

Session: CSS Style sheets

CSS Style Sheets

Cascading Style Sheet(CSS) is used to set the style in web pages that contain HTML elements. It sets the background color, font-size, font-family, color, ... etc property of elements on a web page.

There are three types of CSS which are given below:

- Inline CSS
- Internal or Embedded CSS
- External CSS



CSS Style Sheets

Inline CSS: Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using the style attribute.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline CSS</title>
  </head>
  <body>
    <p style = "color:green; font-size:50px;
              font-style:italic; text-
align:center;">
      Madblocks Technologies
    </p>
  </body>
</html>
```



CSS Style Sheets

Internal or Embedded CSS: This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.

Example:

```
<!DOCTYPE html>
<html>
<head><title>Internal CSS</title>
<style>
.main {text-align:center;}
.MBT {color:#009900;font-size:50px;font-weight:bold;}
.test {font-style:bold;font-size:20px;}</style>
</head>
<body>
<div class = "main">
<div class ="MBT">Madblocks Technologies</div>
<p class ="test">Welcome to the MTA HTML & CSS Bootcamp</p>
</div></body></html>
```



CSS Style Sheets

External CSS: External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, ... etc). CSS property written in a separate file with .css extension and should be linked to the HTML document using **link** tag. This means that for each element, style can be set only once and that will be applied across web pages.

Example: The file given below contains CSS property. This file save with .css extension. For Ex: **styles.css**

```
.main {  
text-align:center;}  
.MBT {  
color:#009900;  
font-size:50px;  
font-weight:bold;}  
#test {  
font-style:bold;  
font-size:20px;}
```



CSS Style Sheets

Below is the HTML file that is making use of the created external style sheet

- **link** tag is used to link the external style sheet with the html webpage.
- **href** attribute is used to specify the location of the external style sheet file.

```
<!DOCTYPE html>
<html>
<head>
<title>External CSS</title>
<link rel="stylesheet" href="styles.css"/>
</head>
<body>
<div class = "main">
<div class ="MBT">Madblocks Technologies</div>
<p id="test">Welcome to the MTA HTML & CSS Bootcamp</p>
</div>
</body>
</html>
```



Style Sheets Precedence

Properties of CSS: Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority.

Multiple style sheets can be defined on one page. If for an HTML tag, styles are defined in multiple style sheets then the below order will be followed.

As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.

Internal or Embedded stands second in the priority list and overrides the styles in the external style sheet.

External style sheets have the least priority. If there are no styles defined either in inline or internal style sheet then external style sheet rules are applied for the HTML tags.



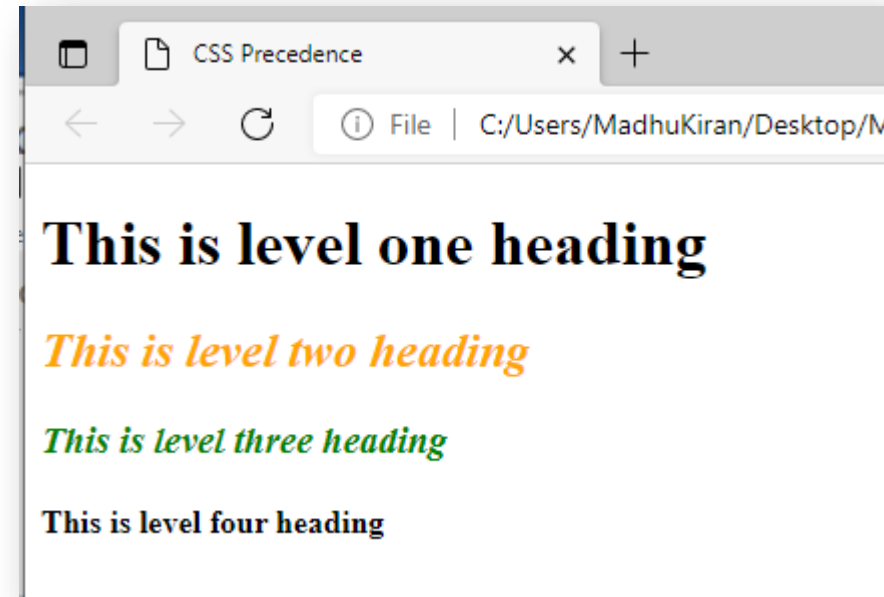
Style Sheets Precedence

Example: Let us add below code to the external style sheet in styles.css

```
h2,h3{  
    font-style: italic;  
    color:blue;  
}
```

Let us add the below code in html file save and run it

```
<!DOCTYPE html>  
<html>  
<head>  
<title>CSS Precedence</title>  
<link rel="stylesheet" href="styles.css"/>  
<style type="text/css">  
h2{color: orange;}</style>  
</head>  
<body>  
<h1>This is level one heading</h1>  
<h2>This is level two heading</h2>  
<h3 style="color: green;">This is level three heading</h3>  
<h4>This is level four heading</h4>  
</body></html>
```



You can observe Inline is first priority
Then internal css and then external css

Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.



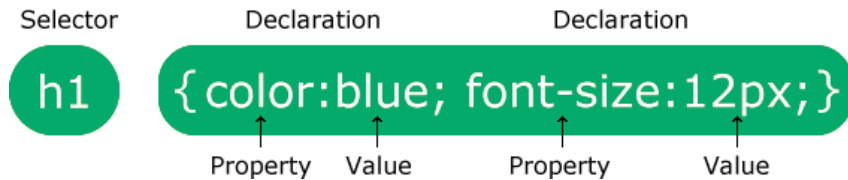


Session : CSS Rule sets

CSS Syntax

A CSS rule consists of a selector and a declaration block.

CSS Syntax:



- ✓ The selector points to the HTML element 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.
- ✓ Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.



CSS Syntax

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;
  text-align: center;
}
</style>
</head>
<body>
<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>
</body>
</html>
```

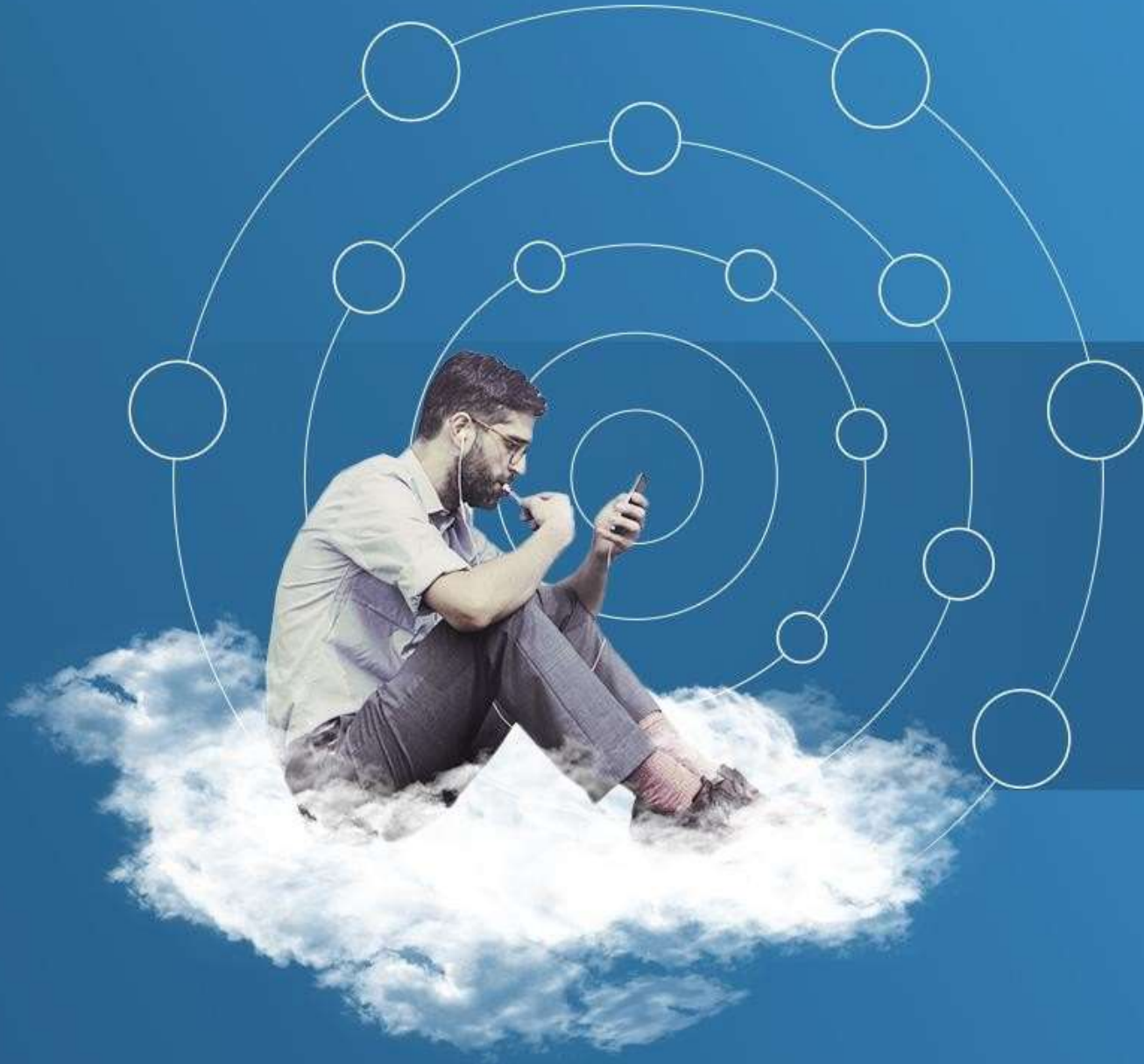


Example Explained

p is a selector in CSS (it points to the HTML element you want to style: <p>).

color is a property, and red is the property value

text-align is a property, and center is the property value



Session : CSS SELECTORS

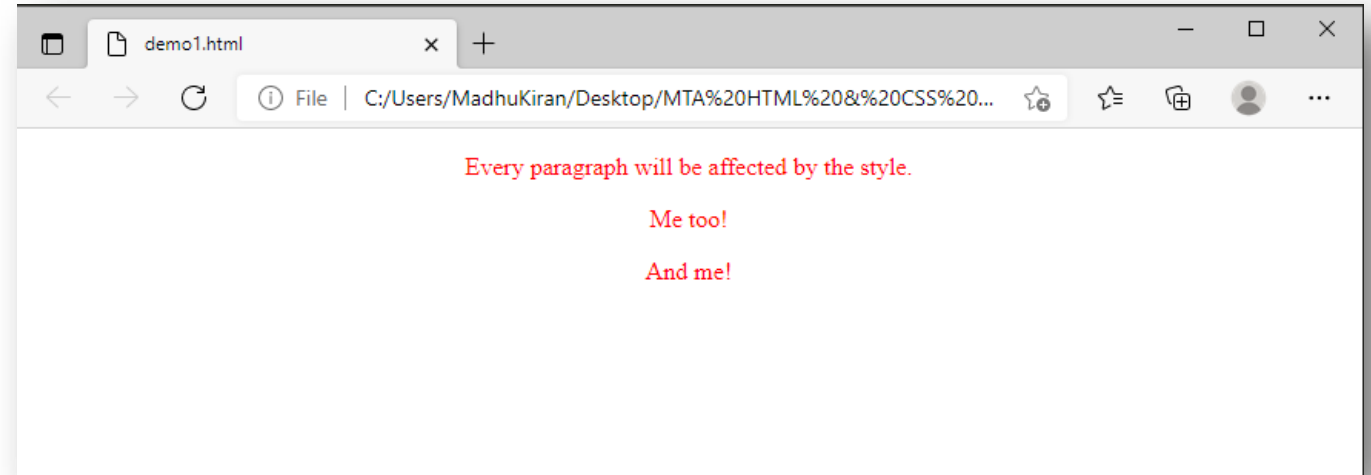
CSS element Selector

The element selector selects HTML elements based on the element name.

Example :

Here, all <p> elements on the page will be center-aligned, with a red text color:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<p>Every paragraph will be affected by the style.</p>
<p>Me too!</p>
<p>And me!</p>
</body></html>
```

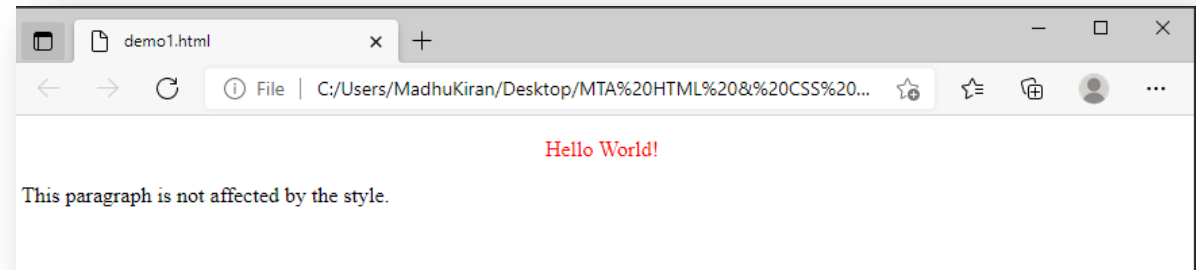


CSS id Selector

- ✓ The id selector uses the id attribute of an HTML element to select a specific element.
- ✓ The id of an element is unique within a page, so the id selector is used to select one unique element!
- ✓ To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {text-align: center; color: red;}
</style>
</head>
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
</body>
</html>
```



CSS class Selector

- ✓ The class selector selects HTML elements with a specific class attribute.
- ✓ To select elements with a specific class, write a period (.) character, followed by the class name.

