Styles with HTML

- Styles are attributes defined for HTML elements to make them more interactive and responsive.
- You can configure styles in 3 ways
  - Inline
  - Embedded
  - External Style Sheet [CSS – Cascade Style Sheets]

Inline Styles

- Inline is a technique where styles are defined for every element individually by using "Style" attribute.
- These styles are not accessible to other elements. You can re-use the styles.
- They are fast in applying as they are native to element.

Ex:

```
<!DOCTYPE html>

<html>

  <head>

    <title>Inline</title>

  </head>

  <body>

    <h2 style="background-color: darkcyan; color:white; text-align: center;">Styles in
HTML</h2>

    <h2>HTML</h2>

  </body>

</html>
```

Embedded Style

- In this technique the styles are embedded into page by using "<Style>" tag.
- You configure in <head> or <body>.
- The styles that you embed in page can be re-used.
- But they are accessible only to elements in current page.
- It is slow in rendering effects when compared to inline.

Ex:

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title>Inline</title>
    <style>
      h2 {
        background-color: red;
        color:white;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <h2>Styles in HTML</h2>
    <h2>HTML</h2>
  </body>
</html>
```

External Style Sheets [Cascade Style Sheets]

- Cascade is a type of Arrangement.
- Styles are maintained in a separate stylesheet with extension ".css"
- So that you can cascade for any HTML page.
- Styles are re-usable across pages.
- Using an external file for your HTML page will increase the number of requests made to page.
- If number of requests to page increases the page load time also increases.

Ex:

- Create a new Folder by name "Styles"
- Add a new File by name "effects.css"
- Add Style attributes into CSS file

```
h2 {
  background-color: red;
  color: white;
  text-align: center;
}
```

- Link the CSS file into your web page by using “<link>” element.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline</title>
    <link rel="stylesheet" href="../Styles/effects.css">
  </head>
  <body>
    <h2>Styles in HTML</h2>
    <h2>HTML</h2>
  </body>
</html>
```

MIME type for CSS

- Browser can understand the file type by using its MIME.
- You have to define the MIME type of CSS as “text/css”.

```
<style type=”text/css”>
</style>
<link rel=”stylesheet” href=””  type=”text/css”>
```

CDN for Style Sheets:

- Content Distribution Network
- It saves the project space.
- The style sheet is maintained in a separate CDN server.
- You can directly link the server URL.
- Issue is, you have to connect to server every time. This will be slow in accessing.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline</title>
    <link type="text/css" rel="stylesheet" href="../Styles/effects.css">
```

```
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">

    </head>

    <body>

        <h2><span class="fa fa-home"></span>Styles in HTML</h2>

        <h2>HTML</h2>

    </body>

</html>
```

Install: **IntelliSense for CSS class names in HTML in VS Code**

Minification

- It is a technique used to compress the CSS files.
- It will reduce the size of file by keeping all your style attribute in shorthand method.
- It reduces the number of lines, remove unnecessary blank space and keeps your code short.
- Always recommend to use the “minified” files for production [live].
- While developing always uses the un-compressed file.

Ex:

- Copy your actual CSS code
- Go to any Minification tool or Site
- Paste your actual code or upload your CSS file
- Click Minify
- Copy the minified code
- Go to your project
- Create a new file “effects.min.css”
- Paste the minified code.
- Link the minified file to HTML page.

```
<link type="text/css" rel="stylesheet" href="../Styles/effects.min.css">
```

FAQ: Where to embed the styles, in <head> or in <body>?

A. If you want to load the styles while loading the page and apply to elements then keep in <body>. If you want to load into browser memory and use later then keep in <head>.

FAQ: If we define styles both inline and embedded then which one will work?

A. Always the priority is given for inline styles.

```
<!DOCTYPE html>

<html>

  <head>

    <title>Inline</title>

    <style>

      h2{

        background-color: red;

      }

    </style>

   </head>

  <body>

    <h2 style="background-color: yellow;">HTML</h2>   // Inline style will apply.

