**CASCADING STYLE SHEETS**

CSS, or Cascading Style Sheets, is the language used to style and enhance HTML documents. It defines the presentation of HTML elements on a web page, enabling changes to fonts, colors, sizes, spacing, column layouts, and animations.

**Table of Content**

1. [Basic CSS Example](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#basic-css-example)
2. [CSS Tutorial](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-tutorial)
3. [CSS Fundamentals](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-fundamentals)
4. [CSS Styling Techniques](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-styling-techniques)
5. [CSS Responsive Design and Media Queries](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-responsive-design-and-media-queries)
6. [Advanced CSS Topics](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#advanced-css-topics)
7. [CSS Online Quizzes](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-online-quizzes)
8. [CSS Practical Projects](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-practical-projects)
9. [CSS Interview Preparation](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-interview-preparation)
10. [CSS Preprocessors](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-preprocessors)
11. [CSS Frameworks](https://www.geeksforgeeks.org/css-tutorial/?ref=dhm#css-frameworks)

**What is CSS?**

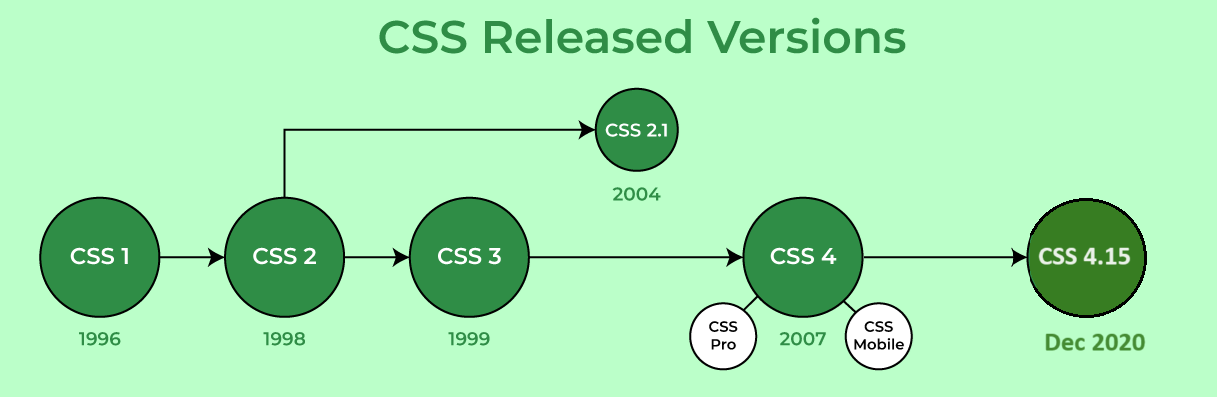
CSS, or **Cascading Style Sheets**, is a language used to style and enhance websites. It controls how **HTML elements**—such as *text*, *images*, and *buttons*—are displayed on a webpage. **With CSS**, you can adjust font sizes and colors, add backgrounds, and manage the layout, transforming a basic webpage into a visually appealing and user-friendly experience. CSS also simplifies **layout management** across multiple web pages by **using external stylesheets** stored in CSS files.

CSS (Cascading Style Sheets) is a language designed to simplify the process of making web pages presentable. It allows you to apply styles to HTML documents, describing how a webpage should look by prescribing colors, fonts, spacing, and positioning. CSS provides developers and designers with powerful control over the presentation of HTML elements.

HTML uses tags and CSS uses rulesets. CSS styles are applied to the HTML element using selectors. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

**Why CSS?**

* **Saves Time:**Write CSS once and reuse it across multiple HTML pages.
* **Easy Maintenance:** Change the style globally with a single modification.
* **Search Engine Friendly:** Clean coding technique that improves readability for search engines.
* **Superior Styles:** Offers a wider array of attributes compared to HTML.
* **Offline Browsing:**CSS can store web applications locally using offline cache, allowing offline viewing.



**CSS Syntax**

CSS consists of style rules that are interpreted by the browser and applied to the corresponding elements. A style rule set includes a selector and a declaration block.

* **Selector:** Targets specific HTML elements to apply styles.
* **Declaration:** Combination of a property and its corresponding value.

// HTML Element

<h1>GeeksforGeeks</h2>

// CSS Style

h1 { color: blue; font-size: 12px; }

**Where -**

Selector - h1

Declaration - { color: blue; font-size: 12px; }

* The **selector points** to the HTML element that you want to style.
* The **declaration block** contains one or more declarations separated by semicolons.
* Each **declaration** includes a CSS property name and a value, separated by a colon.

p {

color: blue;

text-align: center;

}

CSS **declaration always ends with a semicolon**, and declaration blocks are surrounded by **curly braces**. In above example, all paragraph element (<p> tag) will be centre-aligned, with a blue text color.

**Web Page with & without CSS**

**Without CSS:** In this example, we have not added any CSS style.

<!DOCTYPE html>

<html>

<head>

<title>Simple Web Page</title>

</head>

<body>

<main>

<h1>HTML Page</h1>

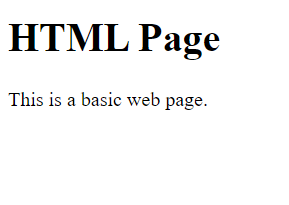
<p>This is a basic web page.</p>

</main>

</body>

</html>

**OUTPUT:**



**Using CSS:** In this example, we will add some CSS styles inside the HTML document to show how CSS makes a HTML page attractive and user-friendly.

<!DOCTYPE html>

<html>

<head>

<title>Simple web page</title>

<style>

main {

width: 600px;

height: 200px;

padding: 10px;

background: beige;

}

h1 {

color: olivedrab;

border-bottom: 1px dotted darkgreen;

}

p {

font-family: sans-serif;

color: orange;

}

</style>

</head>

<body>

<main>

<h1>My first Page</h1>

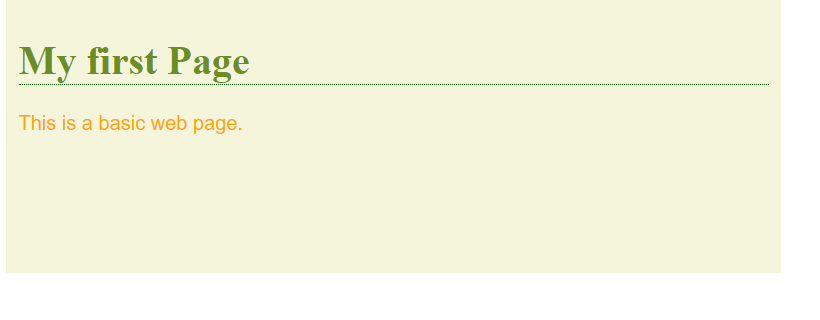
<p>This is a basic web page.</p>

</main>

</body>

</html>

**OUTPUT:**



CSS is essential for creating visually appealing and maintainable web pages. It enhances the website look and feel and user experience by allowing precise control over the presentation of HTML elements.

