



ABES
Engineering College
College Code 032

Web Development Training

Mr. Amit Goel

Assistant Professor

Department of Computer Applications



ABES
Engineering College
College Code 032

Front End Development

Module 2 - CSS

Agenda

- Introduction to CSS
- Types of CSS
- Working with Selectors
- Cascading Order
- Understanding Box Model
- CSS Layout
- Specificity
- CSS Transition and Animation
- Website Design Fundamentals and Typography
- CSS Best Practices

What is CSS?

- CSS stands for cascading style sheets.
- It is a design language that makes a website look more appealing than just plain or uninspiring pieces of text.
- Whereas HTML largely determines textual content, CSS determines visual structure, layout, and aesthetics.
- HTML is a markup language, and CSS is a style sheet language.
- Think “look and feel” when you think CSS.

Benefits of using CSS

- Faster Page Speed
- Better User Experience
- Quicker Development Time
- Easy Formatting Changes
- Compatibility Across Devices

How to use the CSS?

- CSS provides a big set of pre-defined properties to apply the styling on HTML contents using **key : value** pair model.
- Each **key : pair** is separated by semicolon
 - **color** : 'color name or code'
 - **background-color** : 'color name or code'
 - **text-align** : 'left | right | center | justify'
 - **font-size** : 'n'
 - **font-weight** : 'bold'
 - **font-family** : 'font name'
 - **text-transform** : 'uppercase | lowercase | capitalize'
 - **text-decoration** : 'underline | overline | line-through | none'
 - **display** : 'none | block | inline'




Different CSS ?

- CSS can be applied using **three** ways
 - ☐ **Inline CSS**
 - ☐ **Internal CSS**
 - ☐ **External CSS**

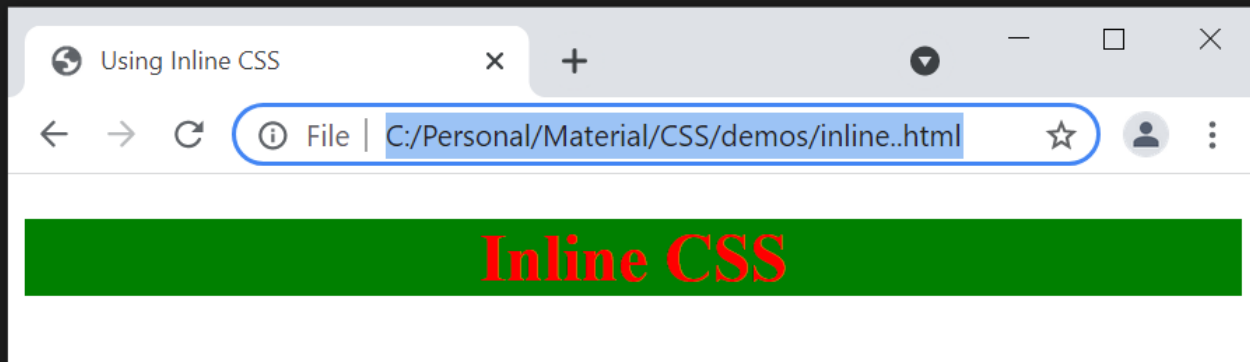
Inline CSS?

- When the CSS codes get applied directly on a tag using **STYLE** attribute, it is known as inline CSS
- Use the codes using **key : value** pair model
- Use **semicolon (;)** when as separator when using multiple codes
- Just like HTML, CSS is also case insensitive






<> inline..html >  html >  body >  h1

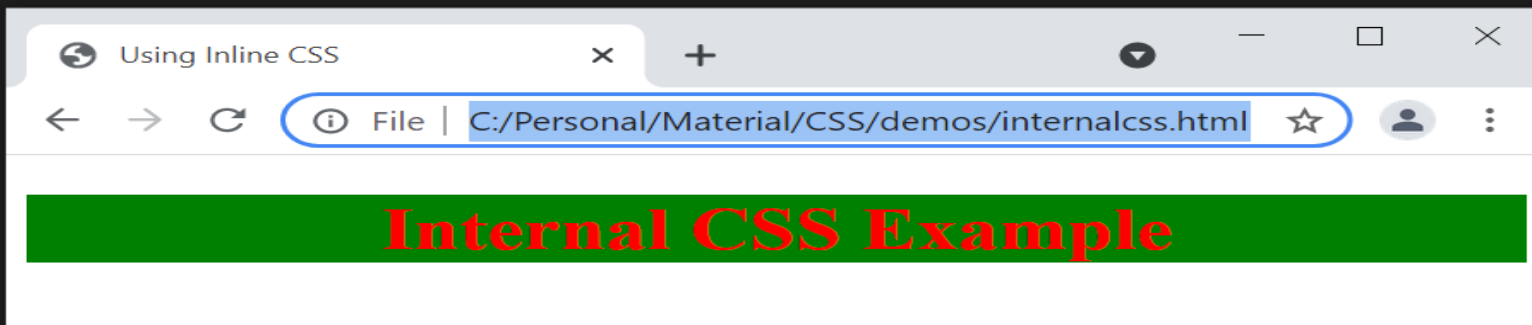
```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Using Inline CSS</title>
5      </head>
6      <body>
7          <h1 style='color: red;background-color: green;text-align:center'>Inline CSS</h1>
8      </body>
9  </html>
```



Internal CSS?

- When we apply styling on all or selected set of elements in a web page, it is called as internal CSS
- Use `<style> </style>` tag to create such styles




```
<> internalcss.html >  html
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Using Inline CSS</title>
5          <style>
6              h1 {color:  red;background-color:  green;text-align:center}
7          </style>
8      </head>
9      <body>
10         <h1>Internal CSS Example</h1>
11     </body>
12 </html>
```



How to define styling for specific element?