  </body>

</html>
```

FAQ: What is “media” type for Styles?

- You can use “media” attribute for <style> & <link> elements.
- Media type descript the general category of a device, which can be printer, screen, speech etc.
- You can define media values as
  - All
  - Print
  - Screen
  - Speech
- The effects are applied only when the page is used on specific media.

Ex:

```
<!DOCTYPE html>

<html>

  <head>

    <title>Inline</title>

    <style type="text/css" media="print">
```

```
        body{
           border:2px solid darkcyan;
           padding: 20px;
        }
     </style>
   </head>
  <body>
     <h2>HTML</h2>
  </body>
</html>
```

Styles Syntax

- **Inline Syntax**
  <div style="styleProperty:value; styleProperty:value"> </div>
- **Embedded and External Style Sheet**

```
Selector
{
 styleProperty: value;
 styleProperty: value;
}
```

- **Selector** specifies the target location where the given set effects are applied.
- The primary selectors used in styles are:
  - Type Selector
  - ID Selector
  - Class Selector

Type Selector

- It specifies directly the tag name where you want the styles to apply. Like **div, p, span, ol, select, input, form etc.**
- It will apply to every occurrence of that tag in page.
- You can't ignore at any specific location.

Ex:

```
<!DOCTYPE html>

<html>
```

```
    <head>
      <title>Inline</title>
      <style type="text/css">
        h2, p
        {
          background-color: red;
          color: white;
          text-align: center;
        }
      </style>
     </head>
    <body>
      <h2>Web Technologies</h2>
      <p>Para-1</p>
      <h2>HTML</h2>
      <p>Para-2</p>
      <h2>CSS</h2>
      <p>Para-3</p>
      <h2>JavaScript</h2>
      <p>Para-4</p>
    </body>
  </html>
```

ID Selector

- Every element can be defined with “ID” attribute.
- You can define effects only to the element that matches the given ID.
- ID is configured by using following syntax:

  <h2 id=”heading”> </h2>

- ID is accessed by using “#” reference in styles and defined with effects.

Syntax:

```
    <style>
```

```
            #effects {
            }
            </style>
            <div id=”effects”> </div>
```

Ex:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Inline</title>
      <style type="text/css">
        #effects
        {
          background-color: red;
          color: white;
          text-align: center;
        }
      </style>
   </head>
   <body>
      <h2>Web Technologies</h2>
      <p>Para-1</p>
      <h2 id="effects">HTML</h2>
      <p>Para-2</p>
      <h2>CSS</h2>
      <p id="effects">Para-3</p>
      <h2>JavaScript</h2>
      <p>Para-4</p>
   </body>
</html>
```

- Every element can be defined with only one ID.
- If your effects are categorized into various groups with ID selector then you can’t apply all effects to one element.

Class Selector

- Class is defined in style by using “.” Operator.
- Class is accessed and used for element by using “class” attribute.
- Every element can implement multiple classes.
- You can define multiple categories of effects to one element.
- Multiple classes are separated with a blank space.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline</title>
    <style type="text/css">
      .textEffects {
        color:red;
        text-align: center;
      }
      .borderEffects {
        border:5px solid black;
      }
      .backgroundEffects {
        background-color: yellow;
      }
    </style>
   </head>
  <body>
    <h2 class="textEffects borderEffects backgroundEffects">Styles in HTML</h2>
  </body>
```

