CSS – CASCADING STYLE SHEETS

Nirali Vaghela

ITView Progressive Learning

CSS

❖ What is CSS

- CSS stands for **Cascading Style Sheets**. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML Documents. CSS is a simply designed language intended to simplify the process of making web pages presentable.
- CSS is easy to learn and understood, but it provides powerful control over the presentation of an HTML document.
- Now let's try to break the acronym:
  - Cascading: Falling of Styles
  - Style: Adding designs/Styling our HTML tags
  - Sheets: Writing our style in different documents

❖ CSS History

- 1994: First Proposed by Hakon Wium Lie on 10th October
- 1996: CSS was published on 17th November with influencer Bert Bos
- Later he became co-author of CSS
- 1996: CSS became official with CSS was published in December
- 1997: Created CSS level 2 on 4th November
- 1998: Published on 12th May

❖ CSS Versions

- CSS1
- CSS2
- CSS3
- CSS4

  Version 4 comes with:-
  - CSS-Pro
  - CSS-Mobile

❖ CSS Editors

- Some of the popular editors that are best suited to wire CSS code are as following:
  1. Atom
  2. Visual Studio Code (Best Editor)
  3. Brackets

4. Espresso(For Mac OS User)
5. Notepad++(Great for HTML & CSS)
6. Komodo Edit (Simple)
7. Sublime Text

❖ What does CSS do

- You can add new looks to your old HTML documents.
- You can completely change the look of your website with only a few changes in CSS code.

❖ Why use CSS / Advantages of CSS

- **CSS saves time** – You can write CSS once using external CSS and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** – If you are using CSS, you do not need to write HTML tag attributes every time. Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem. It was a W3C recommendation.
  Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** – CSS provides more detailed attributes than plain HTML to define the look and feel of the website, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** – Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.
- **Offline Browsing:** CSS can store web applications locally with the help of an offline cache. Using this we can view offline websites.

❖ Disadvantages of CSS

- Cross-browser related issues
- Vulnerable
- Issues due to multiple levels
- Lack of security
- Fragmentation

❖ Conclusion

- CSS gives the power to the web designer so that extensive changes could be given to the web layout of all pages in a single website by making use of just a single file. It helps in designing light and a creative website with high responsiveness and which impresses the audience when displayed. Therefore, it is an integral part of the websites today which should not be overlooked.

❖ Characteristics of CSS

- The major characteristics of CSS include styling rules which are interpreted by the client browser and applied to various elements in your document. Major characteristics include:
  1. A style rule consists of a selector component and a declaration block component.
  2. The selector is used to point to the HTML component which you want to get styled.
  3. Inside the declaration block, one or more declarations are contained along with semicolons.
  4. Every declaration which is put has a CSS property name, a semicolon, and a value. For example, color is the property, and the value is red in color. Font size is the property, and the 15px is the value.
  5. CSS declaration ends with a semicolon, and these blocks are surrounded by curly braces.
  6. CSS selectors are the ones which are used to find HTML elements which are based on the element name, id, attribute, class and more.
  7. One unique element will be selected by the ID of an element.
  8. If you wish to select the particular element with a specific id, the # function along with the id attribute should be used.
  9. If you wish to select the elements with a specific class, the period character along with the name class should be written.
  10. **Universal selector:** If you are not interested in choosing the elements of a certain type, the universal selector simply matches the element name.
  11. **Element selector:** These selectors choose the element based on the element name.
  12. **Descendent selector:** When a particular element lies inside another element, then it is called as the descendent selector.

13. **ID selector:** This selector uses the id of the HTML element so that a specific element could be selected.
14. **Class selectors:** It selects the element with a specific class attribute.
15. **Grouping selectors:** It will be a good option to group the selectors so as to minimize the code. Each selector, along with a comma, should be used to group the selectors.

❖ CSS Syntax

- A CSS rule set contains a selector and a declaration block.

Syntax

```
Selector
{
        Property 1 : value;
        Property 2 : value;
        Property 3 : value;
}
```

Example

```
h1  - Selector
{
     color: red;
     font-size: 11px;
}
```

Declaration Block

- **Selector:** Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> or class, id, etc.
- **Declaration Block:** The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:
    1. color: red;
    2. font-size: 11 px;

  Each declaration contains a property name and value, separated by a colon.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.
- **Property:** A Property is a type of attribute of HTML element. It could be color, border etc.
- **Value:** Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

❖ Types of CSS

- CSS is added to HTML pages to format the document according to information in the style sheet. There are three ways to insert CSS in HTML documents.