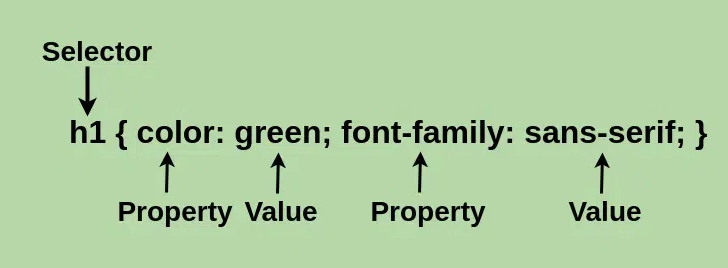
**CSS Syntax**

A **CSS Syntax** rule consists of a selector, property, and its value. The selector points to the HTML element where the CSS style is to be applied. The CSS property is separated by semicolons. It is a combination of the selector name followed by the **property: value** pair that is defined for the specific selector. let us see the syntax and how we can use the CSS to modernize the website.

**Syntax:**

selector { Property: value; }

For instance, we have declared a heading tag (h1) along with having assigned some ***property: value*** pair that is used to style the heading tag. Here, ***h1***is the selector,***{ color: green; font-family: sans-serif; }***is a declaration block and it can contain one or more declarations separated by semicolons,***color: green;***is a *property: value* pair that is applied to the HTML element to style them.



Every declaration has a CSS property name and a value, separated by a **colon(:)** and is surrounded by **curly braces({ })**. For declaring the multiple CSS properties, it can be separated by the **semicolon(;)**.

Let’s define each of these:

* **Declaration:**A combination of a property and its corresponding value.
* **Selector:**Used to target and select specific HTML elements to apply styles to.
* **Property:** Defines the specific aspect or characteristic of an element that you want to modify.
* **Value:** Assigned setting or parameter for a given property, determining how the selected element should appear or behave.

**Different Ways to Use CSS**

CSS has three ways to style the HTML:

* **Inline**: Add styles directly to HTML elements using the style attribute (limited use).
* **Internal**: Place styles within a *<style>* tag inside the HTML file, usually within the *<head>* section
* **External**: Create a separate CSS file with a *.css* extension and link it to your HTML file using the*<link>* tag.

**Advantages of CSS:**

* CSS you simply got to specify a repeated style for element once & use it multiple times as because CSS will automatically apply the required styles.
* CSS style is applied consistently across variety of sites. One instruction can control several areas which is advantageous.
* Web designers need to use few lines of programming for every page improving site speed.
* CSS simplifies both website development and maintenance as a change of one line of code affects the whole web site and maintenance time.
* It is less complex therefore the effort is significantly reduced.
* It helps to form spontaneous and consistent changes.
* CSS changes are device friendly. With people employing a batch of various range of smart devices to access websites over the web, there’s a requirement for responsive web design.
* It has the power for re-positioning. It helps us to determine the changes within the position of web elements who are there on the page.
* These bandwidth savings are substantial figures of insignificant tags that are indistinct from a mess of pages.
* Easy for the user to customize the online page
* It reduces the file transfer size.

**Disadvantages of CSS:**

* CSS, CSS 1 up to CSS3, result in creating of confusion among web browsers.
* With CSS, what works with one browser might not always work with another. The web developers need to test for compatibility, running the program across multiple browsers.
* There exists a scarcity of security.
* After making the changes we need to confirm the compatibility if they appear. The similar change effects on all the browsers.
* Browser compatibility (some styles sheet are supported and some are not).
* CSS works differently on different browsers. IE and Opera supports CSS as different logic.
* There might be cross-browser issues while using CSS.
* There are multiple levels which creates confusion for non-developers and beginners.

**Basic CSS Example**

Below example shows inline, internal, and external style sheet with different properties.

<!-- File name: **index.html** -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

**<!-- Importing External CSS -->**

**<link rel="stylesheet" href="style.css" />**

**<!-- Using Internal CSS -->**

**<style>**

**h2 {**

**color: green;**

**background-color: orange;**

**}**

**</style>**

    <title>Inline Internal and External CSS Demo</title>

</head>

<body>

        <!-- Using Inline CSS -->

        <h2 style="text-align: center;">Welcome To CSS</h2>

        <p>CSS Tutorial</p>

</body>

</html>

/\* External CSS \*/

/\* File name: **style.css** \*/

p {

    text-align: center;

    color: chartreuse;

    font-size: 33px;

  }

**OUTPUT:**



* **FAQs:**

1. **What is CSS?**

*CSS (Cascading Style Sheets) is a language for styling HTML or XML documents, controlling layout, colors, fonts, and overall appearance to enhance user experience.*

1. **Why is CSS important?**

*CSS separates content from design, improves accessibility, enhances user experience, and provides responsive designs for different devices and screen sizes.*

1. **How do you add CSS to a web page?**

* *Inline CSS: Using style attribute within HTML.*
* *Internal CSS: Using <style> block in <head>.*
* *External CSS: Linking a CSS file with <link> tag.*

1. **What is the syntax of CSS?**

*CSS syntax: selector { property: value; }. Selectors target elements; properties define style attributes; values specify styling details.*

1. **What are CSS selectors?**

*CSS selectors target elements for styling. Common types: element, class, ID, and attribute selectors.*

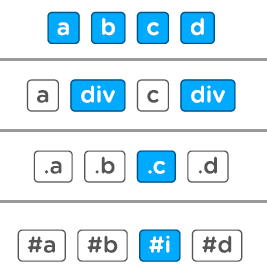
1. **What are CSS properties?**

*CSS properties define styles like color, font-size, margin, and padding. Example: p { color: blue; }.*

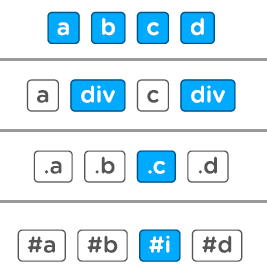
**SELECTORS IN CSS:**

**CSS selectors** are used *to select the content you want to style*. Selectors are the part of CSS rule set. CSS selectors select HTML elements are categorised in to following

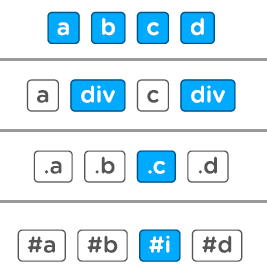
1. Basic/Simple Selectors
2. Combination
3. Attribute
4. Pseudo Element
5. Pseudo Class State
6. Pseudo Class position / other
7. Basic/Simple Selectors:
   1. **Universal Selector (\*)** : Selects all elements present on HTML page and applies the style to all elements.