- To define styling for specific element in the web page, first we need to define an identity to the element using **ID** attribute
- Use **#idname** while defining styling for the element inside `<style></style>` tag



```
<> idselector.html >  html
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Using Inline CSS</title>
5          <style>
6              #main {color:  red;background-color:  green;text-align:center}
7          </style>
8      </head>
9      <body>
10         <h1>Internal CSS Example</h1>
11         <h1 id='main'>ID Test</h1>
12     </body>
13 </html>
```



How to define styling for group of elements?

- To define styling for group of elements in the web page, first we need to classify them using **CLASS** attribute
- Use **.classname** while defining styling for the group of elements inside `<style></style>` tag

<> classselector.html > html > body > p.x

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Using Internal CSS</title>
5          <style>
6              .x{color: red}
7          </style>
8      </head>
9      <body>
10         <h1>Internal CSS Example</h1>
11         <h1 class='x'>ID Test</h1>
12         <p class="x">Hello India</p>
13     </body>
14 </html>
```



What is external CSS?

- We create a **separate file** having styling effects and use that file in all the web pages of a website or even in multiple websites
- Such method of using styling is called as **external CSS**
- Here we need to use **<link> tag** to define the external CSS file name

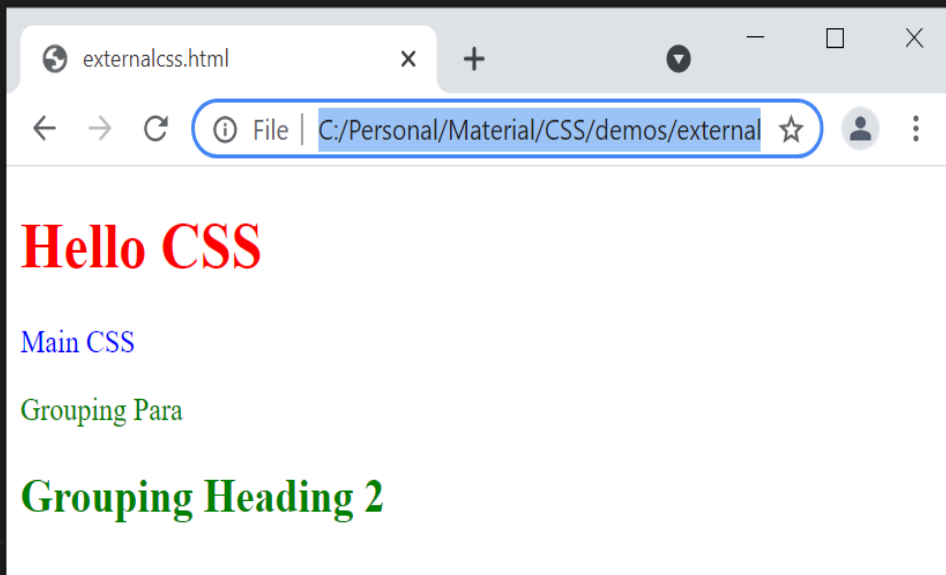
<link href='filename.css' rel='stylesheet'>



```
# styles.css > ...  
1    h1 {color: red}  
2    #main {color: blue}  
3    .x {color: green}
```

<> externalcss.html > html

```
1  <!DOCTYPE html>  
2  <html>  
3      <head>  
4          <link href="styles.css" rel="stylesheet">  
5      </head>  
6      <body>  
7          <h1>Hello CSS</h1>  
8          <p id='main'>Main CSS</p>  
9          <p class="x">Grouping Para</p>  
10         <h2 class="x">Grouping Heading 2</h2>  
11     </body>  
12 </html>
```



What are selectors?

- CSS selectors are used to "find" (or select) the HTML elements you want to style.
- CSS selectors are divided into five categories
 - ☐ Simple selectors
 - ☐ Combinator selectors
 - ☐ Pseudo-class selectors
 - ☐ Pseudo-elements selectors
 - ☐ Attribute

What are the simple selectors?

- Universal Selector
- Element Selector
- Grouping Selector
- ID Selector
- CLASS Selector

Simple Selectors

- The **universal selector** (*) selects all HTML elements on the page.
- The **element selector** selects HTML elements based on the element name
- The **grouping selector** selects all the HTML elements with the same style definitions.
- The **id selector** uses the id attribute of an HTML element to select a specific element.
- The **class selector** selects HTML elements with a specific class attribute.

Examples of Simple Selectors

Universal Selector

```
* {  
  text-align: center;  
  color: blue;  
}
```

Element Selector

```
p {  
  text-align: center;  
  color: red;  
}
```

Grouping Selector

```
h1, h2, p {  
  text-align: center;  
  color: red;  
}
```

ID Selector

```
#para1 {  
  text-align: center;  
  color: red;  
}
```

Class Selector

```
.center {  
  text-align: center;  
  color: red;  
}
```

Combinator Selectors


- A combinator is something that explains the **relationship** between the selectors.
- There are four different combinator selectors in CSS
 - ☐ descendant selector (space)
 - ☐ child selector (>)
 - ☐ adjacent sibling selector (+)
 - ☐ general sibling selector (~)

Descendant Selector

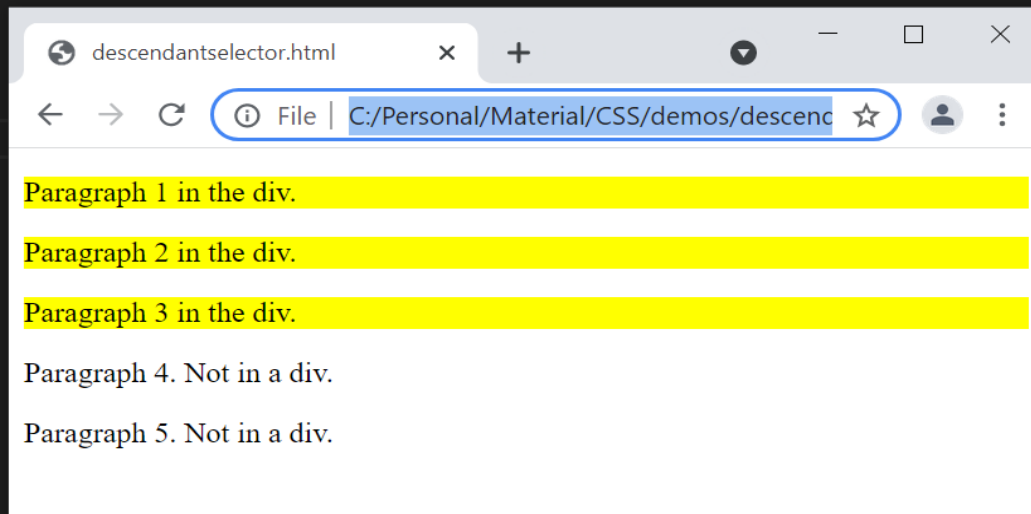
- The descendant selector matches all elements that are descendants of a specified element
- The following example selects all <p> elements inside <div> elements