```
</html>
```

- The CSS selectors are further classified into various groups based on their behaviour.
    - **Combinator / Rational Selectors**
    - Attribute Selectors
    - Pseudo Selectors
    - Structural Pseudo Selectors

Rational or Combinators: These are based on parent and child hierarchy as well as the relationship with other elements. The various combinators are:

Selector	Description

Descendant Selector	Targets all tags under specified parent. It includes any level hierarchy.
	Syntax:
	Parent Child {
	}
	Ex:
	`<!DOCTYPE html>`
	`<html>`
	`  <head>`
	`    <title>Inline</title>`
	`    <style type="text/css">`
	`      ol li {`
	`        color:red;`
	`      }`
	`      div p {`
	`        color:green;`
	`      }`
	`    </style>`
	`  </head>`
	`  <body>`
	`    <h2>Web Technologies</h2>`
	`    <ol>`
	`      <li>HTML`
	`        <ol>`
	`          <li>Void Elements</li>`
	`          <li>Normal Elements</li>`
	`        </ol>`
	`      </li>`
	`      <li>CSS</li>`
	`      <li>JavaScript</li>`
	`    </ol>`
	`    <div>`
	`      <blockquote>Block Quote</blockquote>`
	`      <p>Para-1</p>`

```
    </div>
  </body>
</html>
```

Child Selector

It applies effects only to the direct child of parent element.

Syntax:

```
 Parent > Child {
 }
```

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline</title>
    <style type="text/css">
      tbody > tr {
        background-color: yellow;
      }
      thead > tr > th {
        background-color: lightgreen;
      }
    </style>
   </head>
  <body>
    <table border="1" width="300">
      <thead>
        <tr>
          <th>Name</th>
          <th>Price</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>TV</td>
          <td>45000.55</td>
        </tr>
        <tr>
         <td>Mobile</td>
         <td>11000.55</td>
      </tr>
      </tbody>
    </table>
  </body>
</html>
```

Ex: direct child

```
<!DOCTYPE html>
<html>
  <head>
```

	<title>Inline</title>
	<style type="text/css">
	div > p {
	color:red;
	}
	</style>
	</head>
	<body>
	<div>
	<p>Para-1</p>
	</div>
	<div>
	<span>
	<p>Para-2</p>
	</span>
	</div>
	</body>
	</html>

Adjacent Sibling	It defines effects to an element which is specified immediately after current element.
	It is not parent and child; it is one below another.
	It will apply only to the first adjacent element.
	Syntax:
	A-Element **+** B-Element { }
	
	Ex:
	<!DOCTYPE html>
	<html>
	<head>
	<title>Inline</title>
	<style type="text/css">
	h2+p {
	color:red;
	}
	</style>
	</head>
	<body>
	<h2>HTML Elements</h2>
	<p>Para-1</p>
	<p>Para-2</p>
	<p>Para-3</p>
	<p>Para-4</p>
	</body>
	</html>

General Sibling	It defines effects to all elements which are specified after the current element. Syntax:

	A-Element ~ B-Element { }
	
	Ex:
	<!DOCTYPE html>
	<html>
	  <head>
	    <title>Inline</title>
	    <style type="text/css">
	     h2~p {
	       color:red;
	     }
	    </style>
	   </head>
	  <body>
	   <h2>HTML Elements</h2>
	   <p>Para-1</p>
	   <p>Para-2</p>
	   <p>Para-3</p>
	   <p>Para-4</p>
	  </body>
	</html>

Attribute Selectors:

- Several HTML elements are presented using attribute of a tag.
  <input type=”file”>
- “type” is attribute.
- We have to apply effects for a tag based on its attribute and value.
  Syntax:
  tagName[“attribute”] { }
  tagName[“attributeName=value”] { }

Ex: Uses Attribute and value

```
 <!DOCTYPE html>

<html>

   <head>

      <title>Selectors</title>

      <style>

         input[type=password]{

            background-color: yellow;

         }
```

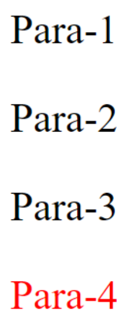
```
        input[type="button"] {
          background-color: lightgreen;
        }
      </style>
    </head>
    <body>
      <form>
        <dl>
          <dt>Name</dt>
          <dd><input type="text"></dd>
          <dt>Password</dt>
          <dd><input type="password"></dd>
        </dl>
        <input type="button" value="Register">
      </form>
    </body>
</html>
```

Ex: Uses only attribute name.