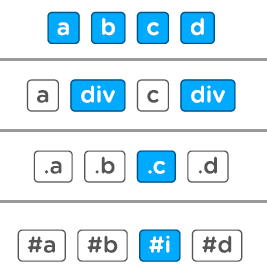
* 1. **Type Selector (Html\_Tag)** : Selects specific html tag and applies the style to only that tag i.e. whenever this tag appears the style gets applied to it.



* 1. **Class Selector(.Class\_Name)** : Creates a group of style properties and we use that class name as style to one or more elements in HTML.



* 1. **ID Selector (#ID)** : Create a group of style properties and assign to only one element in a HTML. We cannot use ID selector more than one.



**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

    <style>

        /\*Universal Selector\*/

        \* {

            background-color: aquamarine;

        }

        /\*Tyle Selector or Element Selector\*/

        span {

            font-style: italic;

            color: blue;

        }

        /\*Class Selector\*/

        .MyClassStyle {

            padding-left: 70px;

            text-decoration: underline;

            background: rgba(133,133,350,0.5);

        }

        /\*ID Selector\*/

        #MyID {

            background: greenyellow;

            text-decoration: dotted;

            font-family: 'Courier New', Courier, monospace;

            font-weight: bolder;

            font-size: 21px;

        }

    </style>

</head>

<body>

    <div class="MyClassStyle">This is a div</div>

    <span>This is a span 1</span>

    <ul>

        <li>Item 1</li>

        <li>Item 2</li>

        <li class="MyClassStyle">Item 3</li>

        <li>Item 4</li>

        <li>Item 5</li>

        <li>Item 6</li>

        <li>Item 7</li>

        <span id="MyID">This is a span 2</span></span>

    </ul>

    <div>

        <label for="username">User Name :</label>

        <input class="MyClassStyle" type="text" name="username" id="username" value="Text box is padded to Left....">

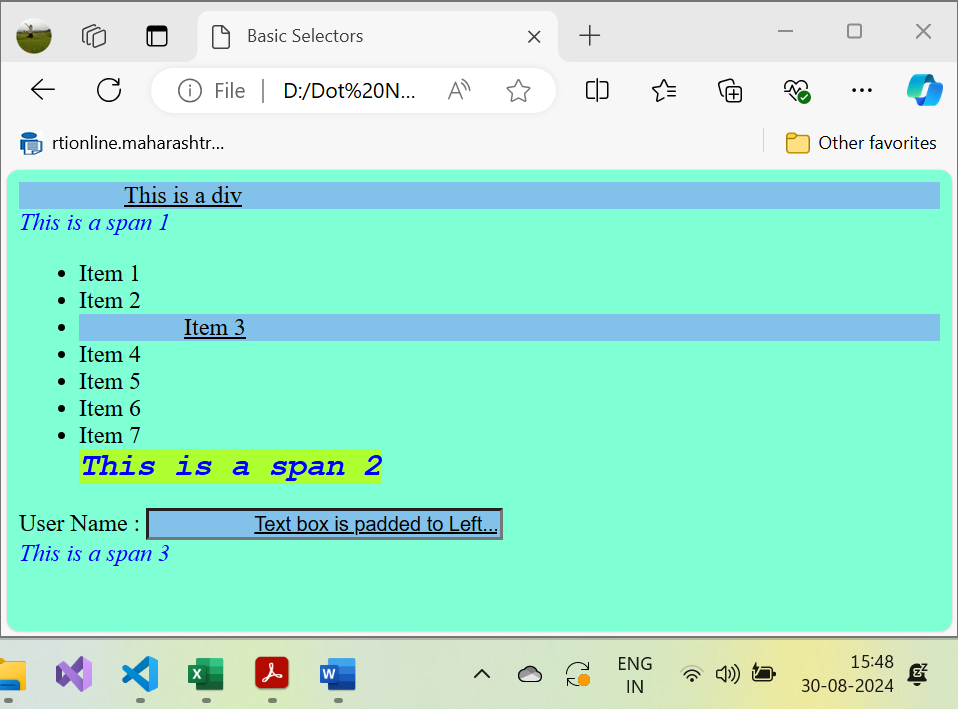
    </div>

    <span>This is a span 3</span></span>

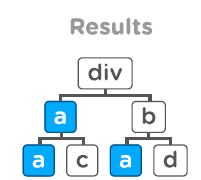
</body>

</html>

Output:



1. **Combination:** A CSS selector can contain more than one Basic/Simple selector. Between the Basic/Simple selectors, we can include a combinator.
   1. **Descendant Selector (div a):** The descendant selector matches all elements that are descendants of a specified first element. Space between two elements indicates anything comes under first element with direct child or child of child of ….child second element, style will get applied to it.



Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Combination Adjacent Sibling Selectors</title>

    <style>

    /\*Adjacent Sibling Selector \*/

    ul li{

        background: lightblue;

    }

    div b{

        background-color: aqua;

        font-size: 33px;

    }

    div .green{

        color: green;

    }

    </style>

</head>

<body>

    <div>This is a div1</div>

    <span>This is a span 1</span>

    <div>

        <span>

            <b>This is nested elements</b>

        </span>

    </div>

    <div>

        <span>

            <b><i class="green">This is nested elements</i></b>

        </span>

    </div>

    <ul>

        <li>Item 1</li>

        <li>Item 2</li>

        <li>Item 3</li>

        <li>Item 4</li>

        <li>Item 5</li>

        <li>Item 6</li>

        <li>Item 7</li>

        <span>This is a span 2</span></span>

        <div>This is a div3</div>

    </ul>

    <div>

        <label class="green" for="username">User Name :</label>

        <input style="color:pink;" type="text" name="username" id="username" value="Text box is padded to Left....">