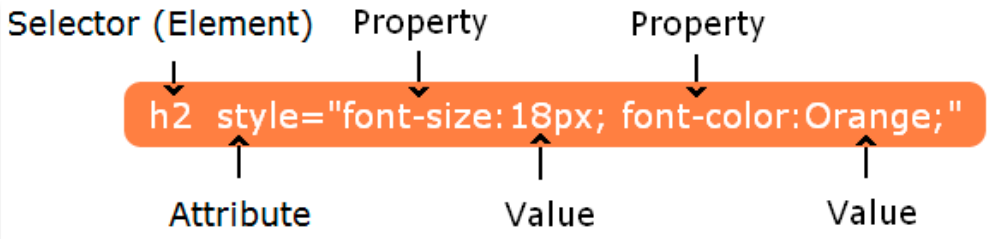
1. **Inline CSS**
2. **Internal CSS**
3. **External CSS**

Inline CSS

- We can apply CSS in a single element by inline CSS technique.
- The inline CSS is also a method to insert style sheets in HTML document. If you want to use inline CSS, you should use the style attribute to the relevant tag.
- The style attribute can contain any CSS property.
- Before CSS this was the only way to apply styles
- Not an efficient way to write as it has a lot of redundancy
- Self-contained
- Uniquely applied on each element
- The idea of separation of concerns was lost
- In-line CSS style consists set of rules in 4 part:
    - Selector (Element)
    - Style (Attribute)
    - Property and
    - Value

How to write Inline CSS style

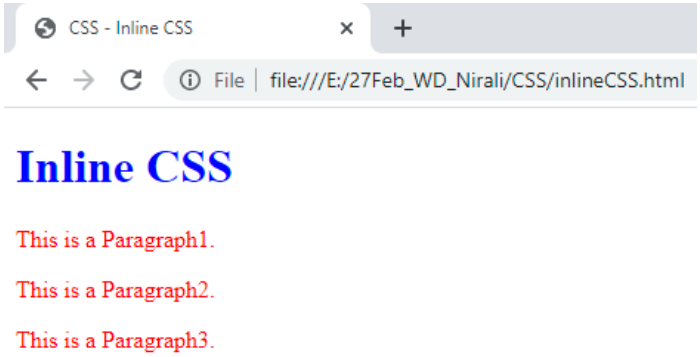
Selector is normally HTML element that element you want to assign CSS style. And style is attribute to assign CSS property and set suitable value.



Example

```
<body>
   <h1 style="color: blue;">Inline CSS</h1>
   <p style="color: red;">This is a Paragraph1.</p>
   <p style="color: red;">This is a Paragraph2.</p>
   <p style="color: red;">This is a Paragraph3.</p>
</body>
```

Output



Disadvantages of Inline CSS

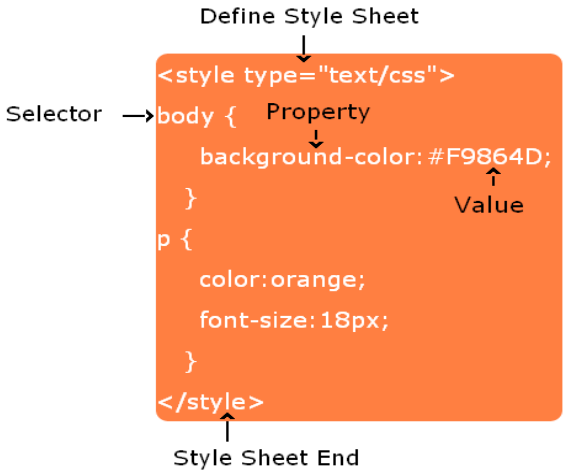
- You cannot use quotations within inline CSS. If you use quotations the browser will interpret this as an end of your style value.
- These styles cannot be reused anywhere else.
- These styles are tough to be edited because they are not stored at a single place.
- It is not possible to style pseudo-codes and pseudo-classes with inline CSS.
- Inline CSS does not provide browser cache advantages.