Example: In this example all HTML elements with class="center" will be red and center-aligned:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

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

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1 class="center">Red and center-aligned heading</h1>
```

```
<p class="center">Red and center-aligned paragraph.</p>
```

```
</body>
```

```
</html>
```



CSS class Selector

You can also specify that only specific HTML elements should be affected by a class.

Example2: In this example only <p> elements with class="center" will be red and center-aligned:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

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

```
</style>
```

```
</head>
```

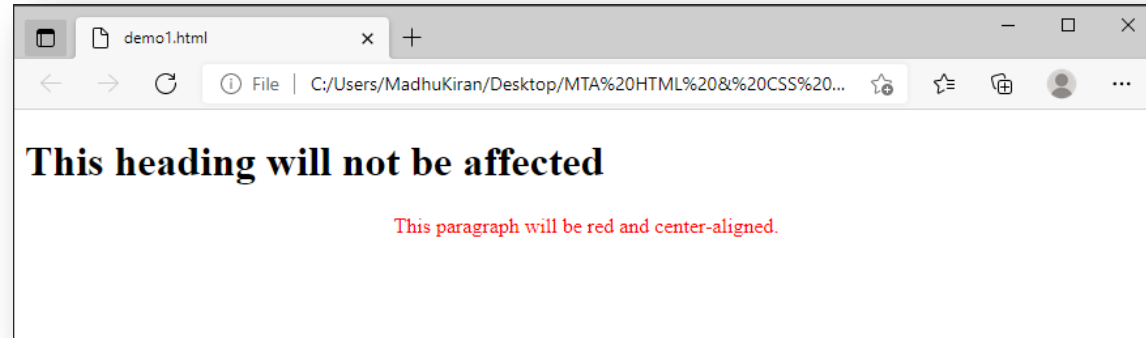
```
<body>
```

```
<h1 class="center">This heading will not be affected</h1>
```

```
<p class="center">This paragraph will be red and center-aligned.</p>
```

```
</body>
```

```
</html>
```



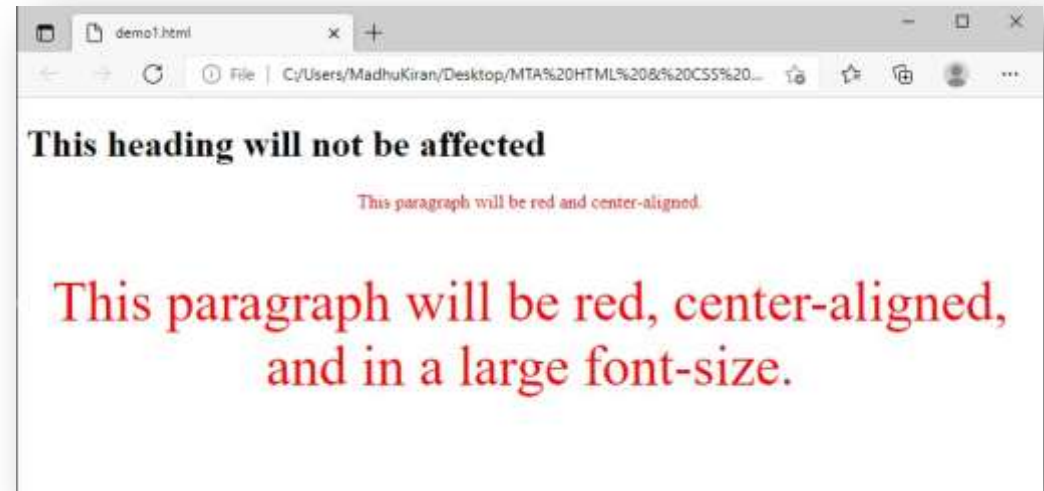
CSS class Selector

HTML elements can also refer to more than one class.

Example3: In this example the <p> element will be styled according to class="center" and to class="large":

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {text-align: center;color: red;}
p.large {font-size: 300%;}
</style>
</head>
<body>
<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
<p class="center large">This paragraph will be red, center-aligned, and in a
large font-size.</p>

</body>
</html>
```



CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

Example: The CSS rule below will affect every HTML element on the page:

```
<!DOCTYPE html>
<html>
<head>
<style>
* {text-align: center; color: blue;}
</style>
</head>
<body>
<h1>Hello world!</h1>
```

```
<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
```

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



CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

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



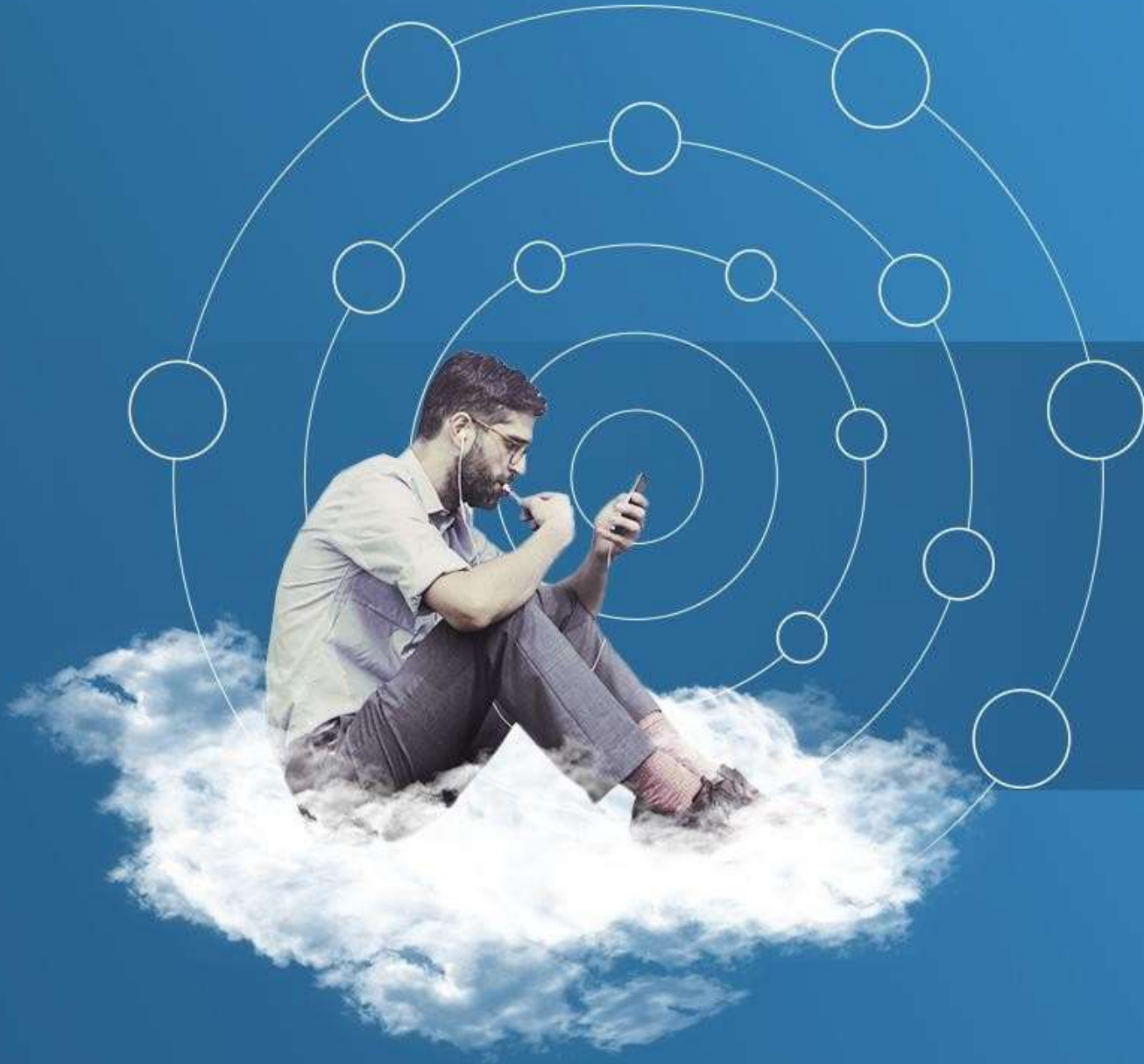
CSS Grouping Selector

It will be better to group the selectors, to minimize the code.
To group selectors, separate each selector with a comma.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
</body></html>
```





Session : CSS COMBINATORS

CSS Combinators

A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)



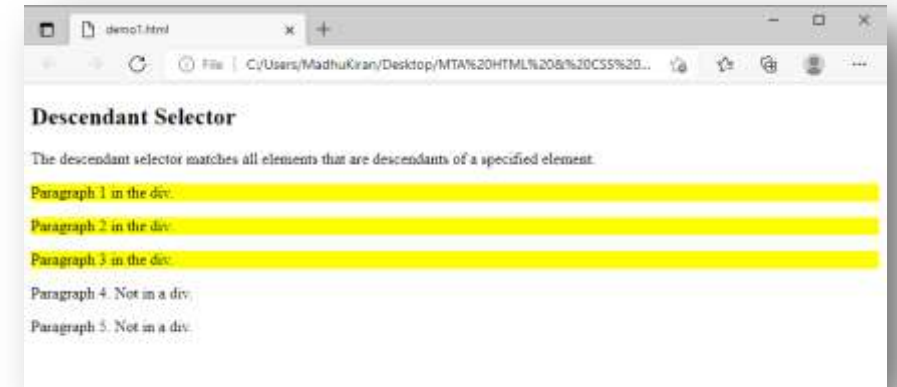
CSS Combinators

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

Example: The following example selects all `<p>` elements inside `<div>` elements

```
<!DOCTYPE html>
<html>
<head>
<style>
div p {background-color: yellow;}</style>
</head>
<body>
<h2>Descendant Selector</h2>
<p>The descendant selector matches all elements that are descendants of a specified
element.</p>
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>
<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>
</body></html>
```



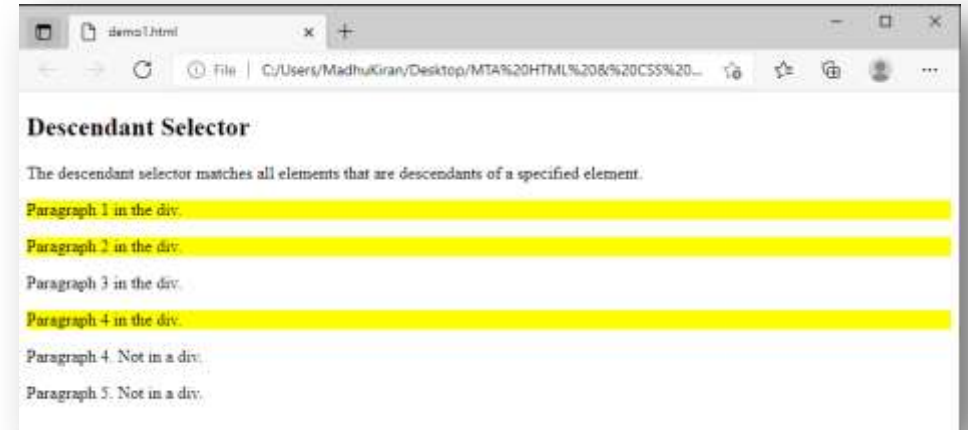
CSS Combinators

Child Selector (>)

The child selector selects all elements that are the children of a specified element.

Example: The following example selects all <p> elements that are children of a <div> element:

```
<!DOCTYPE html>
<html>
<head>
<style>
div > p {background-color: yellow;}</style>
</head>
<body>
<h2>Descendant Selector</h2>
<p>The descendant selector matches all elements that are descendants of a specified
element.</p>
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
<p>Paragraph 4 in the div.</p>
</div>
<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p></body></html>
```



CSS Combinators

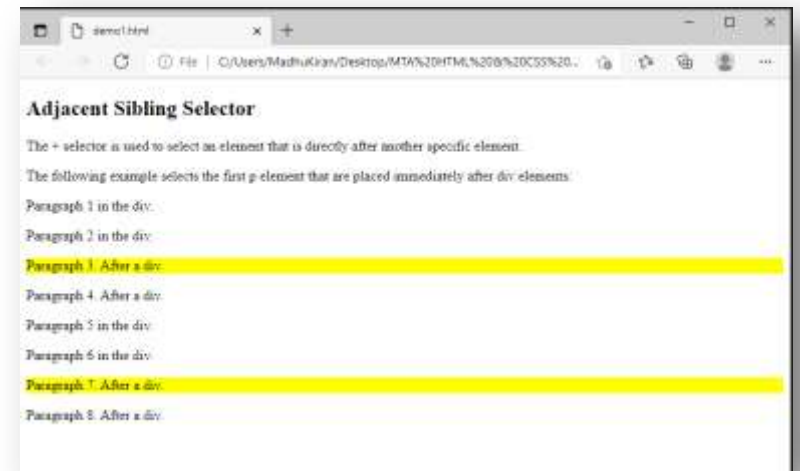
Adjacent Sibling Selector (+)

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".

Example: The following example selects the first <p> element that are placed immediately after <div> elements:

```
<!DOCTYPE html>
<html>
<head>
<style>
div + p {
  background-color: yellow;
}
</style>
</head>
<body>
<h2>Adjacent Sibling Selector</h2>
<p>The + selector is used to select an element that is directly after another specific element.</p>
<p>The following example selects the first p element that are placed immediately after div elements:</p>
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>
<p>Paragraph 3. After a div.</p>
<p>Paragraph 4. After a div.</p>
<div>
  <p>Paragraph 5 in the div.</p>
  <p>Paragraph 6 in the div.</p>
</div>
<p>Paragraph 7. After a div.</p>
<p>Paragraph 8. After a div.</p>
</body></html>
```



CSS Combinators

General Sibling Selector (~)

The general sibling selector selects all elements that are siblings of a specified element.

Example: The following example selects all <p> elements that are siblings of <div> elements:

```
<!DOCTYPE html>
<html>
<head>
<style>
div ~ p {
  background-color: yellow;
}
</style>
</head>
<body>
```

```
<h2>General Sibling Selector</h2>
```

```
<p>The general sibling selector (~) selects all elements that are siblings of a specified element.</p>
```

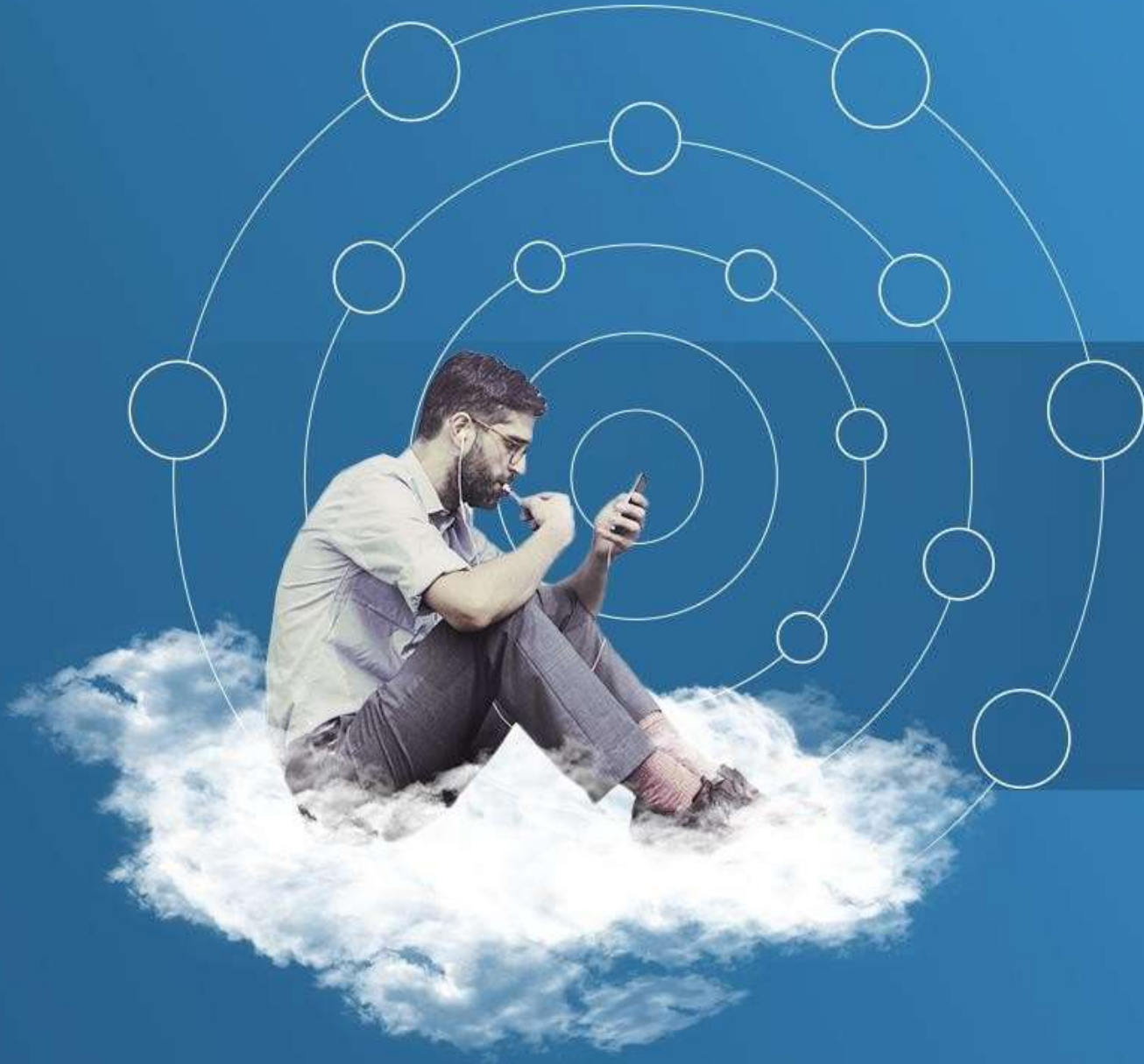
```
<p>Paragraph 1.</p>
```

```
<div>
  <p>Paragraph 2.</p>
</div>
```

```
<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>
```

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





Session :

PSEUDO CLASSES

CSS Pseudo-classes

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Syntax:

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



CSS Pseudo-classes

Anchor Pseudo-classes: Links can be displayed in different ways:

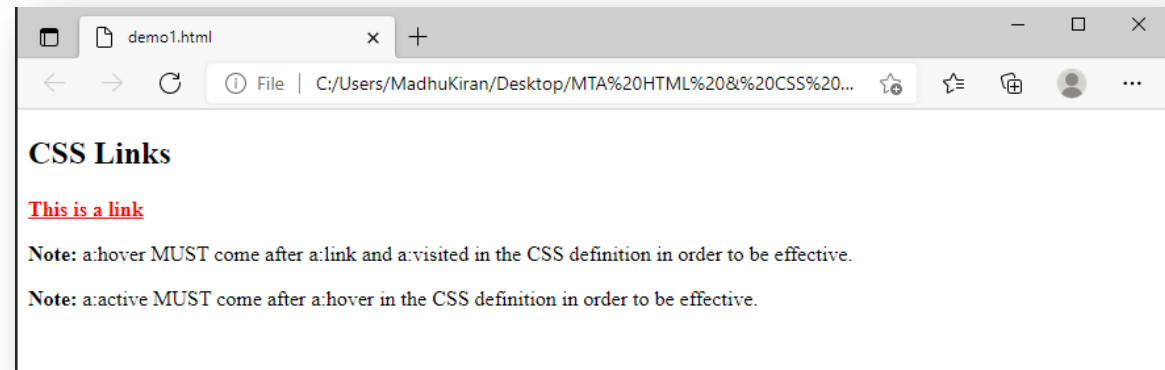
Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {color: red;}

/* visited link */
a:visited {color: green;}

/* mouse over link */
a:hover {color: hotpink;}

/* selected link */
a:active {color: blue;}
</style>
</head>
<body>
<h2>CSS Links</h2>
<p><b><a href="https://facebook.com" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be
effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>
</body>
</html>
```



CSS Pseudo-classes

Pseudo-classes and CSS Classes: Pseudo-classes can be combined with CSS classes:

Example: When you hover over the link in the example, it will change color:

```
<!DOCTYPE html>
<html>
<head>
<style>
a.highlight:hover {
color: #ff0000;
}
</style>
</head>
<body>
```

```
<h2>Pseudo-classes and CSS Classes</h2>
```

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

```
<p><a class="highlight" href="https://facebook.com">Facebook</a></p>
```

```
<p><a href="https://google.com">Google</a></p>
```

```
</body>
```

```
</html>
```



CSS Pseudo-classes

Hover on <div>:An example of using the :hover pseudo-class on a <div> element

Example:

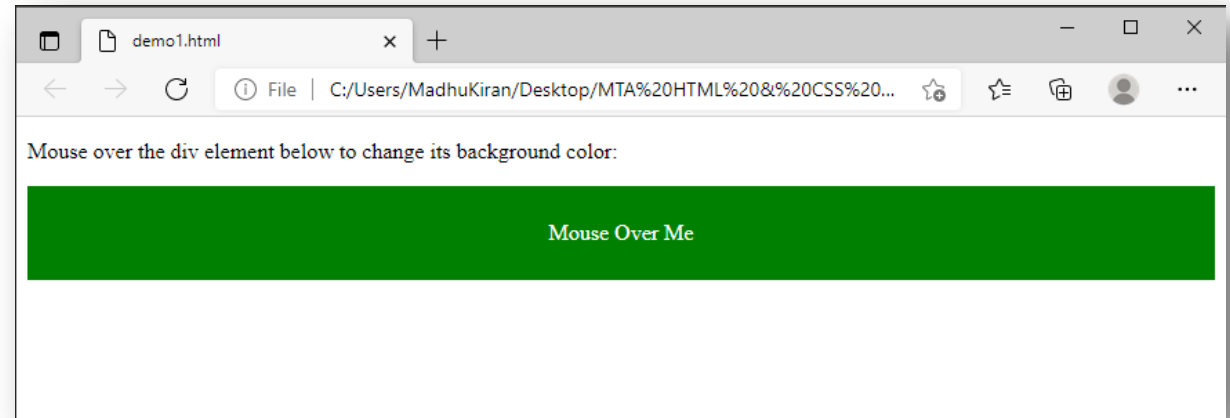
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: green;
  color: white;
  padding: 25px;
  text-align: center;
}

div:hover {
  background-color: blue;
}
</style>
</head>
<body>

<p>Mouse over the div element below to change its background color:</p>

<div>Mouse Over Me</div>

</body>
</html>
```



CSS Pseudo-classes

Simple Tooltip Hover

Hover over a <div> element to show a <p> element (like a tooltip):

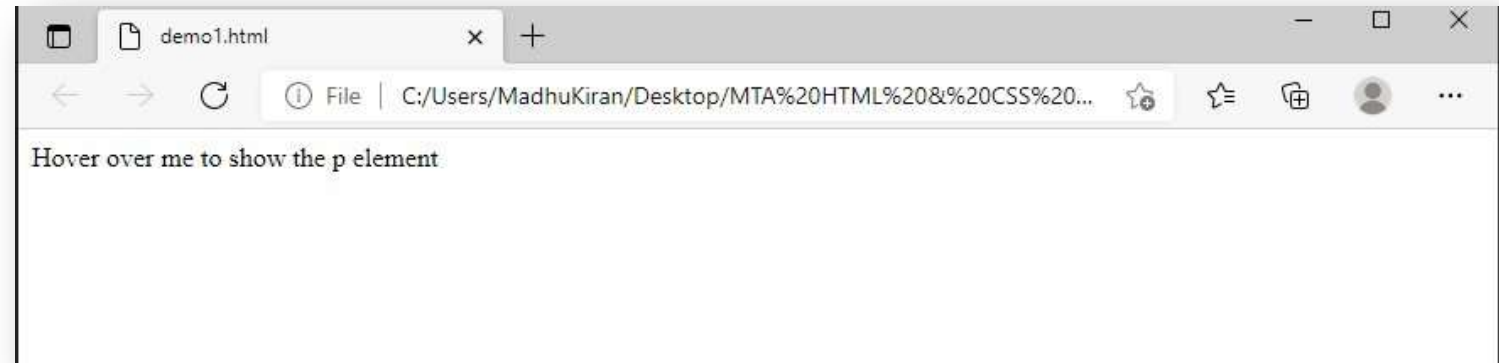
Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  display: none;
  background-color: yellow;
  padding: 20px;
}

div:hover p {
  display: block;
}
</style>
</head>
<body>

<div>Hover over me to show the p element
  <p>Tada! Here I am!</p>
</div>

</body>
</html>
```



CSS Pseudo-classes

CSS - The :first-child Pseudo-class

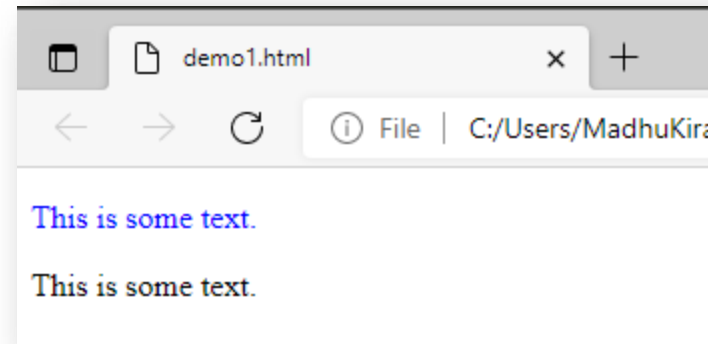
The :first-child pseudo-class matches a specified element that is the first child of another element.

Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p:first-child {
color: blue;
}
</style>
</head>
<body>
<p>This is some text.</p>
<p>This is some text.</p>
</body>
</html>
```



CSS Pseudo-classes

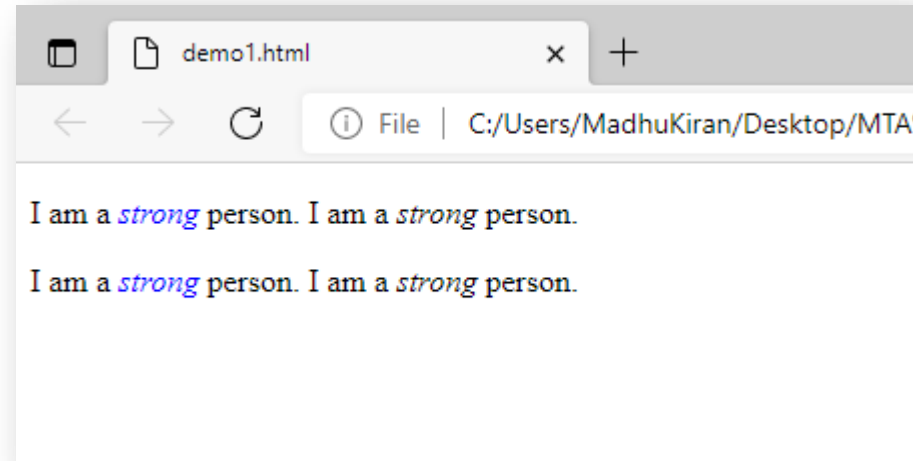
Match the first `<i>` element in all `<p>` elements

In the following example, the selector matches the first `<i>` element in all `<p>` elements:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p i:first-child {
  color: blue;
}
</style>
</head>
<body>
```

```
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
</body></html>
```



CSS Pseudo-classes

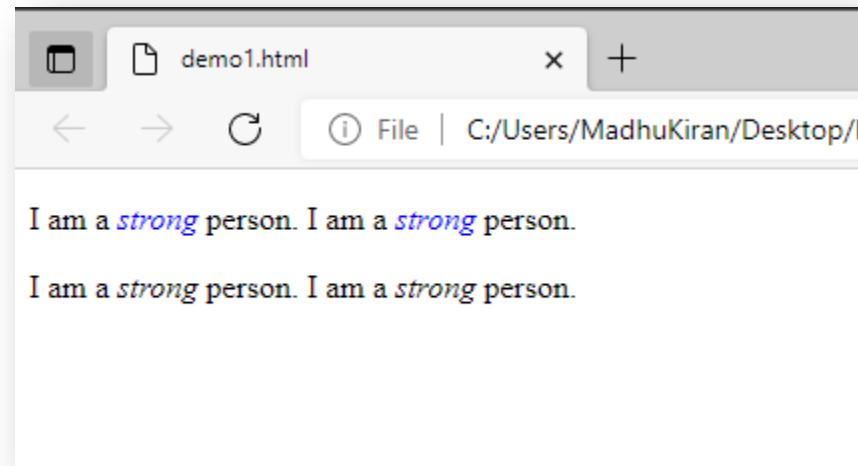
Match all `<i>` elements in all first child `<p>` elements

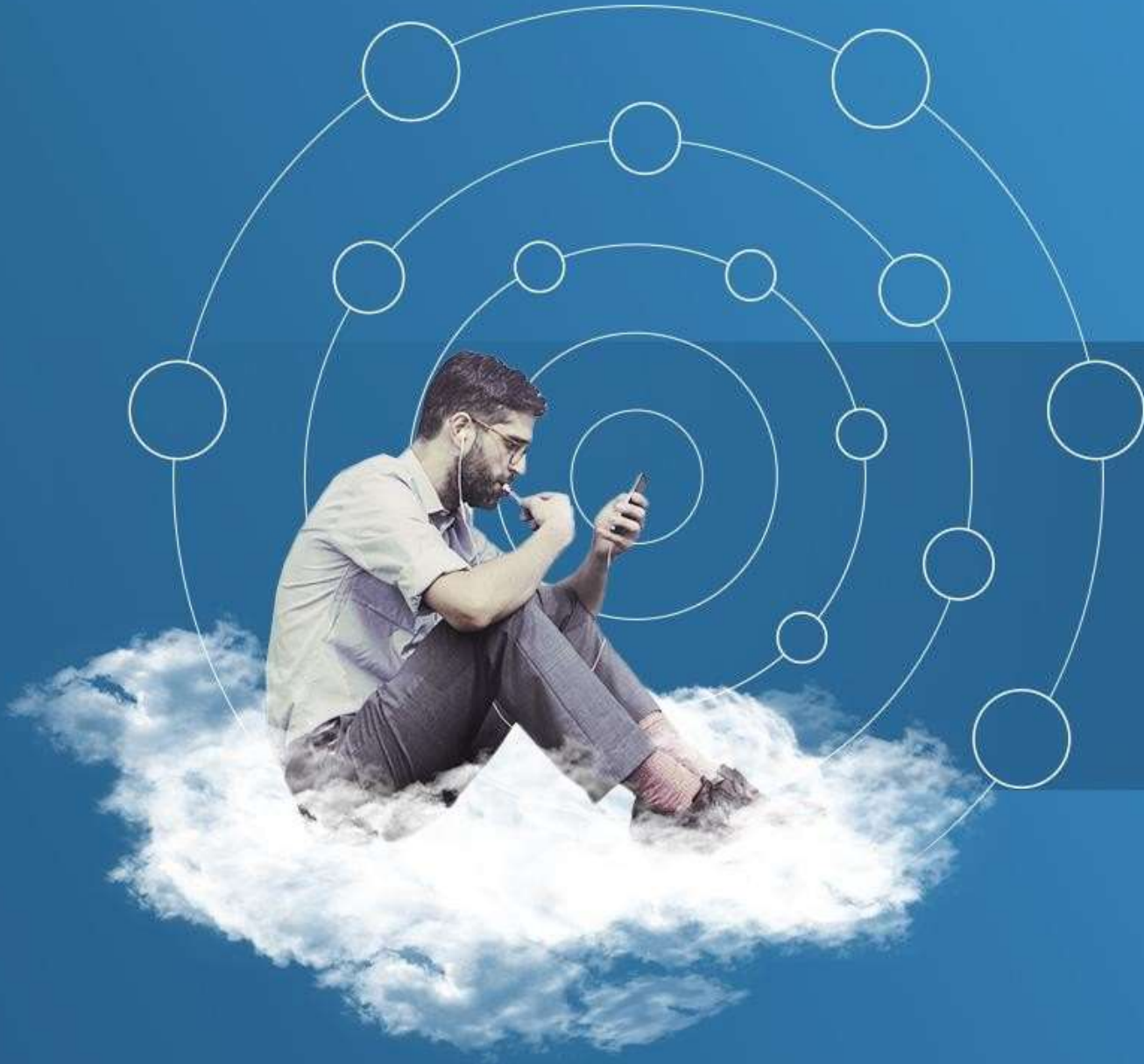
In the following example, the selector matches all `<i>` elements in `<p>` elements that are the first child of another element:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p:first-child i {
  color: blue;
}
</style>
</head>
<body>
```

```
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
</body></html>
```





Session : PSEUDO ELEMENTS

CSS Pseudo-Elements

What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Syntax

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



CSS Pseudo-Elements

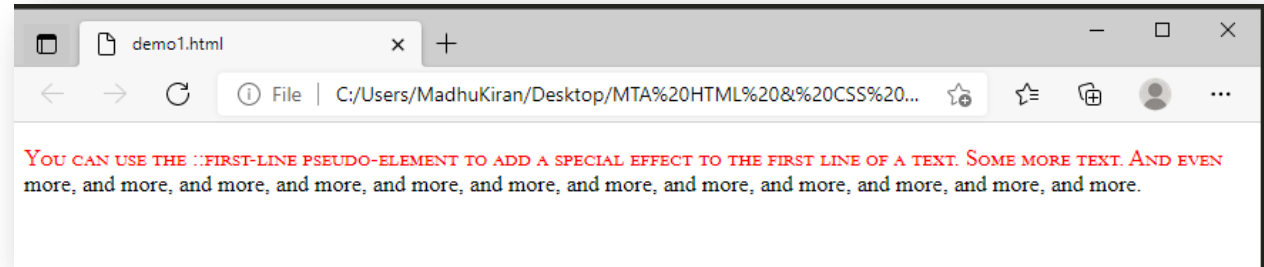
The ::first-line Pseudo-element

The `::first-line` pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all <p> elements:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
</style>
</head>
<body>
```

[illegible]

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

CSS Pseudo-Elements

Note: The `::first-line` pseudo-element can only be applied to block-level elements.

The following properties apply to the `::first-line` pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear



CSS Pseudo-Elements

The ::first-letter Pseudo-element

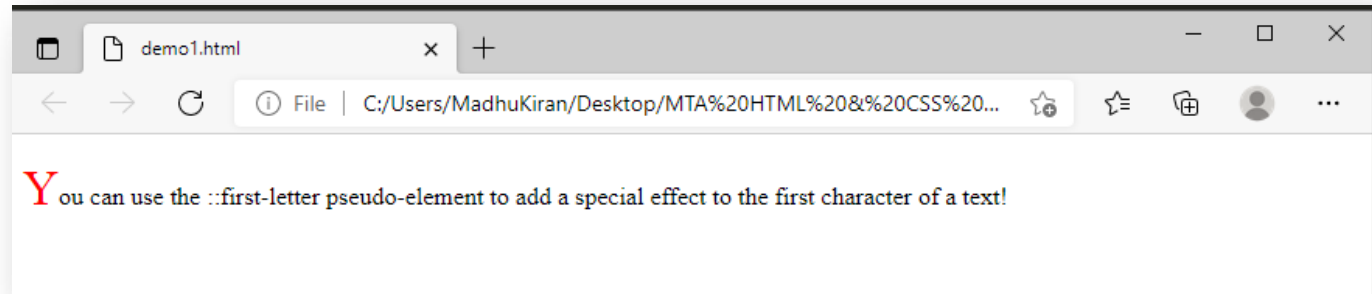
The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

The following example formats the first letter of the text in all <p> elements:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
</style>
</head>
<body>
```

```
<p>You can use the ::first-letter pseudo-element to add a special effect to the
first character of a text!</p>
</body>
</html>
```



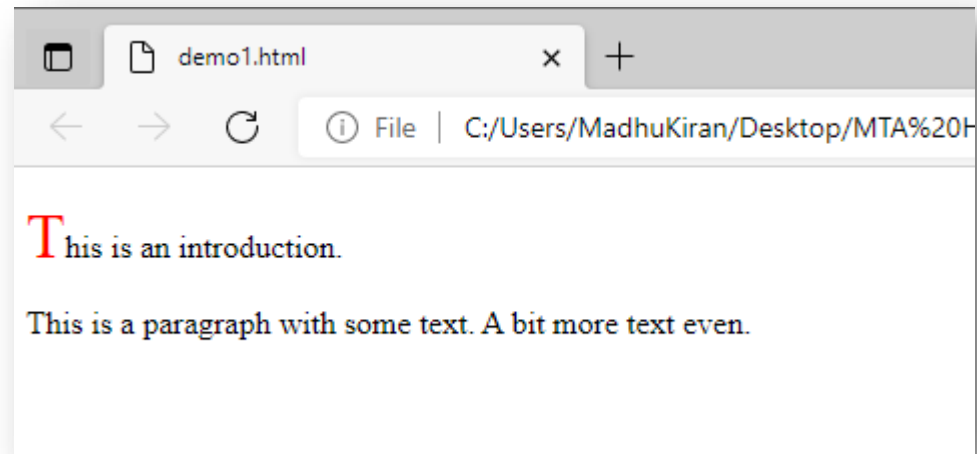
CSS Pseudo-Elements

Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.intro::first-letter {
color: #ff0000;
font-size: 200%;
}
</style>
</head>
<body>
<p class="intro">This is an introduction.</p>
<p>This is a paragraph with some text. A bit more text
even.</p></body></html>
```



CSS Pseudo-Elements

CSS - The ::before Pseudo-element

The ::before pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each <h1> element:

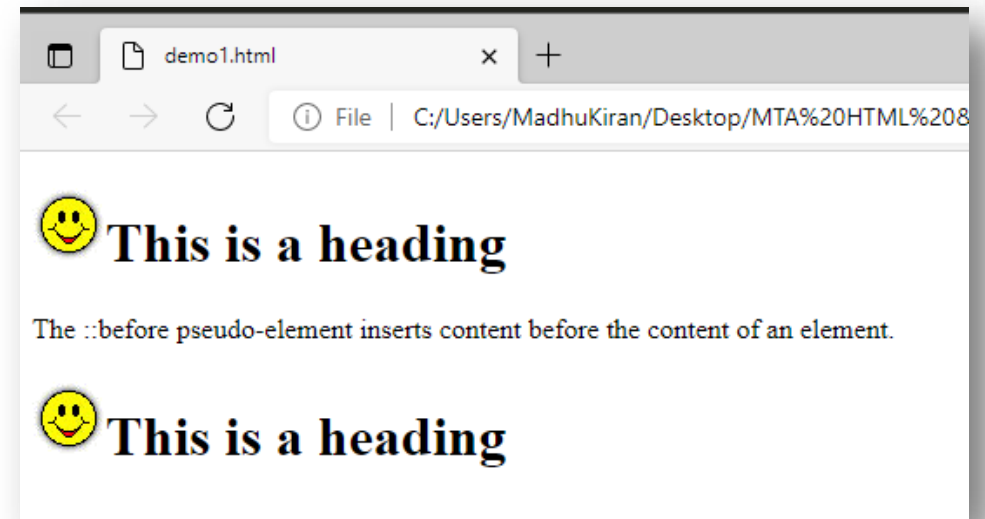
Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1::before {
  content: url(smileyface.gif);
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>The ::before pseudo-element inserts content before the content of an element.</p>

<h1>This is a heading</h1>

</body>
</html>
```



CSS Pseudo-Elements

CSS - The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after the content of each <h1> element:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1::after{
  content: url(smileyface.gif);
}
</style>
</head>
<body>
```

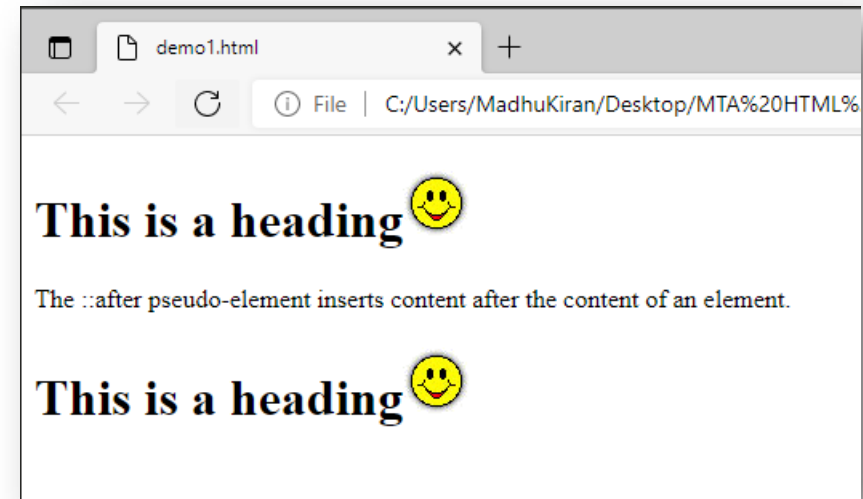
```
<h1>This is a heading</h1>
```

```
<p>The ::after pseudo-element inserts content after the content of an element.</p>
```

```
<h1>This is a heading</h1>
```

```
</body>
```

```
</html>
```



CSS Pseudo-Elements

CSS - The ::marker Pseudo-element

The ::marker pseudo-element selects the markers of list items.

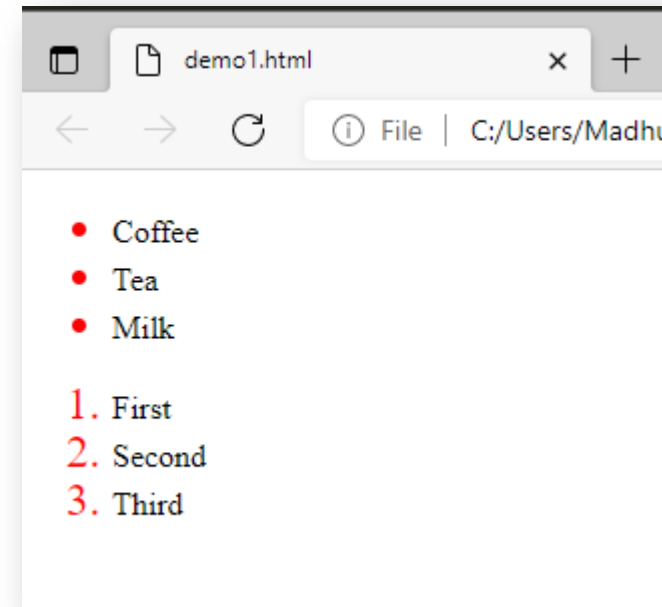
The following example styles the markers of list items:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
::marker {
  color: red;
  font-size: 23px;
}
</style>
</head>
<body>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

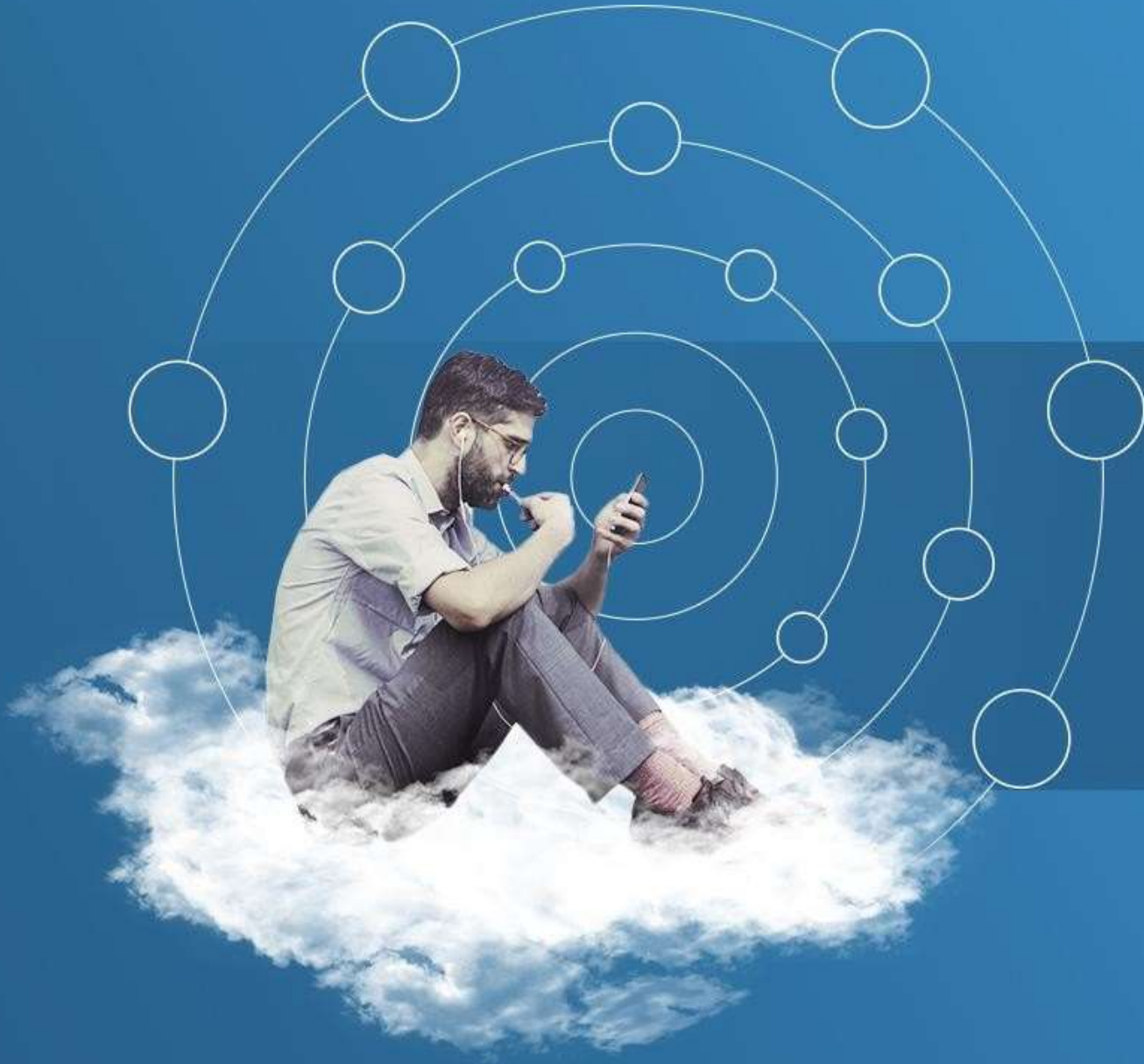
<ol>
  <li>First</li>
  <li>Second</li>
  <li>Third</li>
</ol></body></html>
```



Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.





Session: CSS COMMENTS

CSS Comments

CSS comments are not displayed in the browser, but they can help document your source code.

CSS Comments

- ✓ Comments are used to explain the code, and may help when you edit the source code at a later date.
- ✓ Comments are ignored by browsers.

Syntax: `/* content */`

A CSS comment is placed inside the `<style>` element, and starts with `/*` and ends with `*/`:

Example:

```
<style>
```

```
/* This is a single-line comment */
```

```
p {  
  color: red;  
}
```

```
</style>
```



CSS Comments

You can add comments wherever you want in the code:

Example:

```
<style>
p {
  color: red; /* Set text color to red */
}
</style>
```

Comments can also span multiple lines:

Example:

```
<style>
/* This is
a multi-line
comment */
```

```
p {
  color: red;
}
</style>
```



CSS Comments

HTML and CSS Comments

From the HTML tutorial, you learned that you can add comments to your HTML source by using the `<!--...-->` syntax.

In the following example, we use a combination of HTML and CSS comments:

Example:

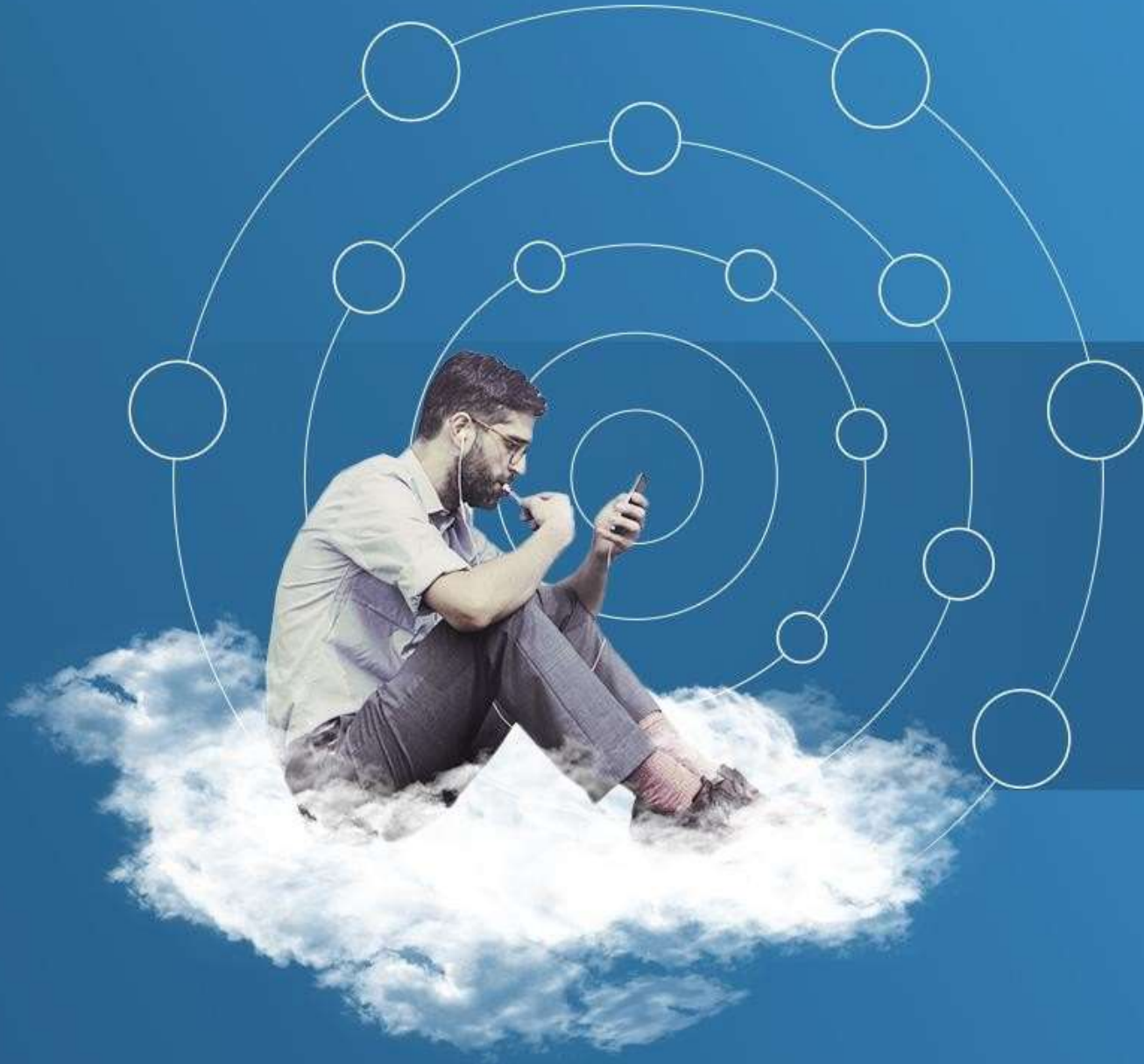
```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red; /* Set text color to red */
}
</style>
</head>
<body>