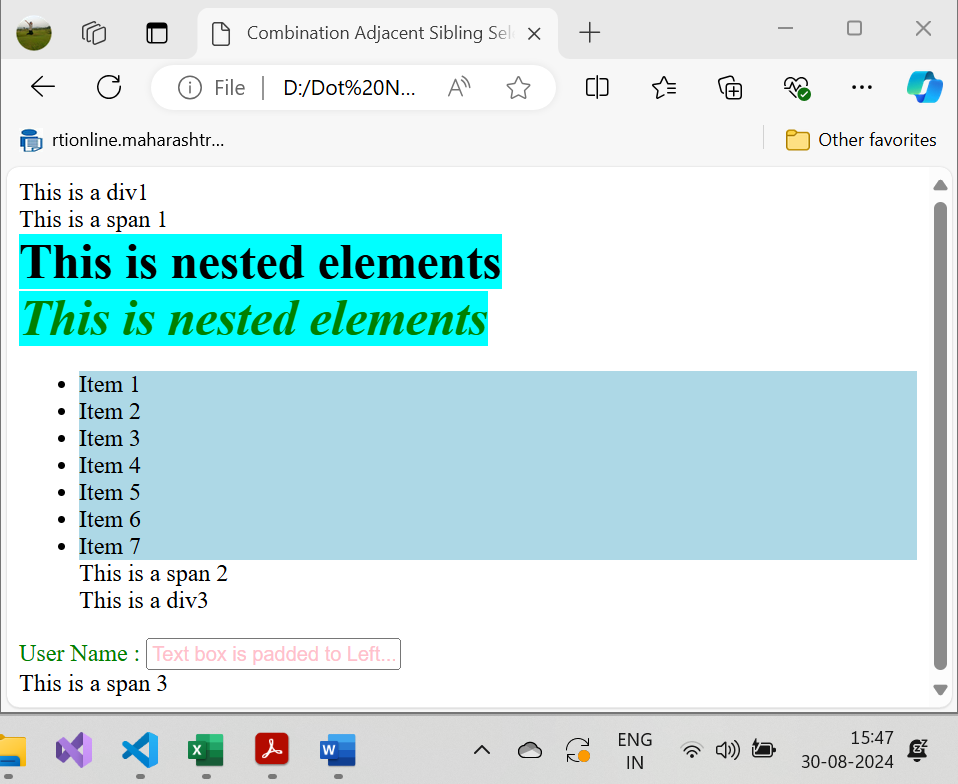
    </div>

    <span>This is a span 3</span></span>

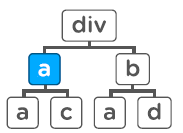
</body>

</html>

Output:



* 1. **Direct Child Selector (div > a):** The child selector selects all elements that are the **Direct** child of a specified element.



**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Combination Direct Child Selectors</title>

    <style>

    /\*Direct Child Selector \*/

    ul > li{

        background: purple;

    }

    span > b{

        background-color: blueviolet;

        font-size: 33px;

    }

    div > .green{

        color: green;

    }

    </style>

</head>

<body>

    <div>This is a div1</div>

    <span>

        <p>

            <b>This is a span 1 Not direct Child</b>

        </p>

    </span>

    <div>

        <span>

            <b>This is nested elements 1</b>

        </span>

    </div>

    <div>

        <span>

            <b><i class="green">This is nested elements 2</i></b>

        </span>

    </div>

    <ul>

        <li>Item 1</li>

        <li>Item 2</li>

        <li>Item 3</li>

        <li>Item 4</li>

        <li>Item 5</li>

        <li>Item 6</li>

        <li>Item 7</li>

        <span>This is a span 2</span></span>

        <div>This is a div3</div>

    </ul>

    <div>

        <label class="green" for="username">User Name :</label>

        <input style="color:pink;" type="text" name="username" id="username" value="Text box is padded to Left....">

    </div>

    <div>

        <H1 class="green">This is H1</H1>

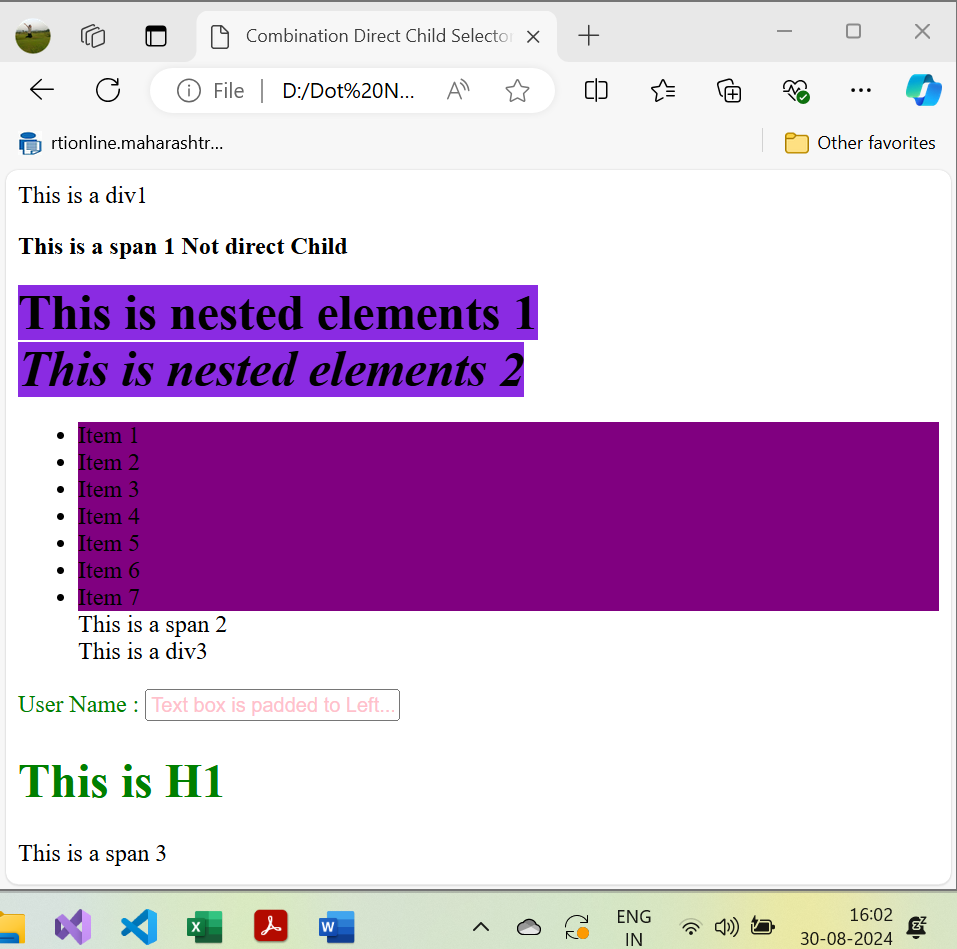
    </div>

    <span>This is a span 3</span></span>

</body>

</html>

**Output:**



* 1. **General Sibling Selector (div ~ a):** Style gets applied to only siblings mentioned after first element. It only selects elements after (i.e to siblings).



