HSL stands for hue, saturation, and lightness. This is another way to set colour properties.

Breaking each keyword:

Hue(h)

- Hue represents the type of color. It is measured in degrees, and its value lies from 0 to 360.
- 0 degree represents black, 120 degree is for green, and 360 degree is for blue.

Saturation (S):

- Saturation controls the intensity or purity of the color. It is measured in percentage, and its value lies between 0% and 100%.
- 0% saturation is no color (grayscale), and 100% saturation is the most intense colour.

Lightness (L):

- Lightness determines how light or dark the colour is. It is measured in percentage, and its value lies between 0% and 100%.
- 0% lightness represents pure black, 50% lightness represents normal colour, and 100% lightness is pure white.

Syntax:

```
selector {    color: hsl(hue, saturation, lightness);}
```

Font Size

The font size property sets the size of the fonts.

It has some predefined sizes, such as small, medium, large, larger, etc.

The most commonly used units for font size are px (pixels), em (ems), rem (root ems), and percentage (%).

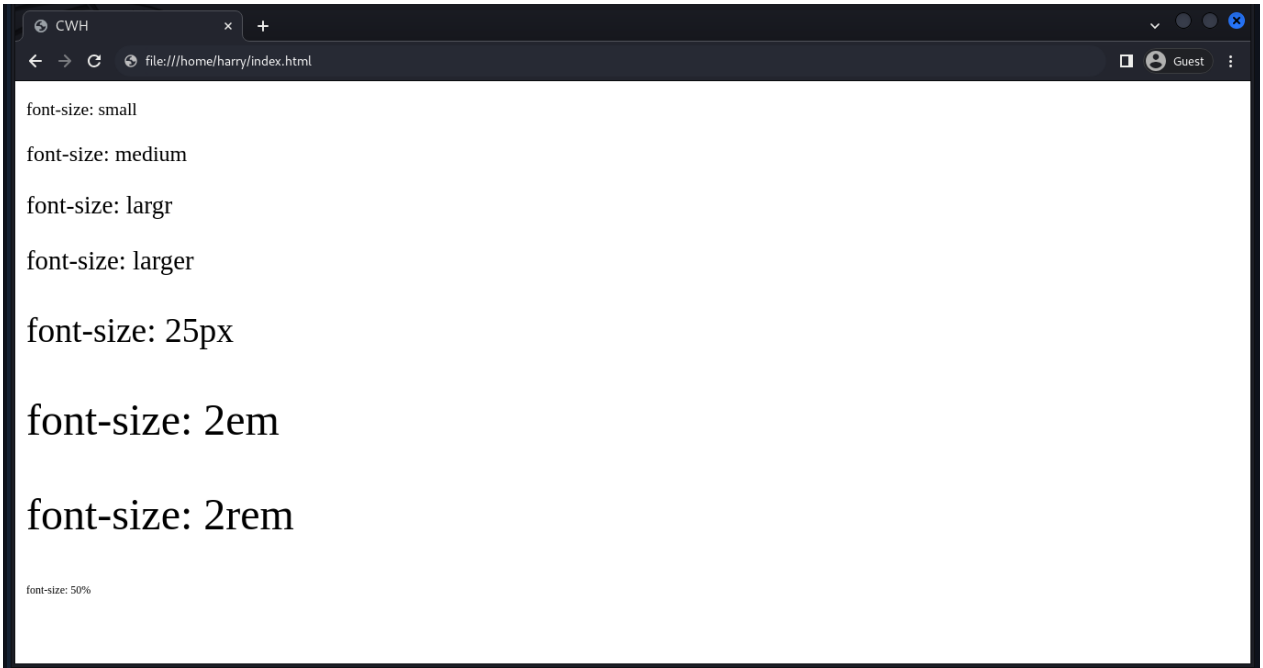
Quick notes:

- **px:** px is the absolute unit. This is useful for setting precise sizes.
- **em:** em is the relative unit, based on the font size of the parent element. If the element's font size is 1.5 em, that means the element will be 1.5 times the size of the parent.
- **rem:** rem is the relative unit, based on the font size of the root element, i.e., `<html>`.

Example:

```
<html lang="en"><head>    <style>        #p1 {            font-
size: small;        }        #p2 {            font-size:
medium;        }        #p3 {            font-size:
large;        }        #p4 {            font-size:
larger;        }        #p5 {            font-size:
25px;        }        #p6 {            font-size: 2em;        }
#p7 {            font-size: 2rem;        }        #p8
{            font-size: 50%;        }    </style></head><body>
<p id="p1">font-size: small</p>    <p id="p2">font-size:
```

```
medium</p>    <p id="p3">font-size: large</p>    <p
id="p4">font-size: larger</p>    <p id="p5">font-size: 25px</p>
<p id="p6">font-size: 2em</p>    <p id="p7">font-size: 2rem</p>
<p id="p8">font-size: 50%</p></body></html>
```



CSS Font Style

The font style property sets the style of the font.

There are three types of font styles: italic, normal, and oblique.

Quick notes:

- **italic:** Italic texts are slightly to the right.
- **normal:** Default text style.
- **Oblique:** Oblique is similar to italic but has less bend.

Sample code :

```
<html lang="en">
```

```
<head>
    <style>
        #p1 {
            font-style: italic;
        }
        #p2 {
            font-style: normal;
        }
        #p3 {
            font-style: oblique;
        }
    </style>
</head>
<body>
    <p id="p1">font-style: italic</p>
    <p id="p2">font-style: normal</p>
    <p id="p3">font-style: oblique</p>
</body>
</html>
```

CSS Display

This property specifies and determines how an element would be displayed on the website. By tweaking the value of the display we can show the content “inline”, “grid”, “flex”, and whatnot.