```
div p {  
    background-color: yellow;  
}
```



<> descendantselector.html >  html

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <style>
5        |   div p {
6        |   |   |   background-color: yellow;
7        |   }
8      </style>
9    </head>
10   <body>
11     <div>
12       <p>Paragraph 1 in the div.</p>
13       <p>Paragraph 2 in the div.</p>
14       <section>
15         |   <p>Paragraph 3 in the div.</p>
16         </section>
17     </div>
18     <p>Paragraph 4. Not in a div.</p>
19     <p>Paragraph 5. Not in a div.</p>
20   </body>
21 </html>
```



Child Selector (>)

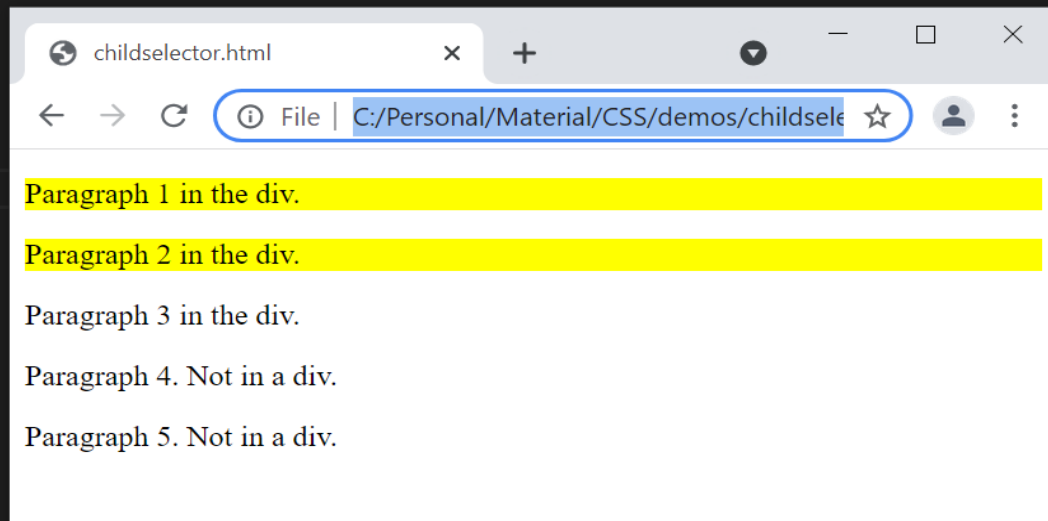
- The child selector selects all elements that are the children of a specified element.
- The following example selects all <p> elements that are children of a <div> element

```
div > p {  
    background-color: yellow;  
}
```



<> childselector.html > html > head > style

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <style>
5        div > p {
6          background-color: yellow;
7        }
8      </style>
9    </head>
10   <body>
11     <div>
12       <p>Paragraph 1 in the div.</p>
13       <p>Paragraph 2 in the div.</p>
14       <section>
15         <p>Paragraph 3 in the div.</p>
16       </section>
17     </div>
18     <p>Paragraph 4. Not in a div.</p>
19     <p>Paragraph 5. Not in a div.</p>
20   </body>
21 </html>
```



Adjacent Sibling Selector (+)

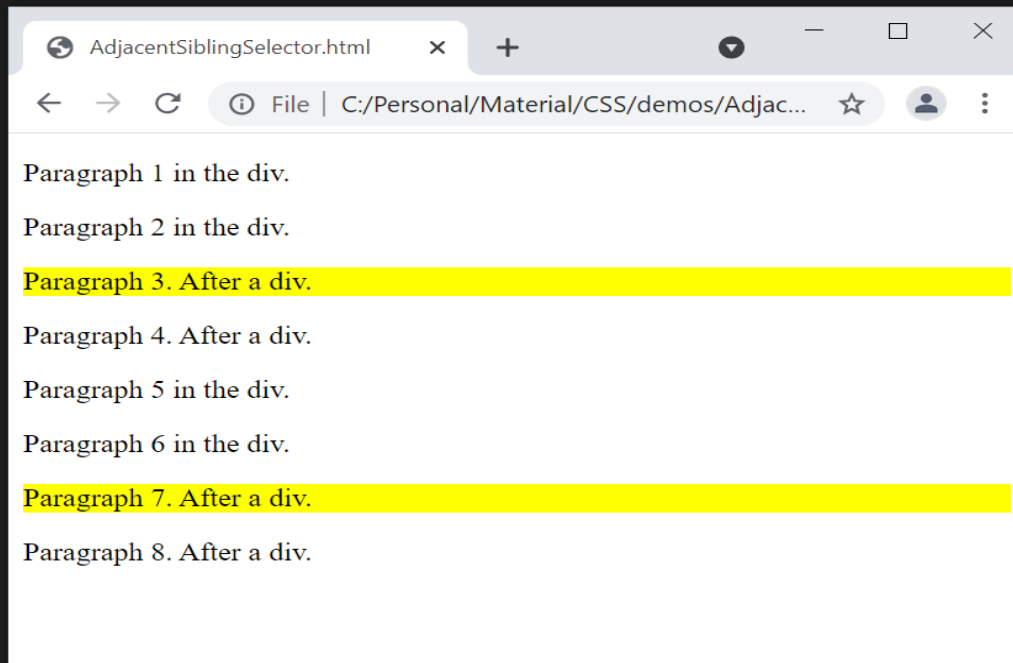
- The adjacent sibling selector is used to select an element that is directly after another specific element
- Sibling elements must have the same parent element, and "adjacent" means "immediately following"
- The following example selects the first <p> element that are placed immediately after <div> elements