Example:

Example combined with Adjacent Siblings

* 1. **Adjacent Sibling Selector (div + a):** Style gets applied to very next sibling only



Example 1:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Combination General & Adjacent Sibling Selectors</title>

    <style>

    /\* General Sibling Selector \*/

    li.green ~li{

        background: green;

    }

    /\* Adjacent Sibling Selector \*/

    li.red + li{

        background: red;

    }

    </style>

</head>

<body>

    <div class="green">This is a div1</div>

    <div>This is a div2</div>

    <span>This is a span 1</span>

    <ul>

        <li>Item 1</li>

        <li class="red">Item 2</li>

        <li>Item 3</li>

        <li>Item 4</li>

        <li class="green">Item 5</li>

        <li>Item 6</li>

        <li>Item 7</li>

        <span>This is a span 2</span></span>

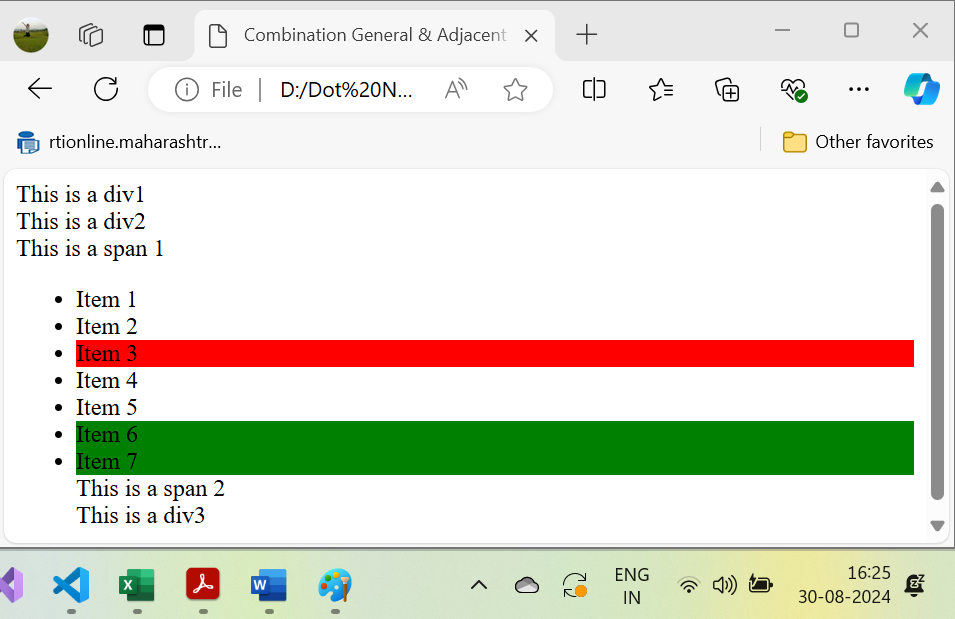
        <div class="green">This is a div3</div>

    </ul>

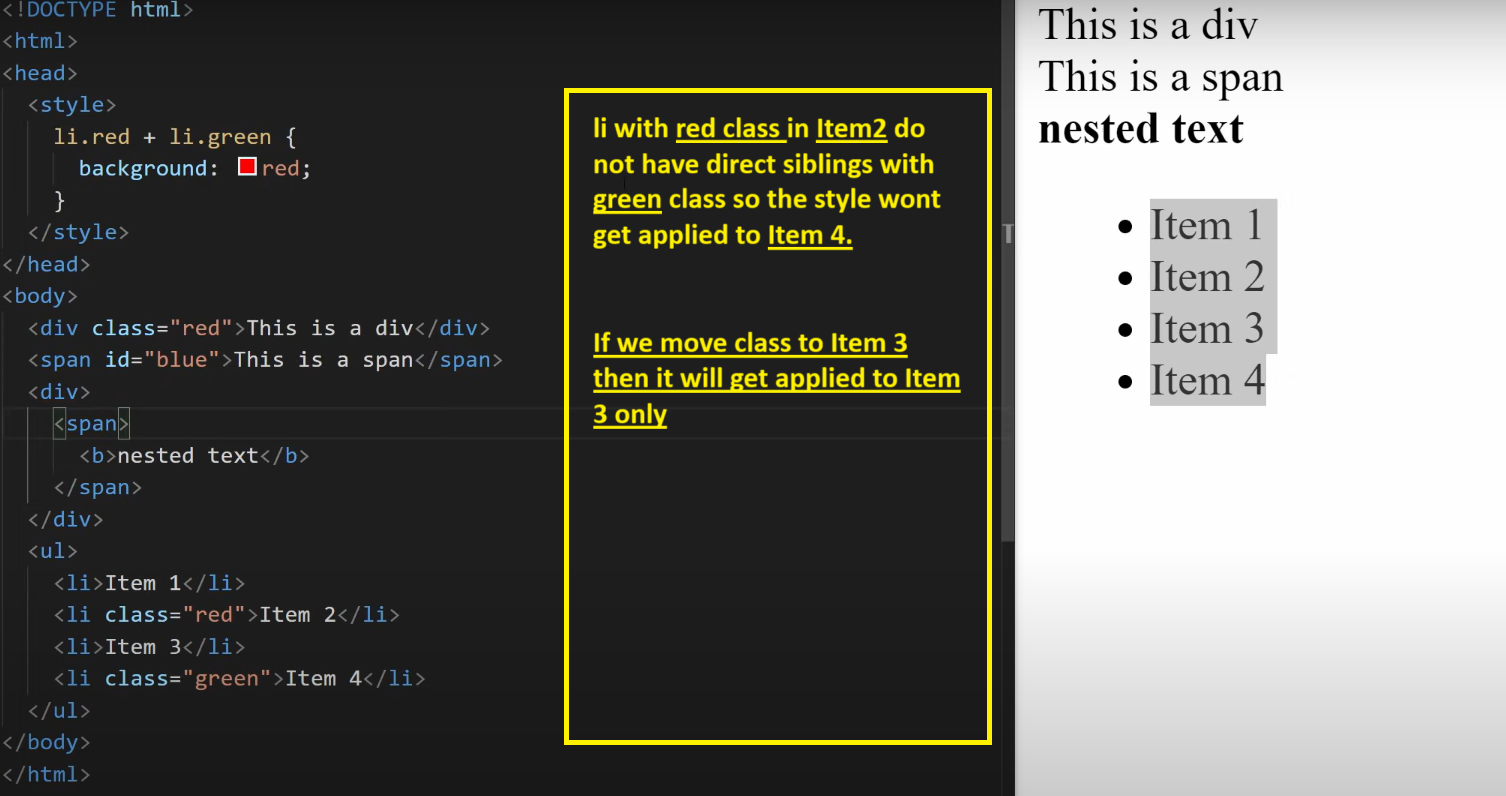
</body>

</html>

Output:



Example 2:



* 1. **Or Selector (div, a)**



Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Combination OR Selectors</title>

    <style>

/\* Or Selector \*/

div, .yellow{

    background: yellow;

}

span, li.red.green-text{

    background: red;

    font-size: 33px;

    color: green;