Let’s look at a few properties.

Display Inline

It only takes the space required for content, leaving the rest space for other elements to come. Setting other dimension properties isn't like width, height, margin or padding is not allowed in Inline Display.

```
Syntax: {display: inline;}
```

Display Block

It takes the full width available across the website page leaving a new line before and after the element.

```
Syntax: {display: block;}
```

Display Inline Block

This property is quite similar to inline but dimension properties are allowed in the case of the inline-block display.

Display None & Hidden

This property removes the element from the document, clearing the space taken by the element. When the display is hidden, though the element isn't visible the space is still taken by the element.

CSS FlexBox

FlexBox, also known as Flexible Box Layout, makes it easier to layout, align, and style items in a container while maintaining the responsiveness of the website.

To create a flexbox, you need to set the display of the container as `flex`.

Eg: `{display: flex;}`

This element is called the flex container, and it stores the sub-elements, which are known as flex items.

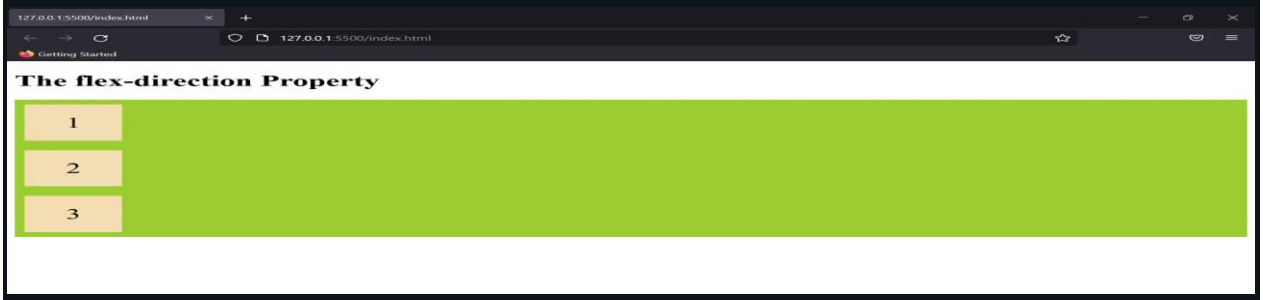
Flex Container Properties

The flex container properties are:

1. Flex Direction

It defines in which direction the flex elements would be displayed. It takes values like `row`, `column`, or "reverse" too.

Output:



2.Flex Wrap

By using this property, we can make our elements responsive for different screen sizes.

Eg:

```
.flex-container {    display: flex;    flex-direction: row;
background-color: yellowgreen;    flex-wrap: wrap;}
```

3. Justify Content

This property is used to set the position of content or rather align content along the main axis.

Eg:

```
.flex-container {    display: flex;    flex-direction: row;
background-color: yellowgreen;    justify-content: center;}
```

4. Align Items

Just like the `justify-content` property, `align-items` defines the alignment of the flex container but along the cross-axis.

Eg:

```
.flex-container {    display: flex;    height: 200px;    flex-direction: row;    background-color: yellowgreen;    align-items: center;}
```

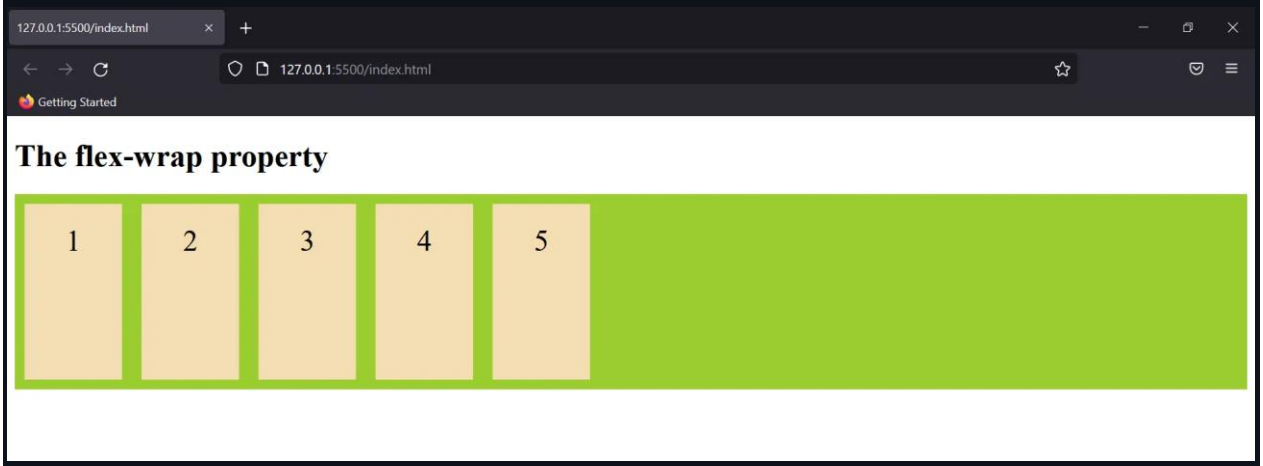
5. Align Content

This property is very similar to `align-items`, but here, rather than the flex items, the content present in the item is selected for the property.

Eg:

```
.flex-container {    display: flex;    height: 200px;    flex-direction: row;    background-color: yellowgreen;    align-content: center;}
```

Output:



Flex Items Properties

The flex item properties are:

1. Order

As the name suggests, this property sets the order in which the flex items are shown.