Internal CSS

- The internal style sheet is used to add a unique style for a single document. It is defined in <head> section of the HTML page inside the <style> tag.
- But the idea of separation of concerns still lost
- Internal CSS style consists set of rules in 3 part:
    - Selector (element, class, id)
    - Property and
    - Value

How to write Internal CSS Style

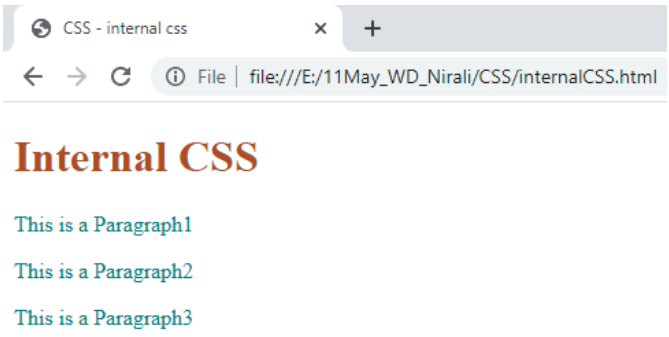
Selector is normally HTML element that element you want to assign CSS style. All elements within web page that elements assign CSS properties and set suitable values.



Example

```
<head>
    <style>
      h1{ color: sienna;}
      p { color: teal; }
   </style>
</head>
<body>
   <h1 >Internal CSS</h1>
   <p>This is a Paragraph1</p>
   <p>This is a Paragraph2</p>
   <p>This is a Paragraph3</p>
</body>
```

Output

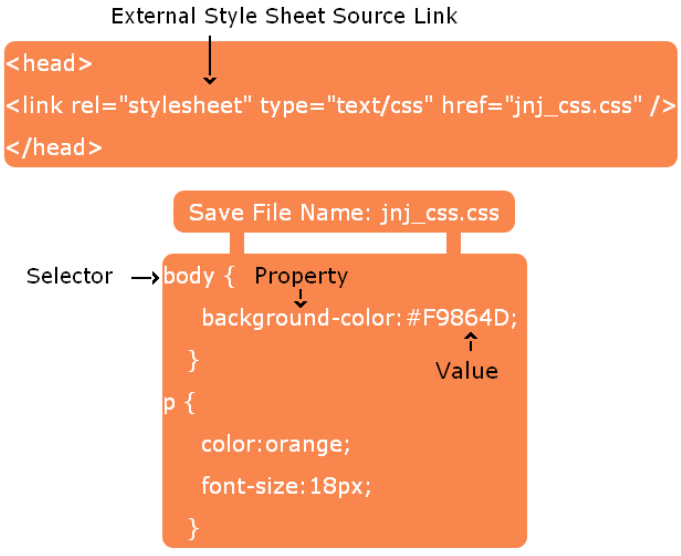


External CSS

- The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file.
- It uses the <link> tag on every pages and the <link> tag should be put inside the head section.
- Reference is added
- File saved with .css extension
- The idea of separation of concerns is maintained
- Uniquely applied to each document
- External style sheet consists set of rules in 4 part:
    - External Source link
    - Selector (element, class, id)
    - Property and
    - Value

How to write External Stylesheet

External stylesheet linked to a web page.
Selector is normally HTML element (or class, id) to assign CSS properties and set suitable values.



Example

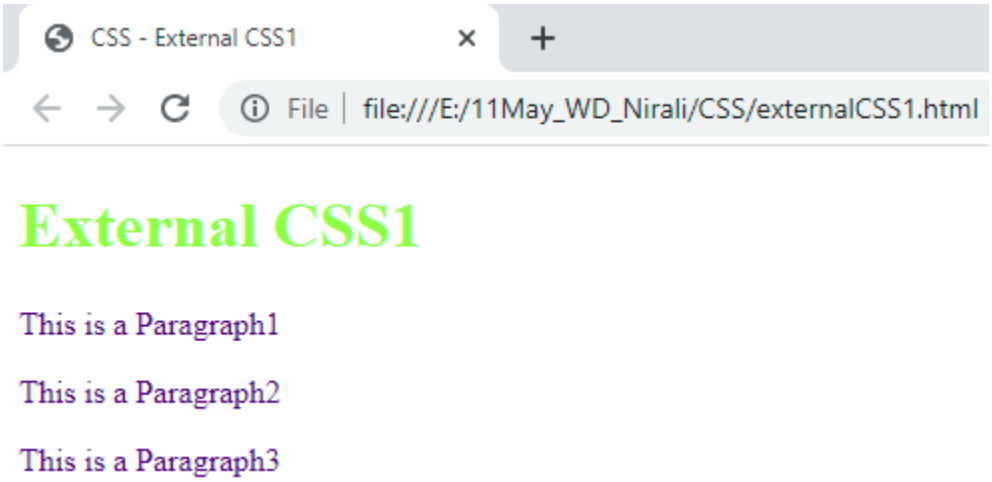
mystyle.css

```
p
{
   color: indigo;
}
h1
{
   color: greenyellow;
}
```

externalCSS1.html

```
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>
   <h1 >External CSS1</h1>
   <p>This is a Paragraph1</p>
   <p>This is a Paragraph2</p>
   <p>This is a Paragraph3</p>
</body>
</html>
```

Output



Cascading Order

What style will be used when there is more than one style specified for an HTML element?