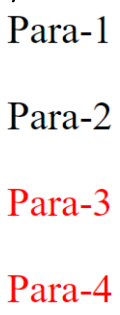
```
         <style>
         p[id] {
           color:red;
        }
      </style>
     <body>
      <p>Para-1</p>
      <p id="p2">Para-2</p>
      <p>Para-3</p>
      <p id="p4">Para-4</p>
         </body>
```

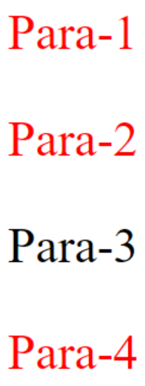
- Attribute selectors can be defined with conditions.
- Effects are applied only to attributes that match the given condition

Condition	Purpose

[attribute=”val”]	Equal specifies that it should be exact match.
	Ex:
	`<!DOCTYPE html>`
	`<html>`
	`  <head>`
	`    <title>Selectors</title>`
	`    <style>`
	`      p[class="Effect"] {`
	`        color:red;`
	`      }`
	`    </style>`
	`  </head>`
	`  <body>`
	`    <p class="paraEffect">Para-1</p>`
	`    <p class="para Effect">Para-2</p>`
	`    <p class="Effectpara">Para-3</p>`
	`    <p class="Effect">Para-4</p>`
	`  </body>`
	`</html>`
	
	Para-1
	Para-2
	Para-3
	Para-4

[attribute^=”val”]	It refers the value starting with specified term.
	Ex:
	`<!DOCTYPE html>`
	`<html>`
	`  <head>`
	`    <title>Selectors</title>`
	`    <style>`
	`      p[class^="Effect"] {`
	`        color:red;`
	`      }`
	`    </style>`
	`  </head>`
	`  <body>`
	`    <p class="paraEffect">Para-1</p>`
	`    <p class="para Effect">Para-2</p>`

	`<p class="Effectpara">Para-3</p>`
	`<p class="Effect">Para-4</p>`
	`</body>`
	`</html>`
	
	Para-1
	Para-2
	Para-3
	Para-4

[attribute$=”val”]	It specifies the value ending with given term.
	Ex:
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`<title>Selectors</title>`
	`<style>`
	`p[class$="Effect"] {`
	`color:red;`
	`}`
	`</style>`
	`</head>`
	`<body>`
	`<p class="paraEffect">Para-1</p>`
	`<p class="para Effect">Para-2</p>`
	`<p class="Effectpara">Para-3</p>`
	`<p class="Effect">Para-4</p>`
	`</body>`
	`</html>`
	
	Para-1
	Para-2
	Para-3
	Para-4

[attribute*=”val”]	It matches at any position.
	Ex:
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`<title>Selectors</title>`

	`<style>`
	`p[class*="Effect"] {`
	`color:red;`
	`}`
	`</style>`
	`</head>`
	`<body>`
	`<p class="paraEffect">Para-1</p>`
	`<p class="para Effect">Para-2</p>`
	`<p class="Effectpara">Para-3</p>`
	`<p class="Effect">Para-4</p>`
	`</body>`
	`</html>`
	Para-1
	
	Para-2
	
	Para-3
	
	Para-4

[attribute|="val"]	Name starts with specified term and separated with "-".
	Ex:
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`<title>Selectors</title>`
	`<style>`
	`p[class|="Effect"] {`
	`color:red;`
	`}`
	`</style>`
	`</head>`
	`<body>`
	`<p class="para-Effect">Para-1</p>`
	`<p class="para Effect">Para-2</p>`
	`<p class="Effect-para">Para-3</p>`
	`<p class="Effect">Para-4</p>`
	`</body>`
	`</html>`