Eg:

```
<div style="order: 4;">1</div><div style="order:
3;">2</div><div style="order: 1;">3</div><div style="order:
5;">4</div><div style="order: 2;">5</div>
```

Output:

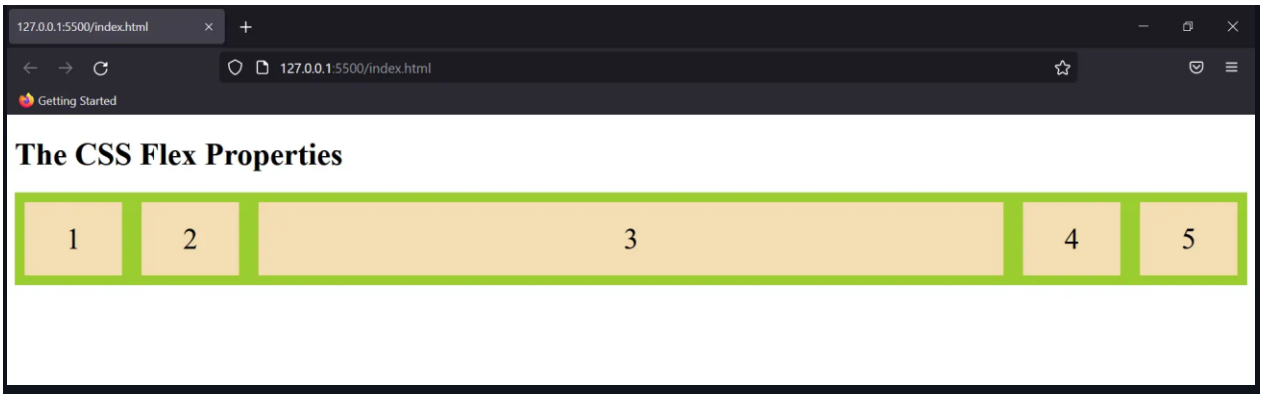
2. Flex Grow & Shrink

Decides the relative size of flex items. By default, this property is zero, and thus items have the same size.

Eg:

```
<div>1</div><div>2</div><div style="flex-grow:
3;">3</div><div>4</div><div>5</div>
```

Output:



We can also use `flex-shrink` to decrease the size of an element.

3. Align Self

This property allows default alignment to be overridden for the individual flex items. Try adding inline CSS to see how this property is used.

CSS Grid

Just like FlexBox, CSS Grid with the use of rows and columns, makes it easier to style website elements.

CSS display property allows two grid properties: Grid and Inline Grid. The elements placed in the grid container are called grid items.

Most of the properties of Grid are similar to FlexBox.

Let’s look at how you can create a grid in CSS.

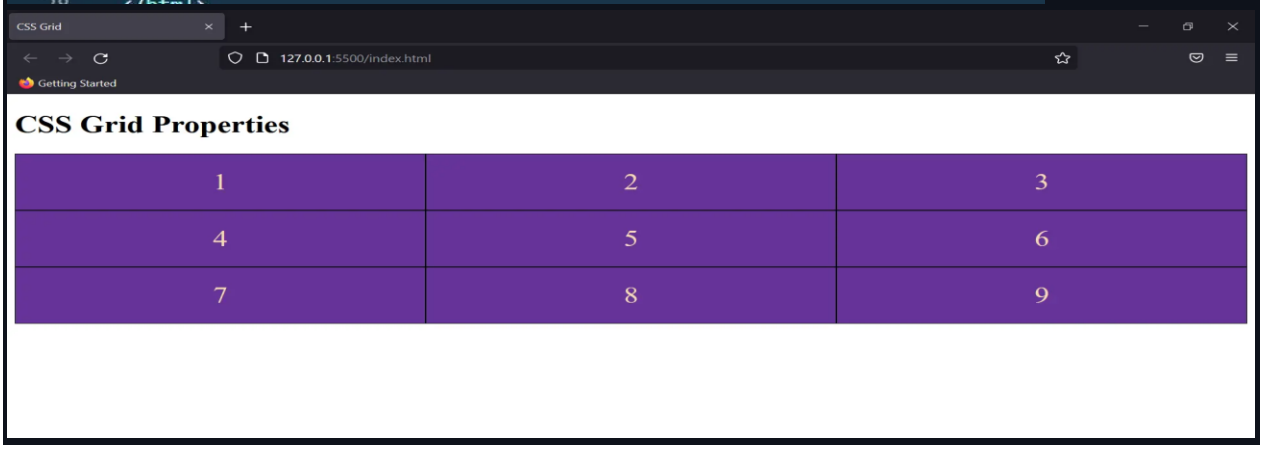
index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>CSS Grid</title>
    <style>
        span {
            display: grid;
            grid-template-columns: auto auto auto;
            background-color: rebeccapurple;
        }

        div {
            color: wheat;
            padding: 20px;
            border: 1px solid black;
            font-size: 30px;
            text-align: center;
        }
    </style>
</head>