```
div + p {  
    background-color: yellow;  
}
```



<> AdjacentSiblingSelector.html > html > body > p

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <style>
5              div + p {
6                  background-color: yellow;
7              }
8          </style>
9      </head>
10     <body>
11         <div>
12             <p>Paragraph 1 in the div.</p>
13             <p>Paragraph 2 in the div.</p>
14         </div>
15         <p>Paragraph 3. After a div.</p>
16         <p>Paragraph 4. After a div.</p>
17         <div>
18             <p>Paragraph 5 in the div.</p>
19             <p>Paragraph 6 in the div.</p>
20         </div>
21         <p>Paragraph 7. After a div.</p>
22         <p>Paragraph 8. After a div.</p>
23     </body>
24 </html>
```



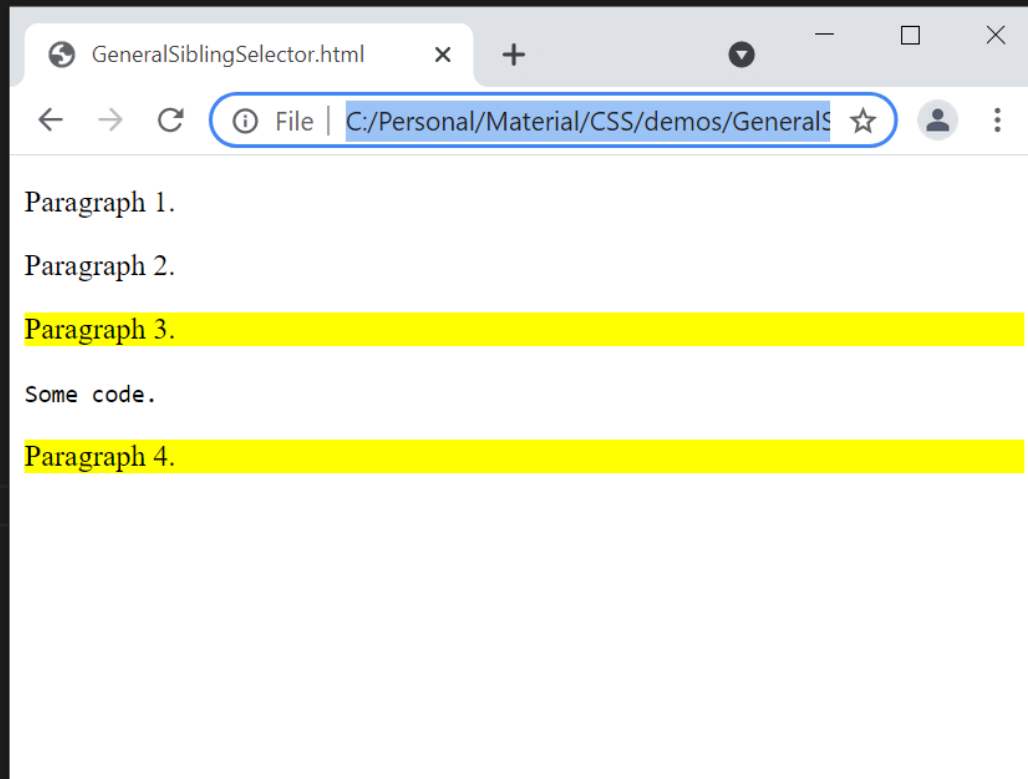
General Sibling Selector (~)

- The general sibling selector selects all elements that are next siblings of a specified element
- The following example selects all <p> elements that are next siblings of <div> elements

```
div ~ p {  
    background-color: yellow;  
}
```



```
<> GeneralSiblingSelector.html > html > body > div
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <style>
5              div ~ p {
6                  background-color: yellow;
7              }
8          </style>
9      </head>
10     <body>
11         <p>Paragraph 1.</p>
12         <div>
13             <p>Paragraph 2.</p>
14         </div>
15         <p>Paragraph 3.</p>
16         <code>Some code.</code>
17         <p>Paragraph 4.</p>
18     </body>
19 </html>
20
```



What are Pseudo-classes?

- Special classes that are used to define some action on specific state of an element e.g.
 - Style an element when it gets the focus
 - Style an element when a user mouse over it
 - Style an element when a link is visited
- Syntax

```
selector:pseudo-class {  
    property: value;  
}
```

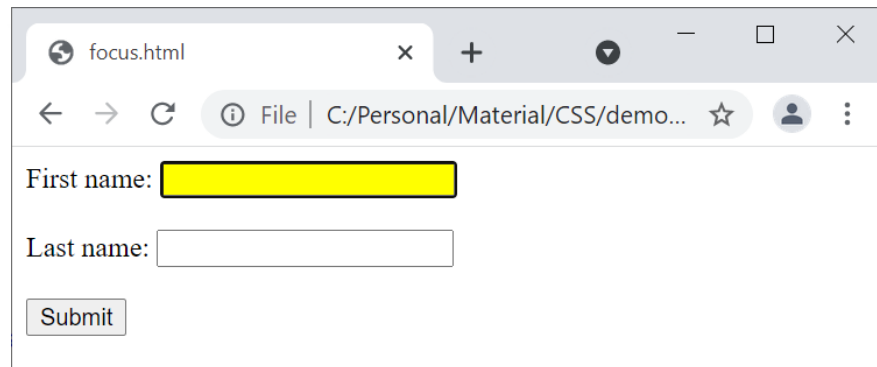
Partial list of pseudo-classes

- ☐ :focus
- ☐ :hover
- ☐ :link
- ☐ :alink
- ☐ :vlink
- ☐ :checked
- ☐ :enabled
- ☐ :disabled
- ☐ :first-child
- ☐ :last-child
- ☐ :nth-child(n)