	Para-1 Para-2 Para-3 Para-4

[attribute~="val"]	Name starts with specified term and can contain blank space.
	Name have the term and space occurred at any location.
	Ex:
	`<!DOCTYPE html>`
	`<html>`
	`  <head>`
	`    <title>Selectors</title>`
	`    <style>`
	`      p[class~="Effect"] {`
	`        color:red;`
	`      }`
	`    </style>`
	`  </head>`
	`  <body>`
	`    <p class="para-Effect">Para-1</p>`
	`    <p class="para Effect">Para-2</p>`
	`    <p class="Effect-para">Para-3</p>`
	`    <p class="Effect">Para-4</p>`
	`  </body>`
	`</html>`
	Para-1
	
	Para-2
	
	Para-3
	
	Para-4

Dynamic Pseudo-classes

- Dynamic indicates that the effect can change according to state and situation.
- Pseudo indicates that it is not referring to exactly the element which is having the same name as selector name.
- The selector name and the element it effects may differ.

Selector	Description
:link	Specify effects for Hyperlink.
:visitied	It defines effects for visited links.
:hover	It defines effects when mouse pointer is over element.
:active	It defines effect when link is in active state.
:focus	It defines effects when element gets focus.

Syntax:

```
element:link { }
#heading:hover { }
.txtName:focus { }
```

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Selectors</title>
    <style>
      a:link {
        color:orangered;
        text-decoration: none;
      }
      a:visited {
        color: greenyellow;
      }
      a:active {
        color:yellow;
      }
      a:hover {
        text-decoration: underline;
      }
      img {
        width: 100px;
        height: 100px;
        transition: 2s;
      }
      img:hover {
        width: 200px;
```

```
                height: 200px;
                transition: 2s;
            }
            input:focus {
                border:2px solid red;
                box-shadow: 2px 2px 3px red;
            }
            input~span {
                display:none;
            }
            input:focus~span{
                display: inline;
            }
        </style>
    </head>
    <body>
      <div>
         <label>Name</label>
         <div>
            <input type="text">
            <span>Name 4 chars</span>
         </div>
      </div>
      <div>
         <img src="../Images/shirt.jpg">
      </div>
       <div>
          <a href="home.html">Home</a>
          <a href="shopping.html">Shopping</a>
          <a href="http://amazon.in">Amazon</a>
       </div>
    </body>
</html>
```

Target pseudo class

Selector	Description
:target	- It defines effect to element when it is accessed as target on link click. - You can define an ID for element and access by using ID reference. - It is mostly used while working with “intra-document” navigation.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Target Selector</title>
    <style>
      .topic {
        background-color: lightgreen;
        color: black;
        margin-top: 20px;
        border:2px solid darkgreen;
        padding:10px;
      }
      ul {
        list-style:none;
        display: flex;
      }
      li {
        margin-left: 50px;
        border:2px solid darkblue;
        padding:10px;
        width: 200px;
        text-align: center;
        font-size: 20px;
        background-color: darkblue;
        color:white;
      }
      a:link {
        text-decoration: none;
        color:white;
        }
        a:visited {
            color: white;
        }
        .topic:target {
            background-color: black;
            color:white;
        }
    </style>
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="#html">HTML</a></li>
                <li><a href="#css">CSS</a></li>
                <li><a href="#js">JavaScript</a></li>
            </ul>
        </nav>
    </header>
    <section>
        <div id="html" class="topic">
            <h3>HTML</h3>
            <p>It is a markup language.</p>
        </div>
        <div id="css" class="topic">
            <h3>CSS</h3>
            <p>It defines styles for HTML elements.</p>
        </div>
```

```
          <div id="js" class="topic">
              <h3>JavaScript</h3>
              <p>It is a language used to reduce burden on server.</p>
          </div>
      </section>
   </body>
</html>
```

The UI element states pseudo-classes

Selector	Description
:enabled	It defines effects when element is enabled.
:disabled	It defines effects when element is disabled.
:read-only	It defines effects when element is specified read-only.
:checked	It defines effects when radio or check box are checked.