}

    </style>

</head>

<body>

    <div class="red">This is a div1</div>

    <div>This is a div2</div>

    <span>This is a span 1</span>

    <ul>

        <li>Item 1</li>

        <li class="yellow">Item 2</li>

        <li>Item 3</li>

        <li>Item 4</li>

        <li class="yellow green-text">Item 5</li>

        <li>Item 6</li>

        <li class="red green-text">Item 7</li>

        <span>This is a span 2</span></span>

        <div class="yellow">This is a div3</div>

    </ul>

    <div class="yellow">

        <label for="username">User Name :</label>

        <input type="text" name="username" id="username" value="Text box is padded to Left....">

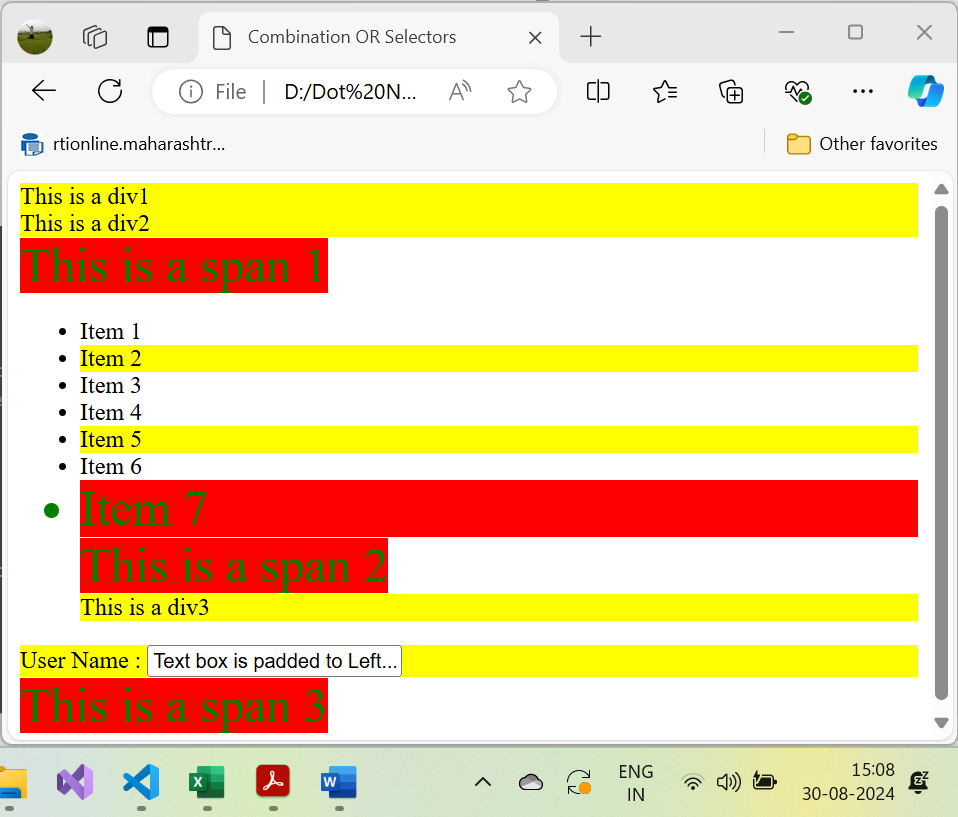
    </div>

    <span>This is a span 3</span></span>

</body>

</html>

Output:



* 1. **And Selector(div.c)**



Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Combination AND Selectors</title>

    <style>

/\* And Selector \*/

div.red{

    background: red;

}

div.red.green-text, span.red.green-text{

    background: green;

    font-size: 33px;

    color: white;

}

    </style>

</head>

<body>

    <div class="red">This is a div1</div>

    <div>This is a div2</div>

    <span>This is a span 1</span>

    <ul>

        <li>Item 1</li>

        <li>Item 2</li>

        <li>Item 3</li>

        <li>Item 4</li>

        <li>Item 5</li>

        <li>Item 6</li>

        <li>Item 7</li>

        <span>This is a span 2</span></span>

        <div class="red green-text">This is a div3</div>

    </ul>

    <div class="red green-text">

        <label for="username">User Name :</label>

        <input style="color:pink;" type="text" name="username" id="username" value="Text box is padded to Left....">

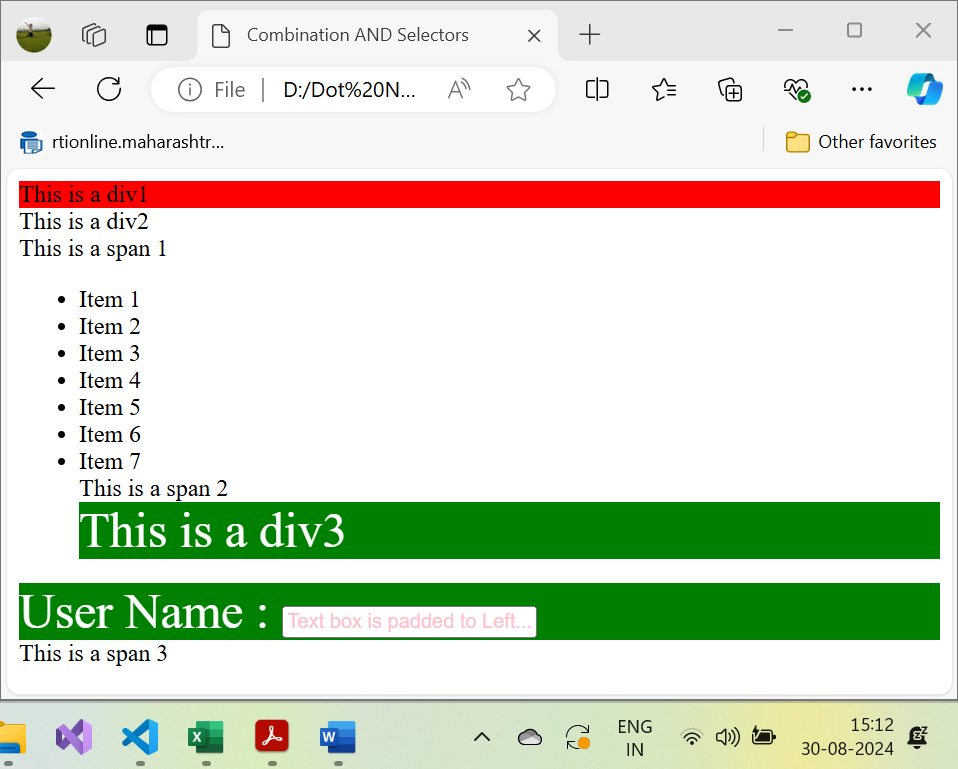
    </div>

    <span>This is a span 3</span></span>

</body>

</html>

Output:



1. Attribute
2. Pseudo Element
3. Pseudo Class State

CSS Pseudo-classes are powerful and allow developers to style elements based on their specific states.