Example of focus pseudo code

<> focus.html > ...

```
1  <style>
2  input:focus {
3  |   background-color: yellow;
4  }
5  </style>
6  <form>
7  |   First name: <input type="text" name="fname"><br><br>
8  |   Last name: <input type="text" name="lname"><br><br>
9  |   <input type="submit" value="Submit">
10 </form>
```



focus.html

File | C:/Personal/Material/CSS/demo... ☆

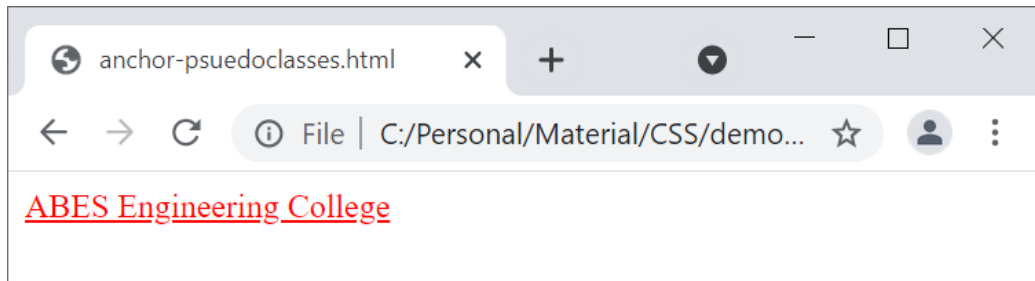
First name:

Last name:

Pseudo classes on anchor tag



<> anchor-psuedoclasses.html > ...

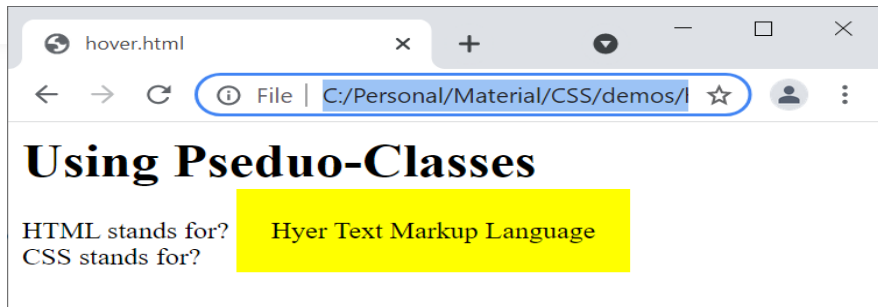
```
1  <style>
2  a:link { /* unvisited link */
3  |   color: red;
4  }
5  a:visited { /* visited link */
6  |   color: green;
7  }
8  a:hover { /* mouse over link */
9  |   color: hotpink;
10 }
11 a:active { /* selected link */
12 |   color: blue;
13 }
14 </style>
15 <a href='https://abes.ac.in'>ABES Engineering College</a>
```



Example of hover pseudo class

<> hover.html >  style

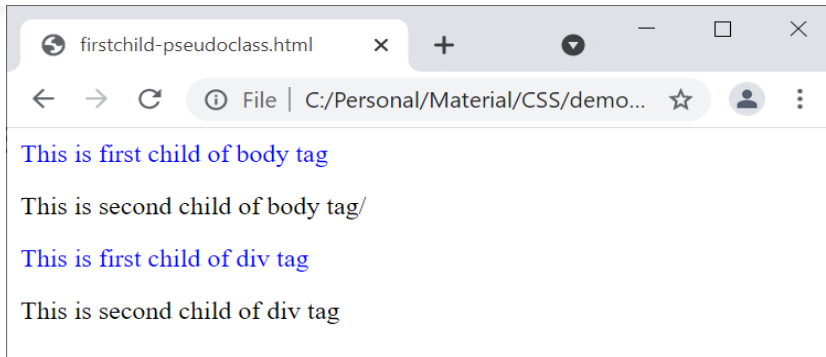
```
1 <style>
2 h1:hover {
3   color:  red;
4 }
5 p {
6   display: none;   background-color:  yellow;   padding: 20px;
7 }
8 div:hover p {
9   display: inline;
10 }
11 </style>
12 <h1>Using Pseduo-Classes</h1>
13 <div>HTML stands for?
14   <p>Hyer Text Markup Language</p>
15 </div>
16 <div>CSS stands for?
17   <p>Cascading Style Sheet</p>
18 </div>
```




Example of first-child pseudo class

<> firstchild-pseudoclass.html > ...

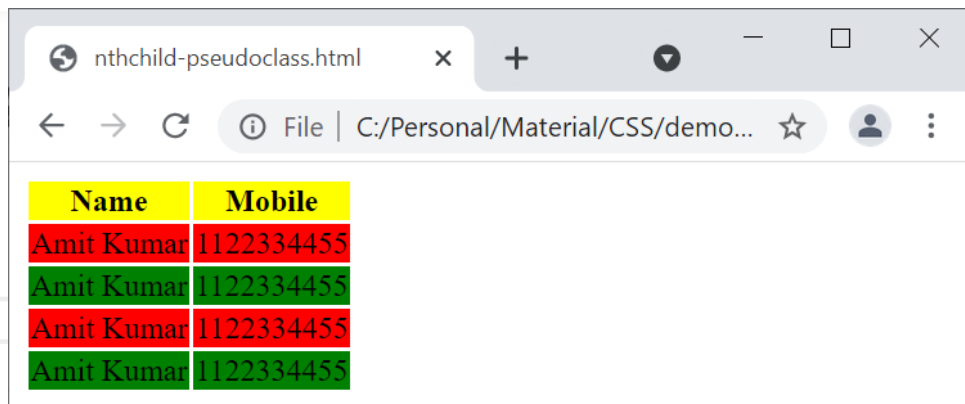
```
1  <style>
2  p:first-child {
3  |   color: ■blue;
4  }
5  </style>
6  <body>
7  <p>This is first child of body tag</p>
8  <p>This is second child of body tag</p>
9
10 <div>
11 |   <p>This is first child of div tag</p>
12 |   <p>This is second child of div tag</p>
13 </div>
14 </body>
```