Ex:1

```
<!DOCTYPE html>
<html>
  <head>
      <title>State selectors</title>
      <style>
         button:disabled{
           background-color: darkgrey;
           color:white;
           cursor:not-allowed;
         }
         button:enabled {
            background-color: green;
            color:white;
            cursor: grab;
         }
```

```
            input:read-only{
                background-color: yellow;
                color:gray;
            }
        </style>
    </head>
    <body>
        <fieldset>
            <legend>User Name</legend>
            <input value="John" readonly type="text">
            <button >Submit</button>
        </fieldset>
    </body>
</html>
```

Ex2: Checked

```
<!DOCTYPE html>
<html>
    <head>
        <title>Checked</title>
        <style>
            textarea:read-only {
                background-color: lightgray;
            }
            input[type="checkbox"]:checked+span {
                color: green;
            }
            input[type="checkbox"]+span {
                color:red;
```

```
      }
    </style>
  </head>
  <body>
    <fieldset>
      <legend>Terms of Service</legend>
      <textarea cols="40" rows="5" readonly>
        Depending on how you obtained the Windows software, this is a license agreement
between (i) you and the device manufacturer or software installer that distributes the
software with your device; or (ii) you and Microsoft Corporation (or, based on where you
live or, if a business, where your principal place of business is located, one of its affiliates) if
you acquired the software from a retailer. Microsoft is the device manufacturer for devices
produced by Microsoft or one of its affiliates, and Microsoft is the retailer if you acquired
the software directly from Microsoft. Note that if you are a volume license customer, use of
this software is subject to your volume license agreement rather than this agreement.
      </textarea>
      <div>
        <input type="checkbox"> <span>I Accept</span>
      </div>
    </fieldset>
  </body>
</html>
```

The UI element validation state pseudo classes

Selector	Description
:valid	It checks the validation state for HTML element and defines effects if it is valid. Validation is verified by using: - Minlength - Required - Pattern - Maxlength - Email etc.
:invalid	It checks the validation state for HTML element and defines effects if it is invalid.
:required	It defined effects when input element is marked with “required” attribute.

:optional	It defines effects for input element when not marked with required.
:in-range	It defines effects for element when input value is within the specified range.
:out-of-range	It defines effects for element when input value is out of given range.

Ex: Required and Optional

```
<!DOCTYPE html>

<html>

  <head>

    <title>Required</title>

    <style>

      .form-group {

        margin-top: 20px;

      }

      label {

        font-weight: bold;

      }

      input:required {

        border:1px solid red;

        box-shadow: 2px 2px 4px red;

      }

      input:optional {

        border:1px solid goldenrod;

        box-shadow: 2px 2px 4px gold;

      }

    </style>

  </head>

  <body>

    <form>
```

```
            <div class="form-group">
                <label>User Name</label>
                <div>
                    <input type="text" required>
                </div>
            </div>
            <div class="form-group">
                <label>Password</label>
                <div>
                    <input type="password">
                </div>
            </div>
        </form>
    </body>
</html>
```

Ex: Valid and Invalid

```
<!DOCTYPE html>
<html>
    <head>
        <title>Required</title>
        <style>
            .form-group {
                margin-top: 20px;
            }
            label {
                font-weight: bold;
            }
            input:invalid {
                border:1px solid red;
                box-shadow: 2px 2px 4px red;
            }
            input:valid {
                border:1px solid green;
                box-shadow: 2px 2px 4px green;
            }
        </style>
    </head>
    <body>
        <form>
            <div class="form-group">
                <label>User Name</label>
                <div>
                    <input type="text" required>
                </div>
            </div>
            <div class="form-group">
                <label>Password</label>
                <div>
                    <input type="password" required minlength="4">
                </div>
            </div>
            <div class="form-group">
                <label>Email</label>
                <div>
                    <input type="email" required>
                </div>
            </div>
        </form>
```

```
 </body>
</html>
```