Pseudo-classes in CSS are used to define the special state of an element. They can be combined with a CSS selector to add an effect to existing elements based on their states. For instance, you can change the style of an element when the user hovers over it, or when a link is visited. All of these can be achieved using Pseudo Classes in CSS.

**Note**that pseudo-class names are not case-sensitive.

Most commonly used Pseudo classes states are as below:

* 1. **:hover** : Select elements that are hovered by the mouse.
  2. **:focus** : Select element that is focused. It is either by clicking or by tabbing to an element.
  3. **:required** : Select the element with required attribute.
  4. **:checked** : Select checkboxes / radio buttons that are with checked attribute.
  5. **:disabled** : Select the inputs that are with disabled attribute.
* [:active](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#active)
* [::after/:after](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#after-after)
* [::backdrop (experimental)](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#backdrop-experimental)
* [::before/:before](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#before-before)
* [:checked](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#checked)
* [:default](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#default)
* [:dir (experimental)](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#dir-experimental)
* [:disabled](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#disabled)
* [:empty](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#empty)
* [:enabled](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#enabled)
* [:first-child](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#first-child)
* [::first-letter/:first-letter](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#first-letter-first-letter)
* [::first-line/:first-line](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#first-line-first-line)
* [:first-of-type](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#first-of-type)
* [:focus](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#focus)
* [:fullscreen (experimental)](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#fullscreen-experimental)
* [:hover](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#hover)
* [:in-range](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#span-class-headeranchor-span-in-range)
* [:indeterminate](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#indeterminate)
* [:invalid](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#invalid)
* [:lang](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#lang)
* [:last-child](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#last-child)
* [:last-of-type](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#last-of-type)
* [:link](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#link)
* [:not](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#not)
* [:nth-child](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#nth-child)
* [:nth-last-child](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#nth-last-child)
* [:nth-last-of-type](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#nth-last-of-type)
* [:nth-of-type](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#nth-of-type)
* [:only-child](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#only-child)
* [:only-of-type](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#only-of-type)
* [:optional](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#optional)
* [:out-of-range](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#out-of-range)
* [::placeholder (experimental)](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#placeholder-experimental)
* [:read-only](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#read-only)
* [:read-write](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#read-write)
* [:required](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#required)
* [:root](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#root)
* [::selection](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#selection)
* [:scope (experimental)](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#scope-experimental)
* [:target](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#target)
* [:valid](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#valid)
* [:visited](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#visited)
* [Bonus content: A Sass mixin for links](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#bonus-content-a-sass-mixin-for-links)

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Selectors Pseudo Class State</title>

    <style>

        /\*Selectors Pseudo Class State\*/

        li:hover{

            background: blueviolet;

        }

        a.highlight:hover {

        color: #ff0000;

        font-size: 22px;

        }

        input:focus{

            background: blue;

        }

        input:required{

            background: yellow;

        }

        input:checked{

            margin: 50px;

        }

        input:disabled{

            margin: 50px;

        }

    </style>

</head>

<body>

<ul>

    <li>Item 1</li>

    <li>Item 2</li>

    <li>Item 3</li>

    <li>Item 4</li>

    <li>Item 5</li>

    <li>Item 6</li>

    <li>Item 7</li>

</ul>

<div>

    <label for="txtUsername">User Name : </label>

    <input type="text" id="txtUsername"/>

</div><br>

<div>

    <label for="txtFirstname">First Name : </label>

    <input required type="text" id="txtFirstname"/>

</div>

<div>

    <input checked type="checkbox" id="txtFirstname"/>

</div>

<div>

    <input disabled type="checkbox" id="txtFirstname"/>

</div>

<p>When you hover over the first link below, it will change color and font size:</p>

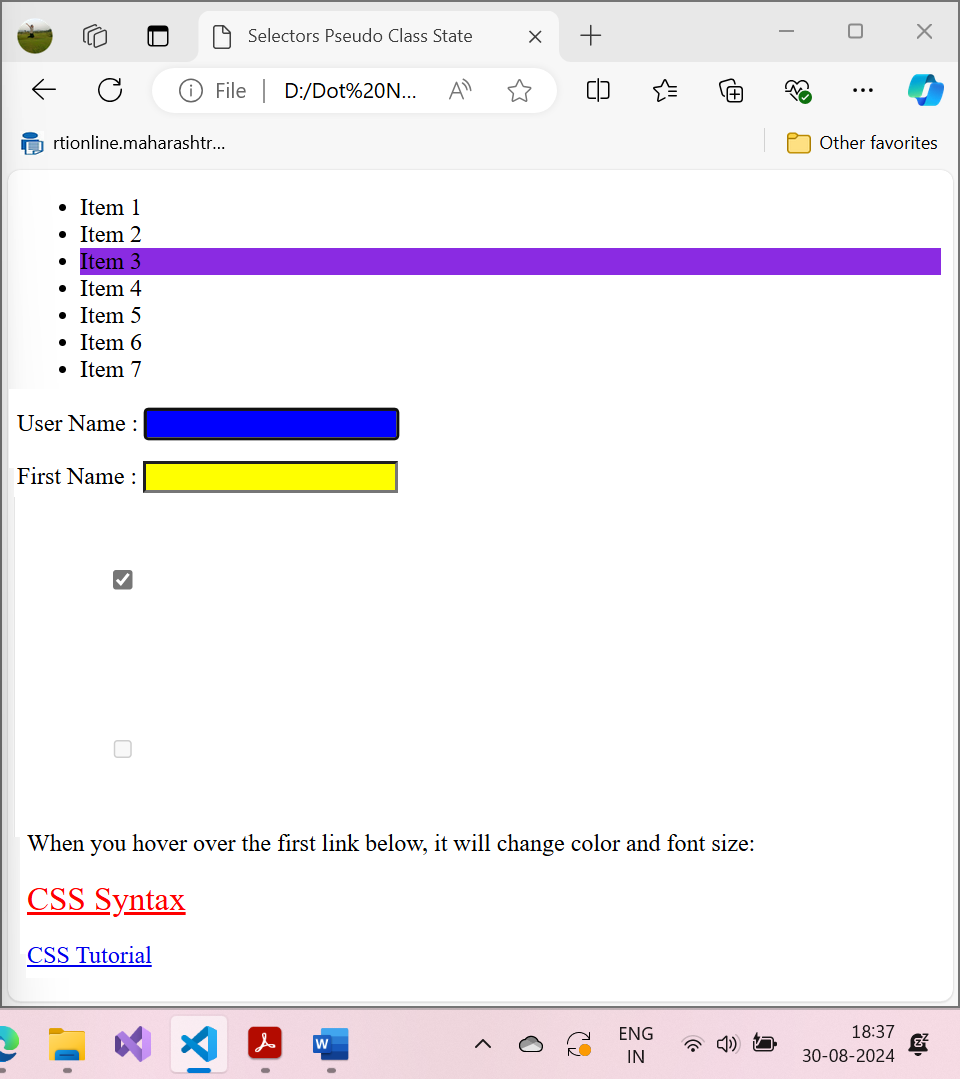
<p><a class="highlight" href="css\_syntax.asp">CSS Syntax</a></p>