Example of nth-child pseudo class

<> nthchild-pseudoclass.html >  style

```
1 <style>
2 th{ background-color: yellow}
3 tr:nth-child(odd) {
4 | background: red;
5 }
6 |
7 tr:nth-child(even) {
8 | background: green;
9 }
10 </style>
11 <table>
12 <thead><th>Name</th><th>Mobile</th></thead>
13 <tr><td>Amit Kumar</td><td>1122334455</td></tr>
14 <tr><td>Amit Kumar</td><td>1122334455</td></tr>
15 <tr><td>Amit Kumar</td><td>1122334455</td></tr>
16 <tr><td>Amit Kumar</td><td>1122334455</td></tr>
17 </table>
```



The screenshot shows a web browser window with the title 'nthchild-pseudoclass.html'. The address bar shows the file path 'C:/Personal/Material/CSS/demo...'. The browser displays a table with two columns: 'Name' and 'Mobile'. The table has four rows. The first and third rows have a yellow background, and the second and fourth rows have a green background. The text in the table is as follows:

Name	Mobile
Amit Kumar	1122334455
Amit Kumar	1122334455
Amit Kumar	1122334455
Amit Kumar	1122334455

What are pseudo elements?

- Special elements which do not really exist inside the web page but refer some part of the an existing element e.g. a letter, a line, before the tag, after the tag
- Syntax

```
selector::pseudo-element {  
    property: value;  
}
```

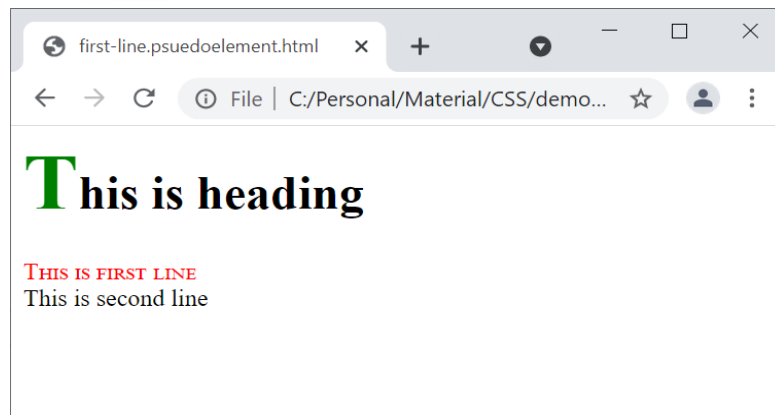
Partial list of pseudo elements

- ☐ ::first-letter
- ☐ ::first-line
- ☐ ::before
- ☐ ::after
- ☐ ::marker
- ☐ ::selection

Example of ::first-letter and ::first-line pseudo elements

<> first-line.pseudoelement.html > ...

```
1  <style>
2      h1::first-letter {
3          color: ■ green;
4          font-size: 40pt;
5      }
6      p::first-line {
7          color: ■ #ff0000;
8          font-variant: small-caps;
9      }
10 </style>
11 <h1>This is heading</h1>
12 <p>This is first line<br>
13     This is second line
14 </p>
```



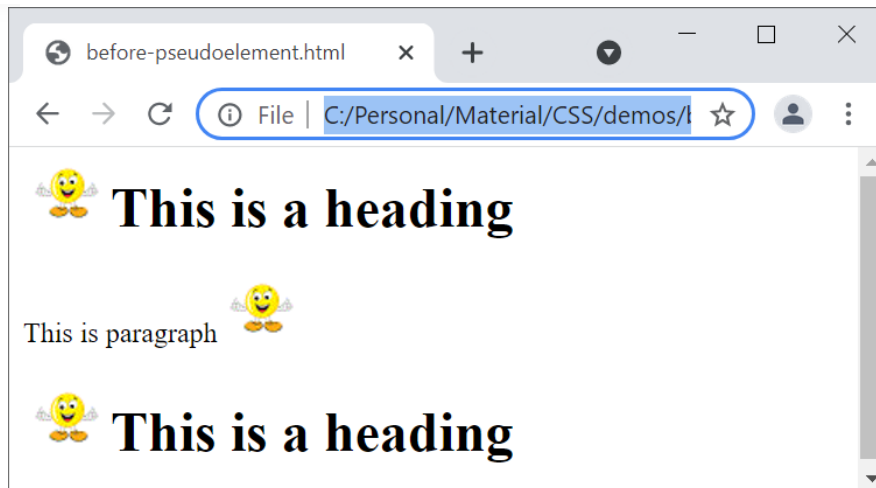
Example of ::before and ::after pseudo elements

<> before-pseudoelement.html > ...

```

1  <style>
2  h1::before {
3  |   content: url(smiley.gif);
4  | }
5  p::after {
6  |   content: url(smiley.gif);
7  | }
8  </style>
9
10 <h1>This is a heading</h1>
11 <p>This is paragraph</p>
12 <h1>This is a heading</h1>

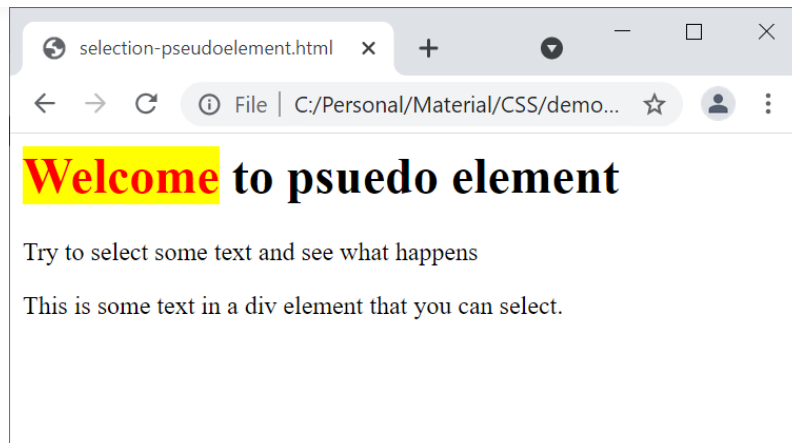
```