Ex: In-range and out-of-range

```
<!DOCTYPE html>
<html>
  <head>
    <title>Required</title>
    <style>
      .form-group {
        margin-top: 20px;
      }
      label {
        font-weight: bold;
      }
      input:in-range {
        border:2px solid green;
        box-shadow: 2px 2px 3px green;
      }
      input:out-of-range {
        border:2px solid red;
        box-shadow: 2px 2px 3px red;
      }
    </style>
  </head>
  <body>
    <form>
      <div class="form-group">
        <label>Age</label>
        <div>
```

```
                <input type="number" min="16" max="35">
            </div>
        </div>
    </form>
  </body>
</html>
```

Ex: **Display message for valid and invalid states**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Required</title>
    <style>
      .form-group {
        margin-top: 20px;
      }
      label {
        font-weight: bold;
      }
      input:in-range {
        border:2px solid green;
        box-shadow: 2px 2px 3px green;
      }
      input:out-of-range {
        border:2px solid red;
        box-shadow: 2px 2px 3px red;
      }
      input:in-range+span {
        display: none;
      }
```

```
        input:out-of-range+span {
            display: inline;
        }
    </style>
</head>
<body>
    <form>
        <div class="form-group">
            <label>Age</label>
            <div>
                <input type="number" min="16" max="35">
                <span style="color: red;">Age 16 to 35 only</span>
            </div>
        </div>
    </form>
</body>
</html>
```

Structural pseudo-classes

- You can target your effects based on the position of element in parent and child hierarchy.

Selector	Description
:first-child	It defines effects only for first child element under any specific parent element.
:last-child	It defines effects only for last child element under any specific parent element.
:nth-child(number)	It defines effects only for any specific child element under any specific parent element based on occurrence number. Ex: <!DOCTYPE html> <html> <head>

	<title>Structural Selectors</title>
	<style>
	li:first-child {
	color:red;
	}
	li:last-child {
	color:green;
	}
	li:nth-child(3) {
	color:blue;
	}
	</style>
	</head>
	<body>
	<ol>
	<li>Item-1</li>
	<li>Item-2</li>
	<li>Item-3</li>
	<li>Item-4</li>
	<li>Item-5</li>
	</ol>
	</body>
	</html>
	Note: You can also use pre-defined occurrence options like “even and odd”.

:nth-of-type(occurance, n)	It will repeat for every nth occurrence.
	Syntax:
	:nth-of-type(2n)
	{
	}
	It will apply effect for every 2nd occurrence. It will start with 2nd element.
	
	Syntax:
	:nth-of-type(2n+1)
	{
	}
	It will apply effect for every 2nd occurrence. It will start with 1st element. (2n+1 – starting numbering)
	
	Ex:
	<!DOCTYPE html>
	<html>
	<head>
	<title>Structural Selectors</title>
	<style>
	li:nth-of-type(2n+1){

	color:red;
	}
	</style>
	</head>
	<body>
	<ol>
	<li>Item-1</li>
	<li>Item-2</li>
	<li>Item-3</li>
	<li>Item-4</li>
	<li>Item-5</li>
	</ol>
	</body>
	</html>
:nth-last-of-type(n)	It will apply effect for every nth occurrence from bottom. [bottom to top]
	Ex:
	<style>
	li:nth-last-of-type(2n){
	color:red;
	}
	</style>
:nth-last-child(n)	Defines effect for nth reference child element from bottom.
	It is not repeating, it will apply from bottom.
	Ex:
	<style>
	li:nth-last-child(2){
	color:red;
	}
	</style>
:root	It refers to root of document, which is "body".
	Ex:
	:root {
	font-family: Arial;
	}
:empty	If any element is empty without any content then it will define the given effects. You can configure empty selector only for containers can support presenting of text like, <td>, <th>, <dd> etc.

Ex: Even and Odd occurrence