<h2>My Heading</h2>

<!-- These paragraphs will be red -->
<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
<p>CSS comments are not shown in the output.</p>

</body></html>
```





Session : LAYOUTS

IMPORTANT CSS LAYOUT PROPERTIES

font-style
font-weight
text-decoration
border
border-width
border-color
border-radius
color
background-color
cursor

font-size
font-family
height
width
background-image
background-size
margin
padding



IMPORTANT CSS PROPERTIES

Syntax:

font-style: normal;

italic;

oblique;

font-weight: bold;

100 to 1000;



IMPORTANT CSS PROPERTIES

Syntax:

text-decoration: underline;
overline;
line-through;

border: solid;
dashed;
dotted;



IMPORTANT CSS PROPERTIES

Syntax:

`border-width: 0px;
 2px;`

`border-color: red;
 green;`



IMPORTANT CSS PROPERTIES

Syntax:

```
border-radius: 0px;  
              10px;
```

```
color: red;  
      green;
```



IMPORTANT CSS PROPERTIES

Syntax:

```
background-color: red;  
                green;
```

```
cursor: pointer;  
        move;  
        crosshair;
```



IMPORTANT CSS PROPERTIES

Syntax:

font-size: 10px;
50px;

height: 100px;
50%;



IMPORTANT CSS PROPERTIES

Syntax:

width: 100px;
50%;

background-image: url('link address');
background-size: cover;



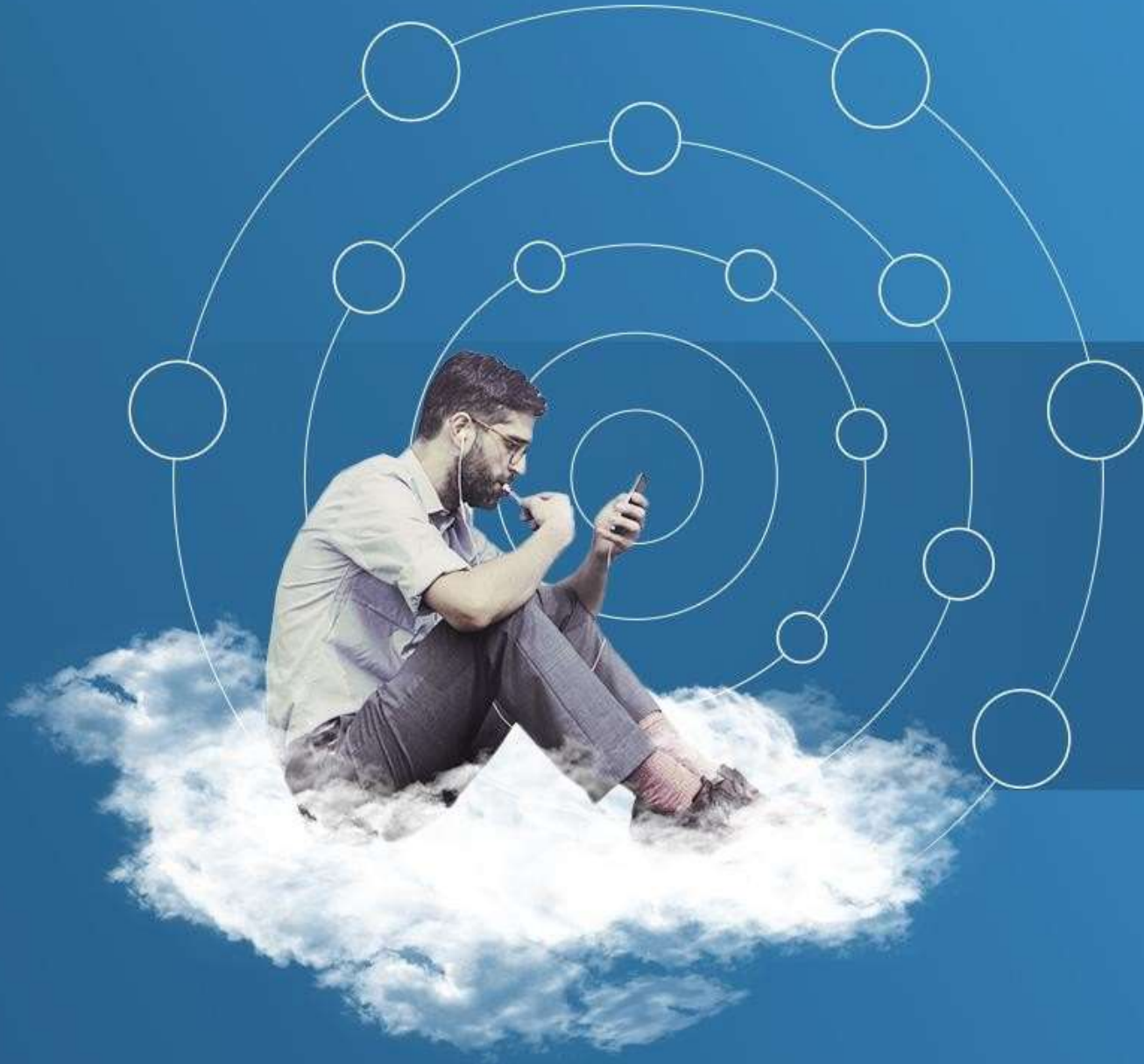
IMPORTANT CSS PROPERTIES

Syntax:

margin: 10px;
5%;

padding: 10px;
5%;





Session: TYPOGRAPHY

CSS Web Safe Fonts

Best Web Safe Fonts for HTML and CSS:

The following list are the best web safe fonts for HTML and CSS:

- Arial (sans-serif)
- Verdana (sans-serif)
- Helvetica (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

Note: Before you publish your website, always check how your fonts appear on different browsers and devices, and always use fallback fonts!



CSS Web Safe Fonts

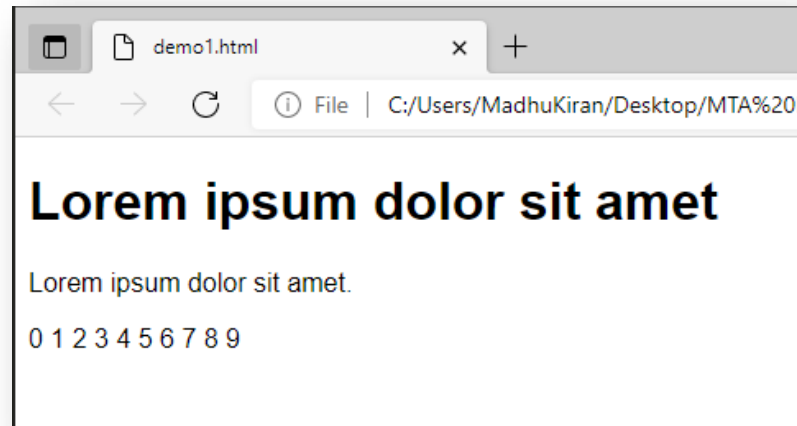
Arial (sans-serif): Arial is the most widely used font for both online and printed media.

Arial is also the default font in Google Docs.

Arial is one of the safest web fonts, and it is available on all major operating systems.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: Arial, sans-serif;
}
</style>
</head>
<body>
<h1>Lorem ipsum dolor sit amet</h1>
<p>Lorem ipsum dolor sit amet.</p>
<p>0 1 2 3 4 5 6 7 8 9</p>
</body></html>
```



CSS Web Safe Fonts

Verdana (sans-serif)

Verdana is a very popular font. Verdana is easily readable even for small font sizes.

Example:

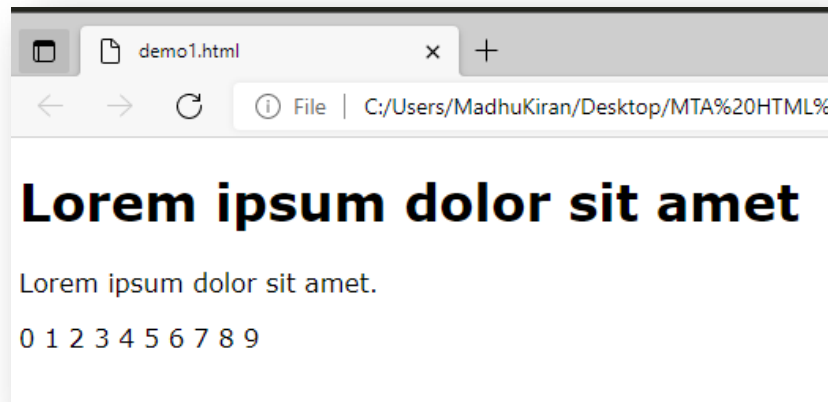
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: Verdana, sans-serif;
}
</style>
</head>
<body>
```

```
<h1>Lorem ipsum dolor sit amet</h1>
```

```
<p>Lorem ipsum dolor sit amet.</p>
```

```
<p>0 1 2 3 4 5 6 7 8 9</p>
```

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



CSS Web Safe Fonts

Helvetica (sans-serif)

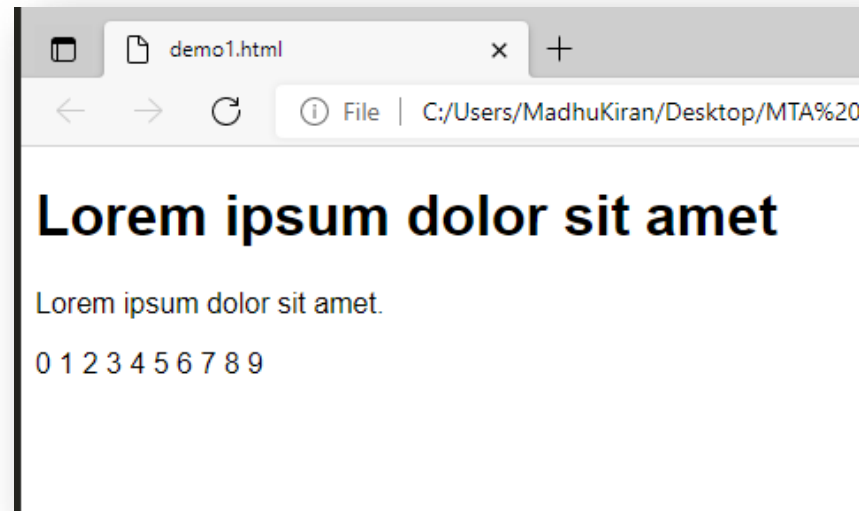
The Helvetica font is loved by designers. It is suitable for many types of business.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: Helvetica, sans-serif;
}
</style>
</head>
<body>

<h1>Lorem ipsum dolor sit amet</h1>

<p>Lorem ipsum dolor sit amet.</p>
<p>0 1 2 3 4 5 6 7 8 9</p>
</body></html>
```



CSS Web Safe Fonts

Tahoma (sans-serif)

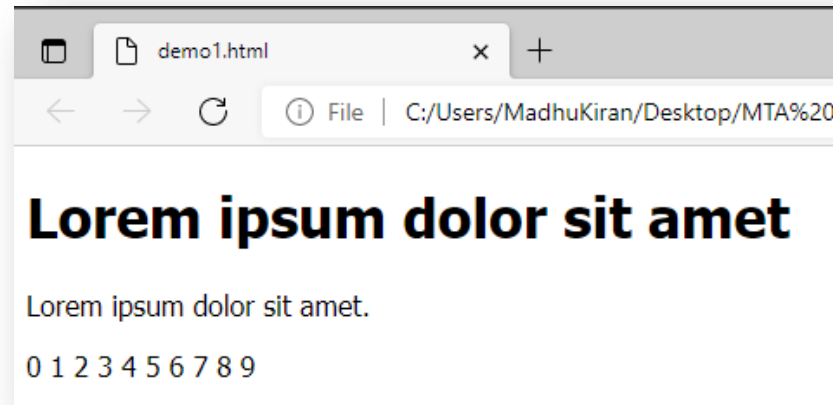
The Tahoma font has less space between the characters.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: Tahoma, sans-serif;
}
</style>
</head>
<body>