Example of ::selection pseudo element

<> selection-pseudoelement.html > ...

```
1  <style>
2  ::selection {
3  |   color: ■red;
4  |   background: ■yellow;
5  | }
6  </style>
7  <h1>Welcome to psuedo element</h1>
8  <p>Try to select some text and see what happens</p>
9  <div>This is some text in a div element that you can select.</div>
```



Cascading Order

- ☐ Browser default
- ☐ External and internal CSS
- ☐ Inline CSS

The inline style has the highest priority, and will override external and internal styles and browser defaults.

Specificity

What is Specificity?

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

Think of specificity as a score/rank that determines which style declaration is ultimately applied to an element.

Every CSS selector has its place in the specificity hierarchy.

There are four categories which define the specificity level of a selector:

- **Inline styles** Example: `<h1 style="color: pink;">`
- **IDs** Example: `#navbar`
- **Classes, pseudo-classes, attribute selectors** Example: `.test, :hover, [href]`
- **Elements and pseudo-elements** Example: `h1, :before`

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 1: 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 rule, it will even override inline styles!

Specificity Example

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.

CSS Layout and Positioning

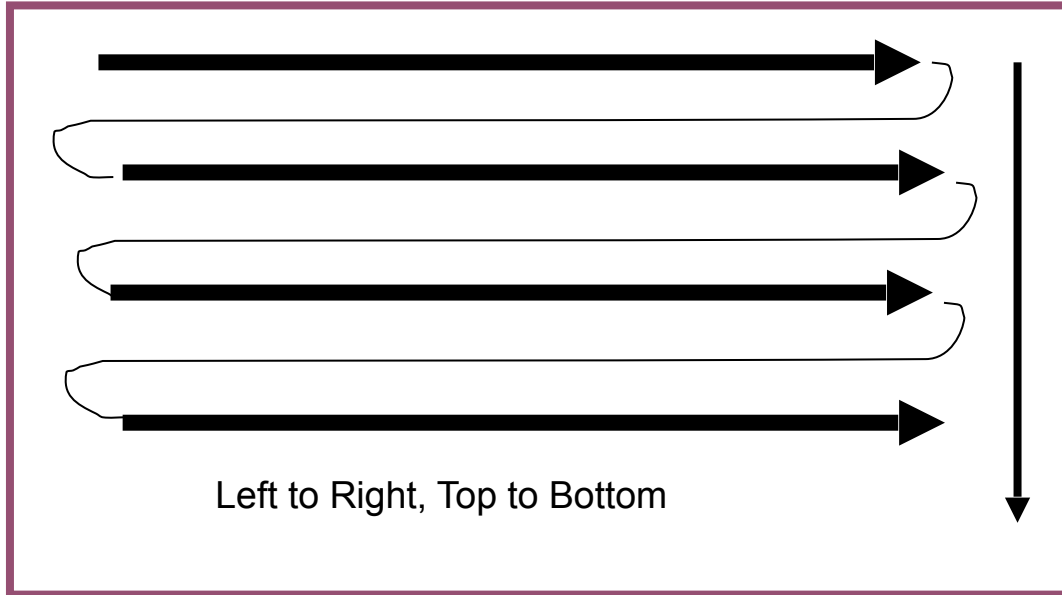
“**CSS Positioning**” refers to layout of the items on your page.

The **position** property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

There are five different position values:

- ☐ static
- ☐ relative
- ☐ fixed
- ☐ absolute
- ☐ sticky

Normal Flow – no “positioning”



Normal Flow – no “positioning”

This is a paragraph to which I have set the width.

If the next paragraph fits next to it on the right, it will line up.

This is a paragraph to which I have set the width.

However, if the second paragraph is too wide to fit the container, it will shift down.

Normal Flow – no “positioning”

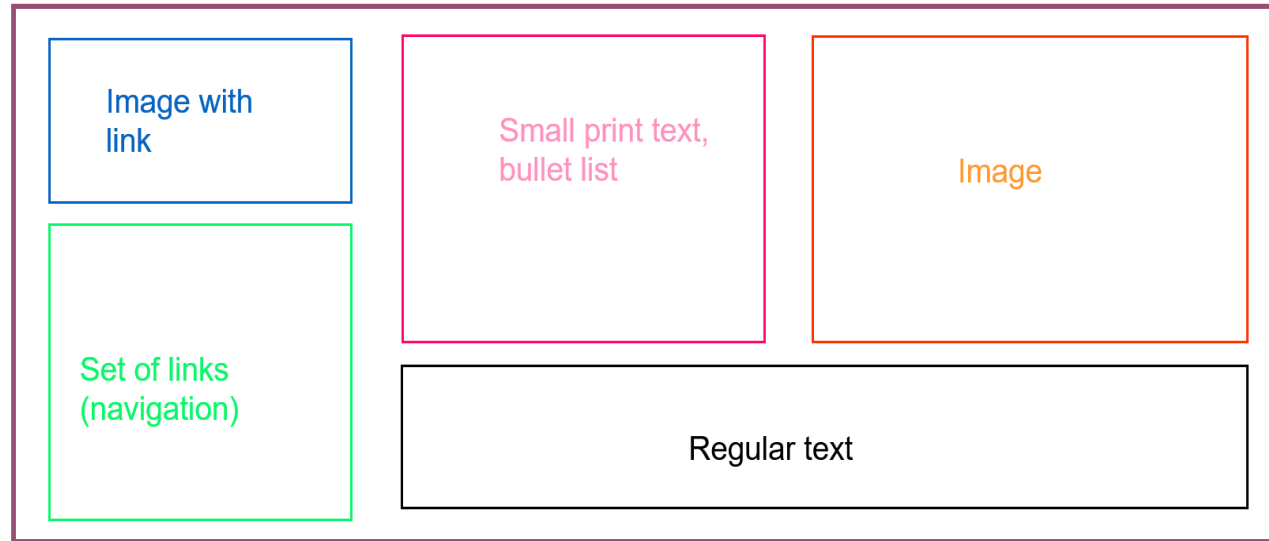
This is a paragraph to which I have set the width.