<body>
    <h1>CSS Grid Properties</h1>
    <span class="grid-container">
        <div class="grid-item">1</div>
        <div class="grid-item">2</div>
        <div class="grid-item">3</div>
        <div class="grid-item">4</div>
        <div class="grid-item">5</div>
        <div class="grid-item">6</div>
        <div class="grid-item">7</div>
        <div class="grid-item">8</div>
        <div class="grid-item">9</div>
    </span>
</body>
```



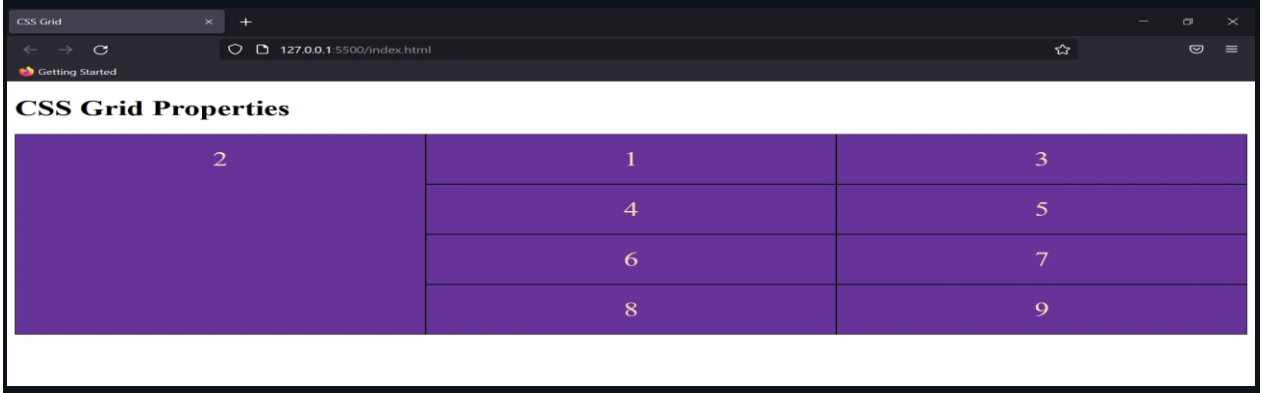
Grid Row & Column Property

This is just adjacent to the flex-grow property in FlexBox. It decides how many columns the selected element will take in the Grid.

Example:

```
<div class="grid-item" style="grid-row: 1/5;">2</div>
```

Output:



CSS Shadows

When it comes to web design, CSS (Cascading Style Sheets) plays an indispensable role in enhancing the visual aesthetics of a website. Among its vast array of properties, the ones that help bring depth and focus to an element are the shadow and outline properties. Let's delve into the magic of CSS shadows and outlines.

Box Shadows

`box-shadow` is a popular CSS property that enables designers to add shadow effects around an element's frame. It can be used to give any element, be it a div, image, or button, a 3D feel or to emphasize on hover.

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

- `h-offset` and `v-offset`: Determines the shadow's horizontal and vertical position.
- `blur`: The larger the value, the blurrier the shadow.

- `spread`: Expands or shrinks the shadow size.
- `color`: Defines the shadow color.
- `inset`: Makes the shadow inner.

Here is an example:

```
.div-element { box-shadow: 5px 5px 15px 5px #888888; }
```

Text Shadows

`text-shadow` is utilized to give shadows specifically to the text. It can elevate the readability of the text or give it an elegant appearance.

The Syntax for Text-Shadows is as follows:

```
text-shadow: h-offset v-offset blur color;
```

Here is an example:

```
.text-element { text-shadow: 2px 2px 4px #888888; }
```

Outlines

An outline is a line that is drawn around elements, outside the borders, to make the element "stand out". It's commonly used for accessibility purposes, like highlighting focused elements.

The Syntax for Outlines is as follows:

```
outline: width style color;
```

- `width`: Sets the outline width.
- `style`: Determines the style of the outline such as solid, dotted, or dashed.
- `color`: Sets the outline color.

Here is an example:

```
.button-element:focus { outline: 2px solid blue; }
```

Differences between Outlines and Borders:

While both can visually appear similar, outlines differ from borders in a few ways:

1. **Position**: Outlines don't take up space; they're drawn around the element, outside of any border.
2. **Offset**: Using the `outline-offset` property, you can set the space between an outline and the edge or border of an element.
3. **Width**: Borders can have varying widths on different sides, outlines have a uniform width.
4. **Rounded Corners**: Borders can have rounded corners using `border-radius`, while outlines generally cannot.

CSS List Styles

Lists are a common element in web design used to prevent content from being un-organized. CSS allows you to style lists to match the design aesthetic and improve the readability of the web page.

Following are the various techniques for styling HTML lists.

Unordered list styling

To style unordered lists (bulleted lists), we can change the list item marker.

Syntax:

```
ul {    list-style-type: value;}
```

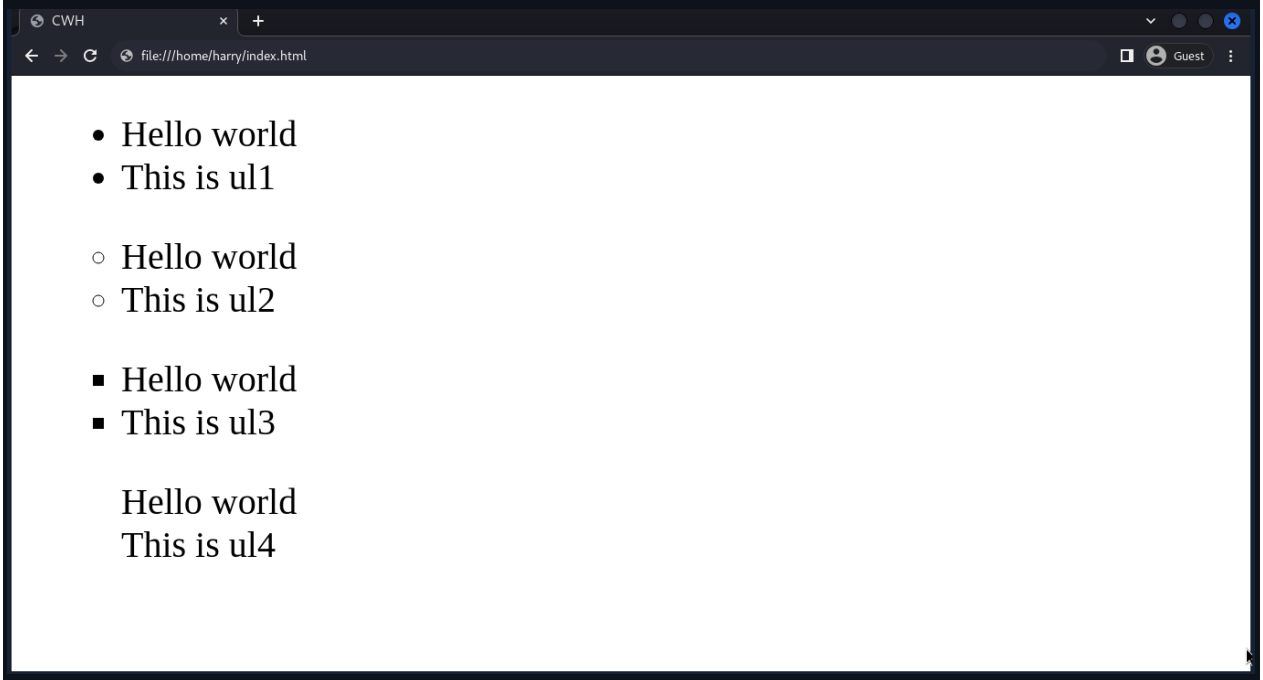
The value can be:

- **disc:** (default) - Filled circle marker

- **circle:** hollow circle marker
- **square:** a filled square marker
- **none:** No marker (remove bullets)

Example:

```
<head>    <style>        .ul1 {           list-style-type:
disc;        }        .ul2 {           list-style-type:
circle;        }        .ul3 {           list-style-type:
square;        }        .ul4 {           list-style-type:
none;        }    </style></head><body>    <ul class="ul1">
<li>Hello world</li>        <li>This is ul1</li>    </ul>    <ul
class="ul2">        <li>Hello world</li>        <li>This is ul2</li>
</ul>    <ul class="ul3">        <li>Hello world</li>        <li>This
is ul3</li>    </ul>    <ul class="ul4">        <li>Hello world</li>
<li>This is ul4</li>    </ul></body></html>
```



Ordered Lists Styling

For ordered lists (numbered lists), we can customize the list item numbering.

Syntax:

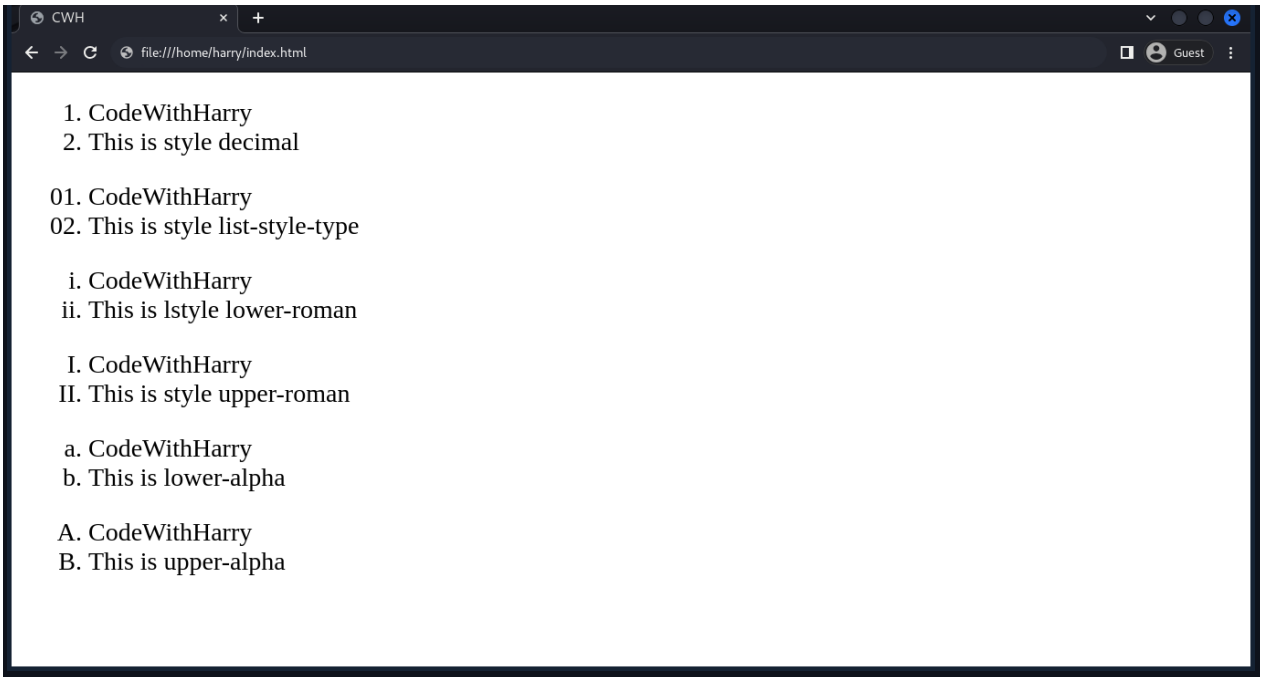
```
ol {    list-style-type: value;}
```

The value can be any of the following:

1. **decimal:** (default) - Decimal numbers (1, 2, 3, etc.)
2. **decimal-leading-zero:** Decimal numbers with leading zeros (01, 02, 03, etc.)
3. **lower-roman:** lowercase Roman numbers (i, ii, iii,...)
4. **upper-roman:** uppercase Roman numbers (I, II, III,...)
5. **lower-alpha:** lowercase alphabetical letters (a, b, c, etc.)
6. **upper-alpha:** uppercase alphabetical letters (A, B, C, etc.)

Example:

```
<html lang="en"><head>    <title>CWH</title>    <style>        .ul1
{            list-style-type: decimal;        }        .ul2
{            list-style-type: decimal-leading-
zero;        }        .ul3 {            list-style-type: lower-
roman;        }        .ul4 {            list-style-type: upper-
roman;        }        .ul5 {            list-style-type: lower-
alpha;        }        .ul6 {            list-style-type: upper-
alpha;        }    </style></head><body>    <ul class="ul1">
<li>CodeWithHarry</li>        <li>This is style decimal</li>    </ul>
<ul class="ul2">        <li>CodeWithHarry</li>        <li>This is
style list-style-type</li>    </ul>    <ul class="ul3">
<li>CodeWithHarry</li>        <li>This is style lower-roman</li>
</ul>    <ul class="ul4">        <li>CodeWithHarry</li>
<li>This is style upper-roman</li>    </ul>    <ul class="ul5">
<li>CodeWithHarry</li>        <li>This is lower-alpha</li>    </ul>
<ul class="ul6">        <li>CodeWithHarry</li>        <li>This is
upper-alpha</li>    </ul></body></html>
```



List-style position

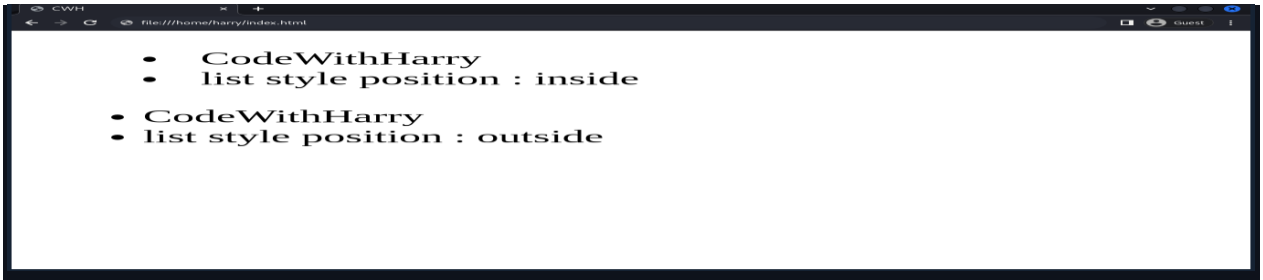
The 'list-style-position' property determines where the list markers (bullets or numbers) are placed in relation to the content.

It has two values:

- **inside:** (default) List markers are inside the content's box. This is the default behavior.
- **outside:** List markers are outside the content's box, typically to the left of the content, creating a hanging indent effect.

Example:

```
<html lang="en"><head>    <style>        .ul1 {            list-style-
position: inside;        }        .ul2 {            list-style-
position: outside;        }    </style></head><body>    <ul
class="ul1">        <li>CodeWithHarry</li>        <li>list style
position : inside</li>    </ul>    <ul class="ul2">
<li>CodeWithHarry</li>        <li>list style position : outside</li>
</ul></body></html>
```



Removing the default list styles

To remove default list styles (bullets or numbers), set the 'list-style' property to 'none'.

Example:

```
<html lang="en"><head>    <style>        .ul1 {            list-style:
circle;        }        .ul2 {            list-style: none;        }
</style></head><body>    <ul class="ul1">
<li>CodeWithHarry</li>        <li>Developer</li>    </ul>    <ul
class="ul2">        <li>CodeWithHarry</li>        <li>list
style :none</li>    </ul></body></html>
```

Custom List Style

To set a custom list style, set the 'list-style-type' to your new required custom style.

Example:

```
<html lang="en"><head>    <style>        .ul1 {            list-style-
type: "😍";        }    </style></head><body>    <ul class="ul1">
```

```
<li>CodeWithHarry</li>        <li>Developer</li>    </ul></body></htm
```

CSS Positioning

The CSS positions allow you to precisely control the placement of an element on the web page.

It helps to determine how elements are placed inside the container element and how they interact with the other elements on the page.

There are various types of position property values, such as:

Static (Default)

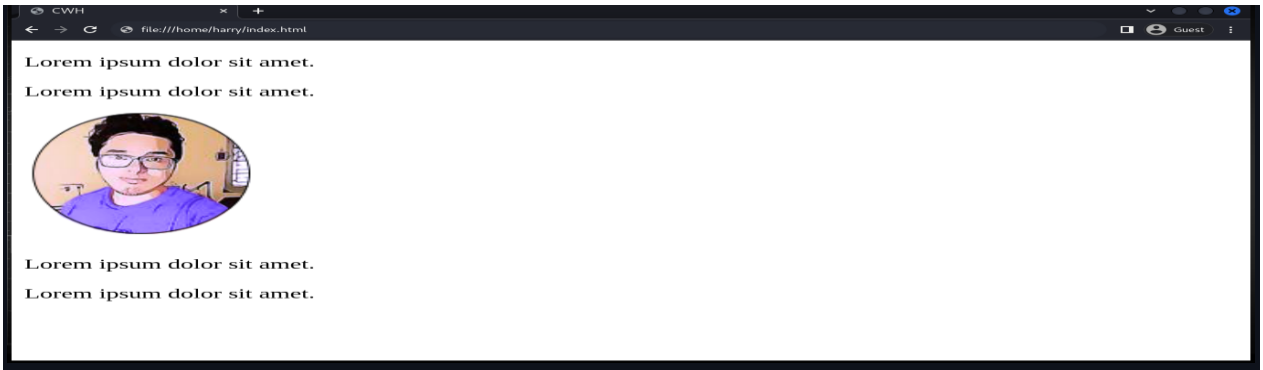
The elements are positioned according to the normal flow of the document.

Syntax:

```
selector {    position: static;}
```

Example:

```
<head>    <style>        img {            position: static;        }
</style></head><body>    <p id="p1">Lorem ipsum dolor sit amet.</p>
<p id="p2">Lorem ipsum dolor sit amet.</p>    <img src="./logo.png"
alt="logo">    <p id="p3">Lorem ipsum dolor sit amet.</p>    <p
id="p4">Lorem ipsum dolor sit amet.</p></body></html>
```



Relative

Elements are positioned relative to the normal position in the document.

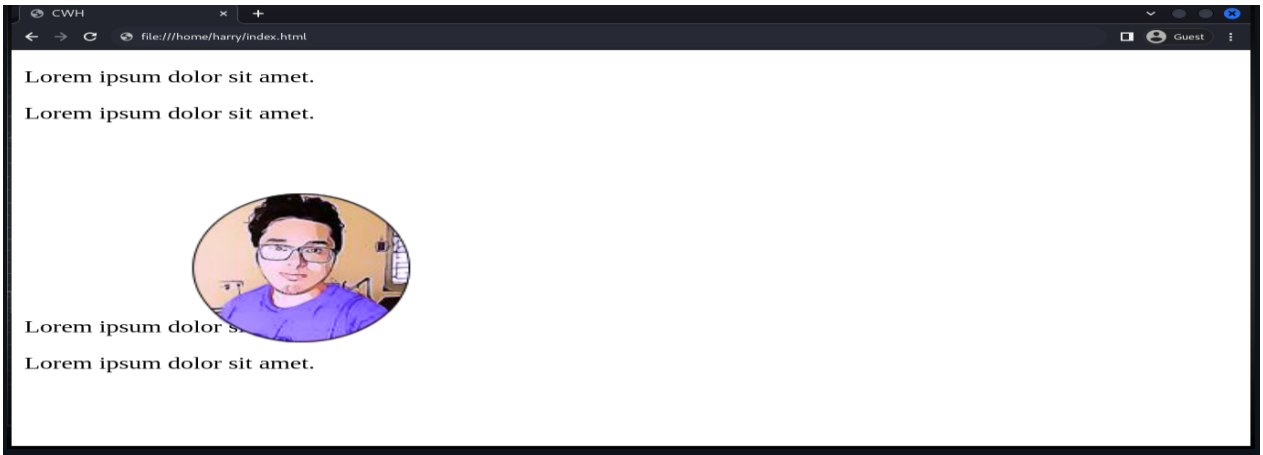
You can use the top, right, bottom, and left properties to move the element from its original position.

Syntax:

```
selector {    position: relative;}
```

Example:

```
<head>    <style>        img {            position: relative;
left: 100px;            top: 50px;        }    </style></head><body>
<p id="p1">Lorem ipsum dolor sit amet.</p>    <p id="p2">Lorem ipsum
dolor sit amet.</p>    <img src="./logo.png" alt="logo">    <p
id="p3">Lorem ipsum dolor sit amet.</p>    <p id="p4">Lorem ipsum
dolor sit amet.</p></body></html>
```



Here you can see that the image is repositioned from its original place, and the gap is not filled by another element.

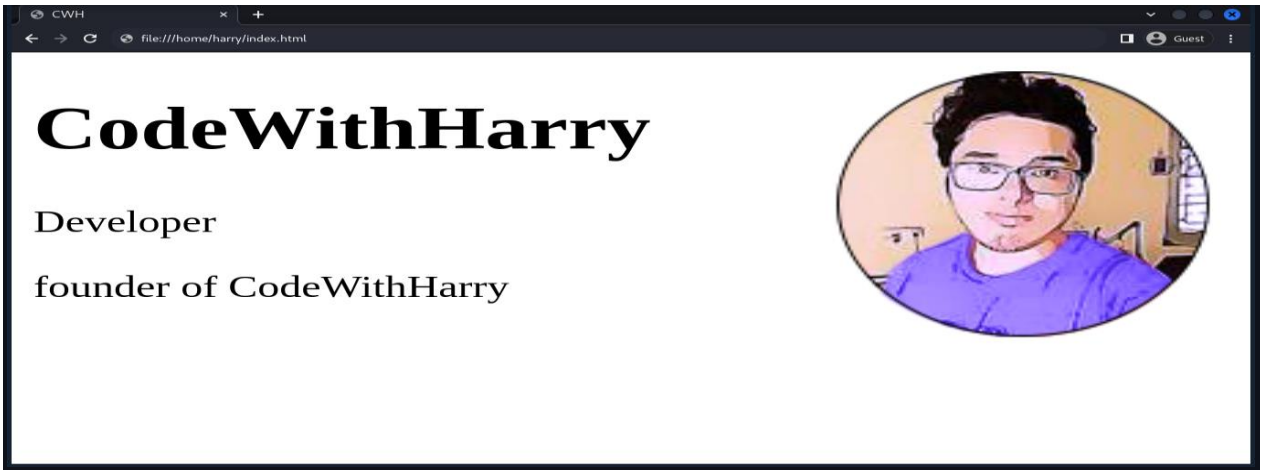
Absolute

Elements are positioned relative to the closest positioned ancestor (parent), which means we need to have a parent element with a positioning other than 'static'.

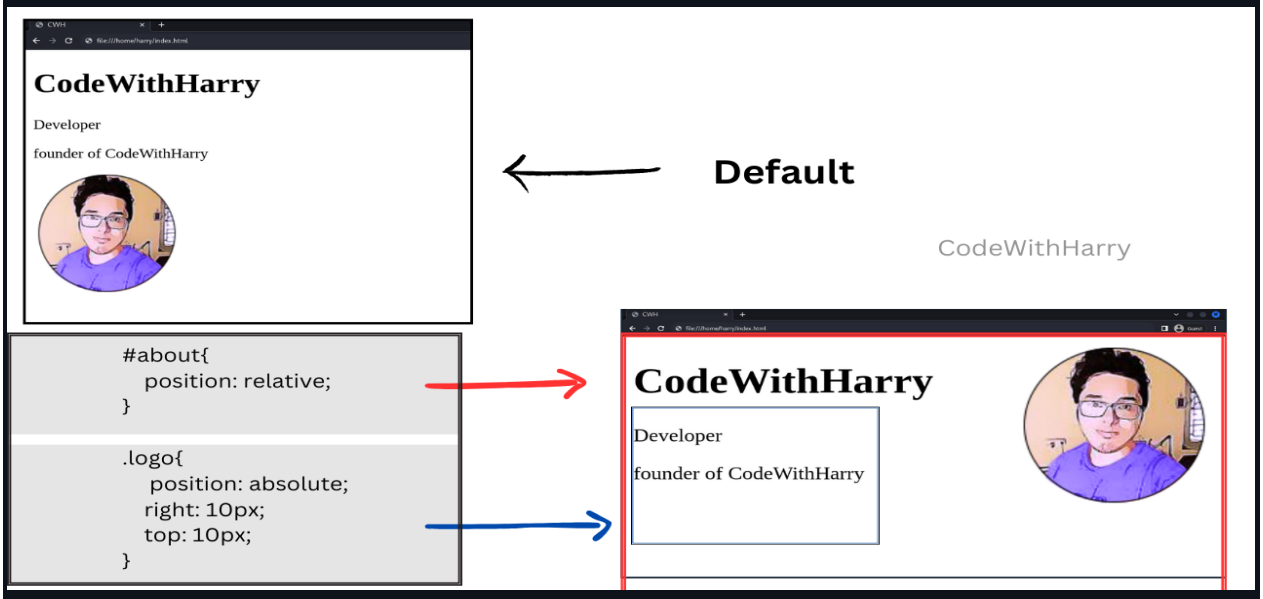
Note: An absolutely positioned element is removed from the normal flow.

Example:

```
<head>    <style>        #about {            position:
relative;        }        .logo {            position: absolute;
right: 10px;            top: 10px;        }    </style></head><body>
<h1>CodeWithHarry</h1>    <div class="about">        <p>Developer</p>
<p>Founder of CodeWithHarry</p>        <img class="logo"
src="./logo.png" alt="logo">    </div></body></html>
```



Explanation:



Here, as we have set position relative to the body and absolute to the about section, the about section position can be manipulated with the left of top, left, right, and bottom.

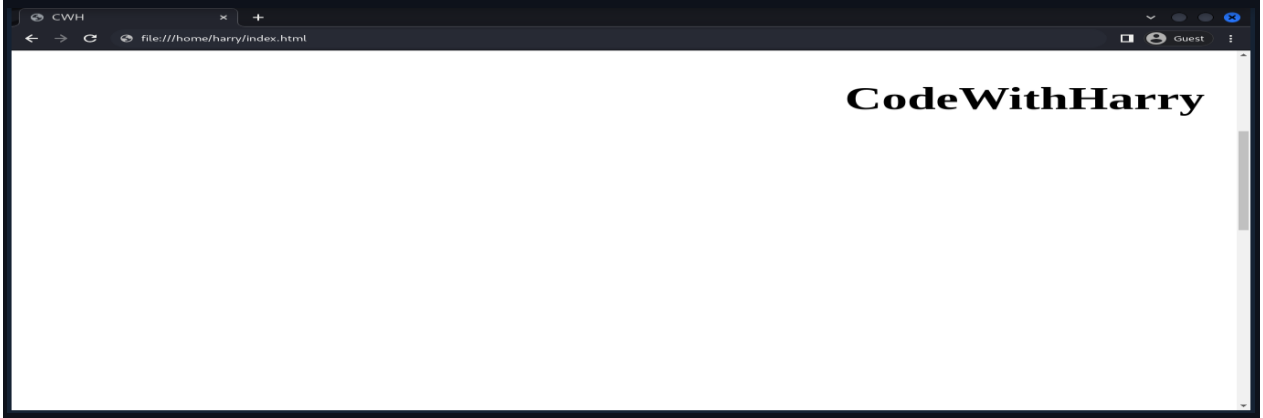
Fixed

Elements are positioned relative to the viewport (screen) and do not move when the page is scrolled.

This is useful for creating elements like fixed headers or footers.

Example:

```
<head>    <style>        h1 {            position: fixed;
top: 10px;            right: 20px;        }    </style></head><body>
<h1>CodeWithHarry</h1></body></html>
```



Here, the image position will be fixed.

Float

The float property is used to shift an element to the left or right within its containing element. For more details, follow CSS Float.

Sticky

Position sticky is a hybrid between 'relative' and 'fixed'.

It allows an element to become "stuck" to the top or bottom of its container when scrolling, but it behaves like relative positioning within the container until it reaches a specified offset.

Example:

```
<head>    <style>        h1 {            position: sticky;
top: 10px;            right: 20px;        }    </style></head><body>
<h1>CodeWithHarry</h1></body></html>
```