<h1>Lorem ipsum dolor sit amet</h1>

<p>Lorem ipsum dolor sit amet.</p>
<p>0 1 2 3 4 5 6 7 8 9</p>
</body>
</html>
```



CSS Web Safe Fonts

Trebuchet MS (sans-serif)

Trebuchet MS was designed by Microsoft in 1996. Use this font carefully. Not supported by all mobile operating systems.

Example:

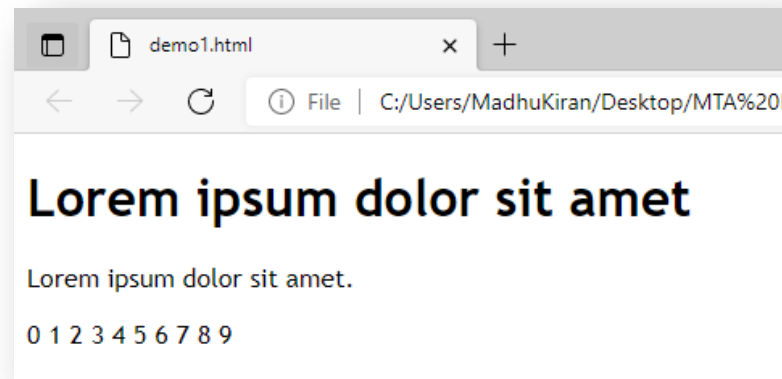
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: 'Trebuchet MS', sans-serif;
}
</style>
</head>
<body>
```

```
<h1>Lorem ipsum dolor sit amet</h1>
```

```
<p>Lorem ipsum dolor sit amet.</p>
```

```
<p>0 1 2 3 4 5 6 7 8 9</p>
```

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



CSS Web Safe Fonts

Times New Roman (serif)

Times New Roman is one of the most recognizable fonts in the world. It looks professional and is used in many newspapers and "news" websites. It is also the primary font for Windows devices and applications.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: 'Times New Roman', serif;
}
</style>
</head>
<body>

<h1>Lorem ipsum dolor sit amet</h1>

<p>Lorem ipsum dolor sit amet.</p>
<p>0 1 2 3 4 5 6 7 8 9</p>
</body></html>
```



CSS Web Safe Fonts

Georgia (serif)

Georgia is an elegant serif font. It is very readable at different font sizes, so it is a good candidate for mobile-responsive design.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: Georgia, serif;
}
</style>
</head>
<body>
```

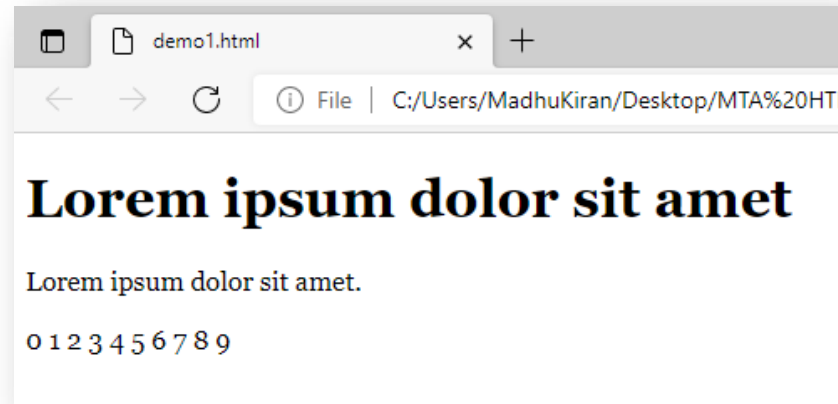
```
<h1>Lorem ipsum dolor sit amet</h1>
```

```
<p>Lorem ipsum dolor sit amet.</p>
```

```
<p>0 1 2 3 4 5 6 7 8 9</p>
```

```
</body>
```

```
</html>
```



CSS Web Safe Fonts

Garamond (serif)

Garamond is a classical font used for many printed books. It has a timeless look and good readability.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: Garamond, serif;
}
</style>
</head>
<body>
```

```
<h1>Lorem ipsum dolor sit amet</h1>
```

```
<p>Lorem ipsum dolor sit amet.</p>
```

```
<p>0 1 2 3 4 5 6 7 8 9</p>
```

```
</body>
```

```
</html>
```



CSS Web Safe Fonts

Courier New (monospace)

Courier New is the most widely used monospace serif font. Courier New is often used with coding displays, and many email providers use it as their default font. Courier New is also the standard font for movie screenplays.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: 'Courier New', monospace;
}
</style>
</head>
<body>

<h1>Lorem ipsum dolor sit amet</h1>

<p>Lorem ipsum dolor sit amet.</p>
<p>0 1 2 3 4 5 6 7 8 9</p>
</body></html>
```



CSS Web Safe Fonts

Brush Script MT (cursive)

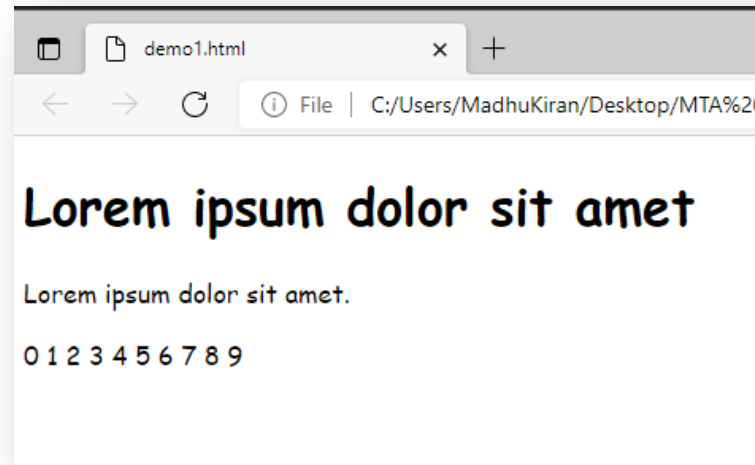
The Brush Script MT font was designed to mimic handwriting. It is elegant and sophisticated, but can be hard to read. Use it carefully.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: 'Brush Script MT', cursive;
}
</style>
</head>
<body>

<h1>Lorem ipsum dolor sit amet</h1>

<p>Lorem ipsum dolor sit amet.</p>
<p>0 1 2 3 4 5 6 7 8 9</p>
</body>
</html>
```



Properties of Fonts

Font Style

The font-style property is used to set the font face style for the text content of an element. The font style can be normal, italic or oblique. The default value is normal.

Syntax:

```
Selector{  
  font-style:style name;  
}
```

Example:

```
E1 > e9.html  
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <title>Styled List Markers</title>  
  <link rel="stylesheet" href="sample.css">  
</head>  
<body>  
  <p class = "a">Hello 1</p>  
  <p class = "b">Hello 2</p>  
  <p class = "c">Hello 3</p>  
</body>  
</html>
```

```
E1 > # sample.css > .c  
.a{  
  font-style:normal;  
}  
.b{  
  font-style:italic;  
}  
.c{  
  font-style:oblique;  
}
```



Properties of Fonts

Font Size

The font-size property is used to set the size of font for the text content of an element. There are several ways to specify the font size values e.g. with keywords, percentage, pixels, ems, etc.

Syntax:

```
selector{  
  font-size: font size value;  
}
```

Example:

```
p { font-size: 14px; }  
p { font-size: 0.875em; } /* 14px/16px=0.875em */  
p { font-size: 1.4em; } /* 1.4em = 14px */  
p { font-size: 1.4rem; } /* 1.4rem = 14px */  
p { font-size: 62.5%; } /* font-size 1em = 10px */  
p { font-size: larger; }  
p { font-size: 1vw; }  
p { font-size: calc(1em + 1vw); }
```



Properties of Fonts

Font Weight

The font-weight property specifies the weight or boldness of the font. This property can take one of the following values: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900 and inherit.

The numeric values 100-900 specify the font weights, where each number represents a weight greater than its predecessor. 400 is same as normal & 700 is same as bold.

Syntax:

```
selector{  
  font-weight: font weight value;  
}
```

Example:

```
P{  
  font-weight : bold;  
  font-weight : 500;  
}
```



Properties of Fonts

Font Variant

The font-variant property allows the text to be displayed in a special small-caps variation. Small caps or small capital letters are slightly different to normal capital letters, in which lowercase letters appear as smaller versions of the corresponding uppercase letters.

Syntax:

```
selector{  
  font-variant: font-variant-value;  
}
```

Example:

```
P{  
  font-variant : normal / small-caps  
}
```



Text formatting

We will use text formatting properties to style the text to get a better look and feel.

There are few text formatting properties those are:

1. text-color
2. text-alignment
3. text-decoration
4. text-transformation
5. text-indentation
6. letter-spacing
7. line-height
8. text-direction
9. text-shadow
10. word-spacing



Properties of Text

Text color

It is used to set to color to the text. We can set the color in different ways like name, hexadecimal, rgb values

.Syntax:

```
selector{  
  color: color_value;  
}
```

Syntax:

Example:

```
h1 {  
    color: green;  
}
```



Properties of Text

Text align

It is used to set the alignment of the text horizontally only. The text can be set to left, right, centered, or justified. If we set justified then the text will stretch to touch the left and right edges if the text is less.

Syntax:

```
selector{  
    text-align : value; (auto | left | right | center )  
}
```

Example:

```
h1 {  
    text-align : left;  
}
```



Properties of Text

Text decoration

It is used to remove decoration of the text. Those decorations can be underline, overline, line-through or none.

Syntax:

```
selector{  
    text-decoration : value; (none | underline | overline | line-through )  
}
```

Example:

```
h1 {  
    text-decoration : underline;  
}
```



Properties of Text

Text transformation

It is used to set the case of the text. Transformation be like uppercase or lowercase, or capitalise. Capitalise means changing the first letter of the each into uppercase.

Syntax:

```
selector{  
    text-transform : value; (none | capitalize | uppercase | lowercase )  
}
```

Example:

```
h1 {  
    text-transform: uppercase;  
}
```



Properties of Text

Text Indentation

this property is used to indent the first line of the paragraph. The size can be in px, cm, pt.

Syntax:

```
selector{  
    text-indent : value; (px | % )  
}
```

Example:

```
h1 {  
    text-indent: 20%;  
}
```



Properties of Text

Letter Spacing

This property is used to specify the space between the characters of the text. The size can be given in px.

Syntax:

```
selector{  
    letter-spacing : value; (px | em | normal )  
}
```

Example:

```
h1 {  
    letter-spacing: 2em;  
}
```



Properties of Text

Line Height

This property is used to set the space between the lines.

Syntax:

```
selector{  
    line-height : value; (px | % | number | normal )  
}
```

Example:

```
h1 {  
    line-height: 1.5;  
}
```



Properties of Text

Text Direction

Text direction property is used to set the direction of the text. The direction can be set by using rtl(right to left) . Left to right is the default direction of the text.

Syntax:

```
selector{  
    direction : value; (ltr | rtl)  
}
```

Example:

```
h1 {  
    direction: ltr;  
}
```



Properties of Text

Text Shadow

Text shadow property is used to add shadow to the text. You can specify the horizontal size, vertical size and shadow color for the text.

Syntax:

```
Selector{  
    text-shadow : h-shadow v-shadow blur-radius color;  
}
```

Example:

```
h1 {  
    text-shadow: 2px 2px black;  
}
```



Properties of Text

Word spacing

Word spacing is used to specify the space between the words of the line. The size can be given in px.

Syntax:

```
Selector{  
    word-spacing: value; (px | em | %)  
}
```

Example:

```
h1 {  
    word-spacing : 2px  
}
```



Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.



BOX MODEL

1. The CSS box model has multiple properties. It is a combination of content, padding, border, and margin properties.
2. By using this box model we can create any layout and design for a web page.
3. It is used as a toolkit to customize the HTML elements.
4. web browsers take every HTML element as a rectangular box.

According to the box model, its properties are:

- Content
- Padding
- Border
- Margin



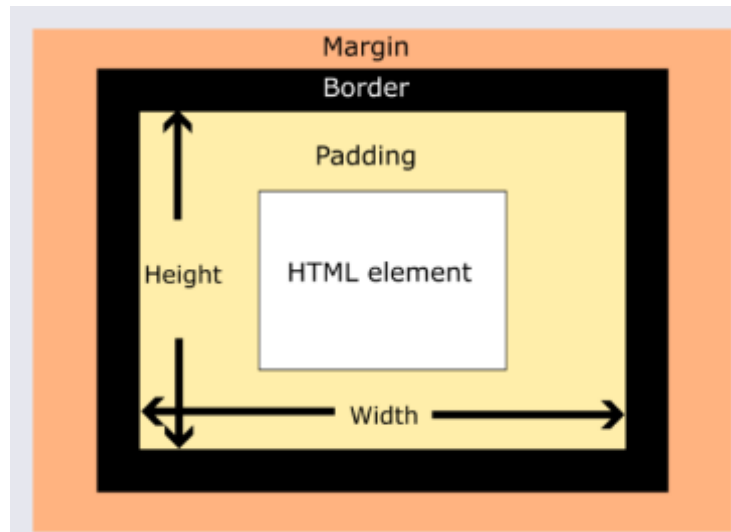
BOX MODEL

content: This property is used to displays the text, images, etc, that can be sized using the width & height property.

padding: This property is used to create space around the element, inside any defined border.

border: This property is used to cover the content & any padding, & also allows to set the style, color, and width of the border.

margin: This property is used to create empty space around the element outside the border ie., around the border area.



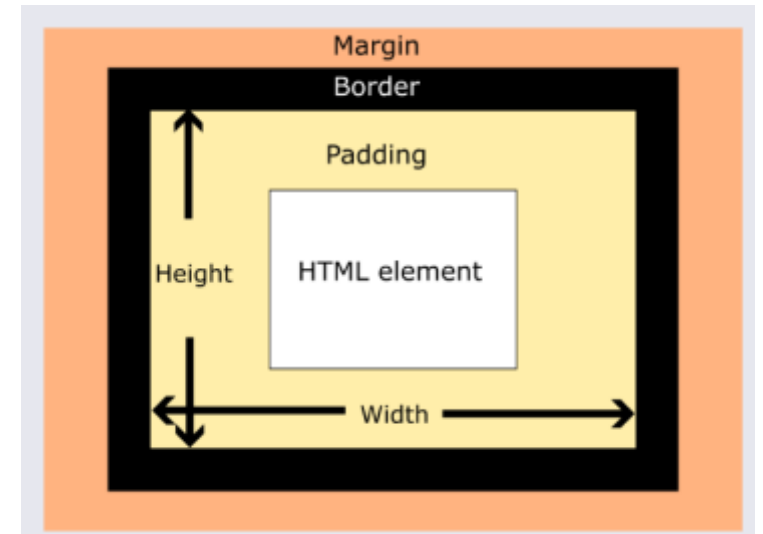
BOX MODEL

Usually when you set the width and height of an element using the CSS width and height properties, in reality you are only setting the width and height of the content area of that element.

The actual width and height of the element's box depends on several factors.

Total width = width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

Total height = height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom



BOX MODEL

```
<html>
  <head>
    <style>
      .main {
        font-size: 32px;
        font-weight: bold;
        text-align: center;
      }
      #box {
        padding-top: 30px;
        width: 300px;
        height: 80px;
        border: 30px solid green;
        margin: 50px;
        text-align: center;
        font-size: 28px;
        font-weight: bold;
      }
    </style>
  </head>
  <body>
    <div class="main">CSS Box-Model Property</div>
    <div id="box">Codegnan IT solutions</div>
  </body>
</html>
```



Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.



Floats

1. The CSS float property is used to position the elements to the left, right, of its container along with permitting the text and inline elements to wrap around it.
2. The float property defines the flow of content in the page. The remaining elements will be part of the flow if the element is removed from the normal flow of the content.
3. This property is ignored by the absolutely positioned elements. It can be written in a CSS file or can be directly specified in the style of an element.

Syntax:

`float: none|left|right|initial|inherit;`

Property values:

- none:** This is the default value & the element does not float.
- left:** Element floats on the left side of the container.
- right:** Element floats on the right side of the container.
- initial:** Element will be set to its default value.
- inherit:** Element inherits floating property of its parent property.



Floats

Example:

```
<html>
  <head>
    <title>Float</title>
  </head>
  <body>
    <div style="float:right">
      <div style="font-size:40px;color:#006400; float:inherit;"> codegnan</div>
    </div>
  </body>
</html>
```



Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.



Navigation Bar with Display & Float

Design of the navigation bar will be added with the properties of float and display to set the position of navigation bar

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  float: right;
}

li {
  display: inline;
}
</style>
</head>
<body>
<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>
</body>
</html>
```



Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.



Positioning

The position property specifies the type of positioning method used for an element

Syntax:

```
position: static|absolute|fixed|relative|sticky;
```

Static:

Default value. Elements render in order, as they appear in the document flow

absolute:

The element is positioned relative to its first positioned (not static) ancestor element

relative:

The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position

Sticky:

The element is positioned based on the user's scroll position

fixed:

The element is positioned relative to the browser window





Questions??



Every engineer has a tendency to tinker on a problem, lets answer few of them.



Layers and Z-index

1. In CSS, you can create layers of various divisions. CSS z-index property can be combined with the z-index property to create a layer effect.
2. The z-index property specifies the stacking order of an element. You can specify which element should come on top and which at the bottom.
3. A z-index property can help you to create more complex webpage layouts.

```
.div1{  
  position: absolute;  
  background-color: aquamarine;  
  border-color: red;  
  height: 3cm;  
  width: 30%;  
  left: 0px;  
  top: 0px;  
}
```

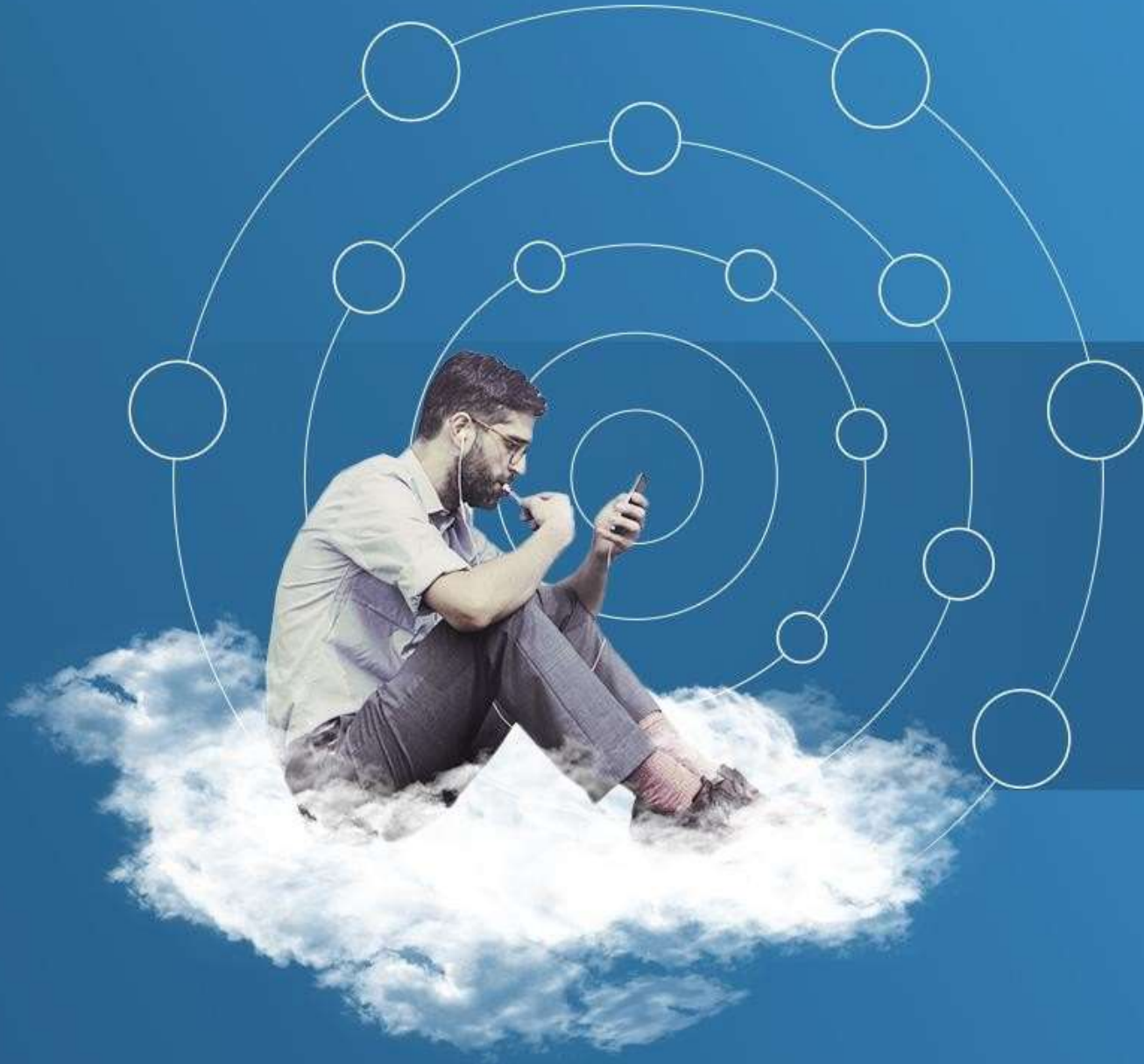
```
.div2{  
  position: absolute;  
  background-color: brown;  
  border-color: black;  
  height: 3cm;  
  width: 30%;  
  left: 100px;  
  top: 100px;  
  z-index: 1;  
}
```

```
.div3{  
  position: absolute;  
  background-color: rgb(181, 31, 144);  
  border-color: black;  
  height: 3cm;  
  width: 30%;  
  left: 200px;  
  top: 200px;  
}
```

Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.





Session : FLEX

FLEX

Definition and Usage:

1. The flex property is a shorthand for three individual properties: flex-grow, flex-shrink and flex-basis.
2. It sets the flexible length on items within a flex container.
3. If an element is not a flexible item, the flex property has no effect.

The initial Values are defined as :

Flex-grow : 0
Flex-shrink : 1
Flex-basis : auto

The shorthand notations of all the three elements are defined as :

Flex : initial | auto | 1



FLEX

The Flex values are defined for (flex : initial):

```
flex-grow : 0;  
flex-shrink : 1;  
flex-basis : auto;
```

The Flex values are defined for (flex : auto):

```
flex-grow : 1;  
flex-shrink : 1;  
flex-basis : auto;
```

The Flex values are defined for (flex : 1):

```
flex-grow : 1;  
flex-shrink : 1;  
flex-basis : 0;
```



FLEX

Properties:

1. flex-direction: row | column | row-reverse | column-reverse
2. flex-wrap : wrap | nowrap | wrap-reverse
3. flex-flow : wrap row | wrap column | etc,
4. justify-content : left | right | center | space-around



Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.





Session : GRID

GRID

CSS Grid Layout is a powerful tool for creating flexible and responsive two-dimensional layouts in web design. Unlike CSS Flexbox, which primarily deals with one-dimensional layouts, CSS Grid allows you to work seamlessly with both columns and rows.

Definition:

1. CSS Grid Layout enables you to divide a web page into major regions or define relationships between parts of a control using HTML primitives.
2. It aligns elements into columns and rows, similar to tables but with more flexibility and control

.Properties:

1. display : grid | inline-grid
2. column-gap : px | %
3. row-gap : px | %
4. gap : px px | % % | px % | % px
5. grid-template-columns : px px px | auto auto | px auto
6. grid-template-rows : px px px | auto auto | px auto



Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.



RESPONSIVE BREAKPOINTS

What are Breakpoints ?

Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes.

Mainly Devices are categorised into 5 types of Responsive Break points. They are :

1. Extra Small Devices
2. Small Devices
3. Medium Devices
4. Large Devices
5. Extra Large Devices



RESPONSIVE BREAKPOINTS

Device	Device Size	
Extra Small Devices	< 576px	
Small Devices	>= 576px	
Medium Devices	>=768px	
Large Devices	>=992px	
Extra Large Devices	>=1200px	



RESPONSIVE BREAKPOINTS

SYNTAX:

```
@media only screen and (max-width : value){  
    properties...  
}
```

EXAMPLE:

```
@media only screen and (max-width: 600px) {  
    body {  
        background-color: lightblue;  
    }  
}
```

Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.



CSS ANIMATIONS

- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times



CSS ANIMATIONS

To learn about CSS Animations you must follow the below properties:

- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode



CSS ANIMATIONS

@keyframes Rule

- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.
- To get an animation to work, you must bind the animation to an element

CSS ANIMATIONS

animation-name:

To start the animation initially we want to declare a value in animation-name

animation-delay:

- The animation-delay property specifies a delay for the start of an animation.

CSS ANIMATIONS

animation-iteration-count:

- The animation-iteration-count property specifies the number of times an animation should run

CSS ANIMATIONS

animation-direction:

The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles.

The animation-direction property can have the following values:

- normal - The animation is played as normal (forwards). This is default
- reverse - The animation is played in reverse direction (backwards)
- alternate - The animation is played forwards first, then backwards
- alternate-reverse - The animation is played backwards first, then forwards

CSS ANIMATIONS

animation-timing-function:

The animation-timing-function property specifies the speed curve of the animation.

The animation-timing-function property can have the following values:

- ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
- linear - Specifies an animation with the same speed from start to end
- ease-in - Specifies an animation with a slow start
- ease-out - Specifies an animation with a slow end
- ease-in-out - Specifies an animation with a slow start and end

CSS ANIMATIONS

animation-fill-mode:

The animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both)

The animation-fill-mode property can have the following values:

- none - Default value. Animation will not apply any styles to the element before or after it is executing
- forwards - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
- backwards - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
- both - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

Questions??

Every engineer has a tendency to tinker on a problem, lets answer few of them.