However, if the second paragraph is too wide to fit the container, it will shift down.

This is the basic principle of Normal Flow

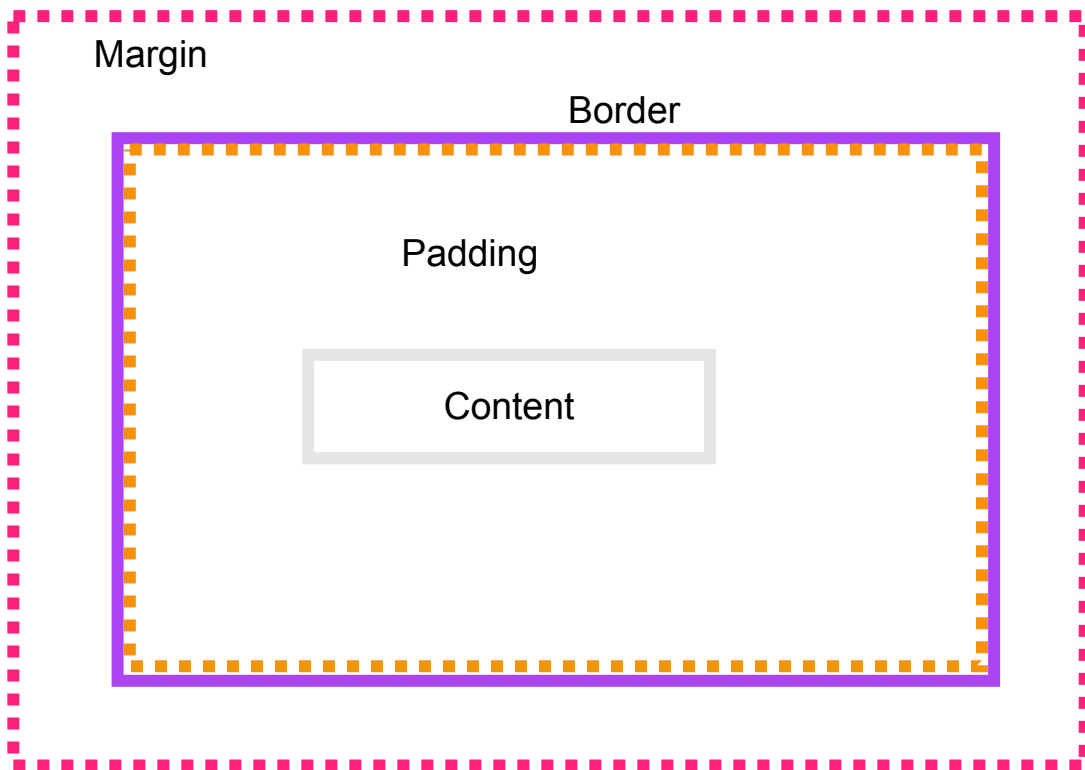
Box Model

All of the items in your webpage generate invisible “boxes” – you have to figure out how all of those boxes will fit into your page, like a puzzle.





Box Model



Margin and Padding

styleX {

margin: 10px 10px 10px 10px;

padding: 5px 5px 5px 5px; }

TRBL

top – right – bottom - left

OR

styleX {

margin: 10px;

padding: 10px; }

Shorthand:

Just one number = all 4 sides

Two numbers = top/bottom, left/right

OR

styleX {

margin: 10px 15px; padding: 5px 10px; }

Interrupt the Flow

- ☐ Absolute
- ☐ Relative
- ☐ Float

When you want to do fancier layout, you can **position** “boxes” or “containers.” By doing this, you interrupt the normal (top to bottom, left to right) flow. You can do this in three ways; **Float**, **Absolute**, and **Relative**.

Float

HTML

```
<div>

  <p> This is the normal...</p>

  <p class="float">This text is floated
right.</p>

</div>
```

CSS

```
.float { float : right; }
```

This is the normal flow of a document; from top to bottom, left to right. When the floated text is added, it moves to the **top right corner** of the containing element, in this case the <div>. Normal text flows **around** the floated text.

This text is floated right.

Absolute

HTML

```
<div>
```

```
    <p> This is the normal... <span class="abs"> This  
text is absolutely positioned.</span>...top to bottom...    </p>
```

```
</div>
```

CSS

```
abs {position: absolute; top: 40px; left: 80px;}
```

This is the normal flow of a document; from top to bottom, left to right. When you add the absolutely positioned text, it moves to the coordinates you set based on the top left corner of the containing element, in this case the <div>. Normal text flows over the absolutely positioned text. There is no gap where the text is taken from.

This text is absolutely positioned.

Relative

This text is relatively positioned.

This is the normal flow of a document; from top to bottom, left to right. When you add the relatively positioned text, it moves to the coordinates you set based on the **top left corner** of the containing element, in this case the `<div>`. Normal text flows **as normal**, but a gap is left where the relative text used to be, and the text **overlaps** with the newly positioned relative text if they are in the same area.

HTML

```
<div>
```

```
<p> This is the normal...
```

```
<span class="rel"> This text is relatively positioned. </span>
```

```
</div> ... from top to bottom...
```

```
</p>
```

CSS

```
.rel {position: relative; top: -50px; left: -150px}
```

CSS Transitions and Animations

CSS transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time.

Please refer the link below for further reading:

https://www.w3schools.com/css/css3_transitions.asp

Website Design and Typography