<p><a href="default.asp">CSS Tutorial</a></p>

</body>

</html>

Output:



1. Pseudo Class position / other

Example:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Pseudo Class Position</title>

    <style>

        li:first-child{

            background: red;

        }

        li:last-child{

            background: green;

        }

        li:nth-child(5){

            background: yellow;

        }

        li:nth-child(2n){

            background: rgb(111, 0, 255);

        }

        li:nth-child(5n-1){

            background: maroon;

        }

        li:nth-last-child(4n-1){

            background: black;

            color: white;

        }

        span:only-child{

            font-size: 33px;

        }

        p:first-of-type{

            font-style: italic;

        }

    </style>

</head>

<body>

    <span>This is span 1</span>

    <div>

        <span>

            <p>Hello inside span 2</p>

        </span>

    </div>

    <ul>

        <li>Item 1</li>

        <li>Item 2</li>

        <li>Item 3</li>

        <li>Item 4</li>

        <li>Item 5</li>

        <li>Item 6</li>

        <li>Item 7</li>

        <li>Item 8</li>

        <li>Item 9</li>

        <li>Item 10</li>

        <li>Item 11</li>

        <li>Item 12</li>

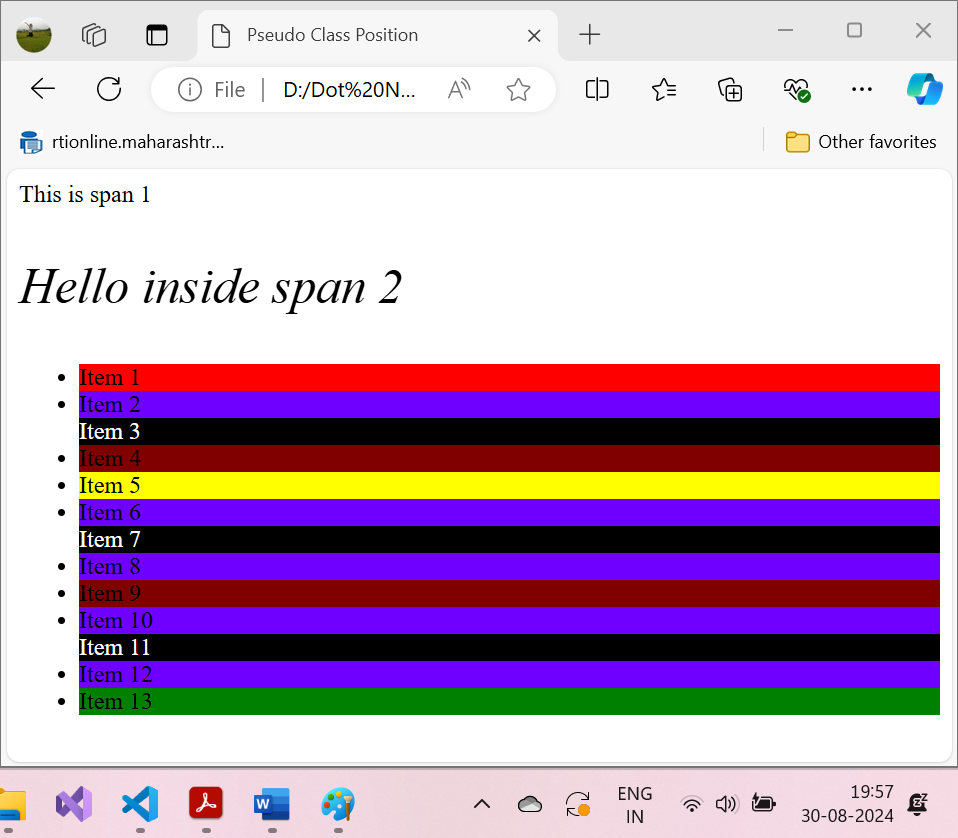
        <li>Item 13</li>

    </ul>

</body>

</html>

Output:



**Other Topics:**

**Q. Priorities of styles that gets applied to element is hight for inline, then internal and least is external, however we can calculate the priority by using below calculations :**

[**https://www.w3schools.com/css/css\_specificity.asp**](https://www.w3schools.com/css/css_specificity.asp)

**How to Calculate Specificity?**

Memorize how to calculate specificity!

Start at 0, add 100 for each ID value, add 10 for each class value (or pseudo-class or attribute selector), add 1 for each element selector or pseudo-element.

Note: Inline style gets a specificity value of 1000, and is always given the highest priority!

Note 2: There is one exception to this rule: if you use the [!important](https://www.w3schools.com/css/css_important.asp) rule, it will even override inline styles!

The table below shows some examples on how to calculate specificity values:

|  |  |  |
| --- | --- | --- |
| **Selector** | **Specificity Value** | **Calculation** |
| **P** | **1** | **1** |
| **p.test** | **11** | **1 + 10** |
| **p#demo** | **101** | **1 + 100** |
| **<p style="color: pink;">** | **1000** | **1000** |
| **#demo** | **100** | **100** |
| **.test** | **10** | **10** |
| **p.test1.test2** | **21** | **1 + 10 + 10** |
| **#navbar p#demo** | **201** | **100 + 1 + 100** |
| **\*** | **0** | **0 (the universal selector is ignored)** |

**The selector with the highest specificity value will win and take effect!**

**Consider these three code fragments:**

**Example**

A: h1  
B: h1#content  
C: <h1 id="content" style="color: pink;">Heading</h1>

The specificity of A is 1 (one element selector)  
The specificity of B is 101 (one ID reference + one element selector)  
The specificity of C is 1000 (inline styling)

Since the third rule (C) has the highest specificity value (1000), this style declaration will be applied.

**Q. Single Or Double Colon For Pseudo-Elements?**[**#**](https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/#single-or-double-colon-for-pseudo-elements)

The short answer is, in most cases, either.

The double colon (::) was introduced in [CSS3](https://www.smashingmagazine.com/learning-css3-useful-reference-guide/) to differentiate pseudo-elements such as ::before and ::after from pseudo-classes such as :hover and :active. All browsers support double colons for pseudo-elements except Internet Explorer (IE) 8 and below.

Some pseudo-elements, such as ::backdrop, accept only a double colon, though.

Personally, I use single-colon notation so that my CSS is backwards-compatible with legacy browsers. I use double-colon notation on those pseudo-elements that require it, of course.

Nesting of CSS code

@layer Directive

@container