```
<!DOCTYPE html>

<html>
```

```
<head>
    <title>Structural Selectors</title>
    <style>
      thead {
          background-color: brown;
          color:white;
      }
      tbody > tr:nth-child(odd){
         background-color: lightgreen;
      }
      tbody > tr:nth-child(even){
          background-color: lightpink;
      }
    </style>
</head>
<body>
   <table border="1" width="300">
       <thead>
           <tr>
               <th>Name</th>
               <th>Price</th>
           </tr>
       </thead>
       <tbody>
           <tr>
               <td>TV</td>
               <td>45000.55</td>
           </tr>
           <tr>
```

```
            <td>mobile</td>

            <td>15000.55</td>

          </tr>

          <tr>

            <td>Shoe</td>

            <td>45000.55</td>

          </tr>

          <tr>

            <td>Watch</td>

            <td>45000.55</td>

          </tr>

          <tr>

            <td>Shirt</td>

            <td>5000.55</td>

          </tr>

        </tbody>

      </table>

   </body>

</html>
```

Pseudo-element Selectors

Selector	Description
::first-line	Defines effects for first line in a paragraph or container.
::first-letter	Effects the first letter in container.
::before	Specifies the effect or content to add before current element.
::after	Specifies the effect or content to add after current element.

Ex:

```
<!DOCTYPE html>

<html>

   <head>

       <title>Pseudo Selectors</title>
```

```
        <style>
          p::first-letter{
            font-size: large;
            font-family: Arial;
            color:blue;
            font-weight: bold;
          }
          p::first-line {
            color:blue;
          }
        </style>
    </head>
    <body>
        <p>Depending on how you obtained the Windows software, this is a license agreement
between (i) you and the device manufacturer or software installer that distributes the
software with your device; or (ii) you and Microsoft Corporation (or, based on where you
live or, if a business, where your principal place of business is located, one of its affiliates) if
you acquired the software from a retailer. Microsoft is the device manufacturer for devices
produced by Microsoft or one of its affiliates, and Microsoft is the retailer if you acquired
the software directly from Microsoft.</p>
    </body>
</html>
```

Ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Pseudo Selectors</title>
        <style>
         ul {
           display: flex;
```

```
            list-style: none;
          }
          li::after {
            content: '/';

          }
          li:last-child::after {
            content: '';
          }
        </style>
    </head>
    <body>
        <ul>
          <li>Home</li>
          <li>About</li>
          <li>Contact</li>
        </ul>
    </body>
</html>
```

Language Selector:

- It is defined by using “:lang()”
- It can apply effects to content based on language defined.
- When your page is multi lingual then you can define effects to content based on specific language.

Ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Pseudo Selectors</title>
        <style>
            p:lang(en) {
                font-style: italic;
            }
        </style>
    </head>
    <body>
     <h2> Language Selector </h2>
     <p>Some text</p>
     <p lang="en">English US</p>
    </body>
</html>
```

Negation Selector

- It is used to define effects for the elements which are not matching with specified criteria.
- The negation selector is “:not()”
- It will ignore effects for specific element and can apply to others.

Ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Negation</title>
        <style>
            p:not(#effects) {
                color:red;
            }
        </style>
    </head>
    <body>
        <p>Para-1</p>
        <p id="effects">Para-2</p>
        <p>Para-3</p>
        <p>Para-4</p>
    </body>
</html>
```

Ex:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Negation</title>
        <style>
           input:not([disabled]) {
               background-color: yellow;
           }
        </style>
    </head>
    <body>
        <dl>
            <dt>Name</dt>
            <dd><input type="text"></dd>
            <dt>Password</dt>
            <dd><input disabled type="password"></dd>
            <dt>Age</dt>
            <dd><input type="number"></dd>
        </dl>
    </body>
</html>
```

Universal Selector

- It is defined by using “*” that represents all.
- It applies effects to all elements.

Syntax:

```
      *{
        font-family: Arial;
     }
```