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

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

CSS The !important Rule

What is !important?

The !important rule in CSS is used to add more importance to a property/value than normal.

In fact, if you use the !important rule, it will override ALL previous styling rules for that specific property on that element and **the !important keyword must be placed at the end of the line,** immediately before the semicolon.

Example

```
<head><style>
      .para{color: blue;}
      p{color: red !important;}
</style></head>
<body>
   <h1>Example of !important Property</h1>
   <p class="para">This is a Paragraph.</p>
</body>
```

Output

Important About !important

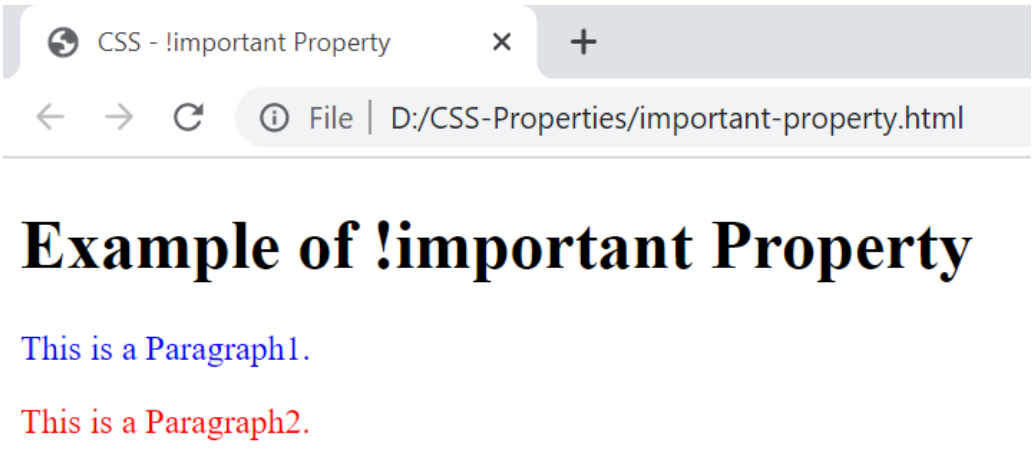
The only way to override an !important rule is to include another !important rule on a declaration with the same (or higher) specificity in the source code - and here the problem starts! This makes the CSS code confusing and the debugging will be hard, especially if you have a large style sheet!

Here we have created a simple example. It is not very clear, when you look at the CSS source code, which color is considered most important.

Example

```
 <head><style>
      p
      {
         color: red !important;
      }
      .para
      {
         color: blue !important;
      }
</style></head>
<body>
   <h1>Example of !important Property</h1>
   <p class="para">This is a Paragraph1.</p>
   <p>This is a Paragraph2.</p>
</body>
```

Output



❖ CSS Comments

- CSS comments are generally written to explain your code. It is very helpful for the Programmers who reads your code so that they can easily understand the code.
- Comments are ignored by browsers.
- Helps to debug our code
- Comments are single or multiple lines statement and written within **/*.....*/** .

Example

Example of Single Line Comment

```
P
{
        Color:red;
        /*Font-size:20px;*/
}
```

Example of Multiple Line Comment

```
P
{
        /*Color:red;
        Font-size:20px;*/
}
```

❖ CSS Selector

- **CSS selectors** are used to select the content you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.
- There are several different types of selectors in CSS.
    1. **CSS Element Selector**
    2. **CSS Class Selector**
    3. **CSS Id Selector**
    4. **CSS Universal Selector**
    5. **CSS Group Selector**
    6. **CSS Attribute Selector**
    7. **CSS Combinators**

- CSS Combinators clarifies the relationship between two selectors, whereas the selectors in CSS are used to select the content for styling.
- There can be more than one simple selector in a CSS selector, and between these selectors, we can include a combinator. Combinators combine the selectors to provide them a useful relationship and the position of content in the document.
- There are four types of combinators in CSS that are listed as follows:
    - **i. Descendant Selector (space)**
    - **ii. Child selector (>)**
    - **iii. Adjacent sibling selector (+)**
    - **iv. General sibling selector (~)**

8. **CSS Pseudo Class Selector**
9. **CSS Pseudo Element Selector**

CSS Element Selector

- The element selector selects the HTML element by name.

Example

```
<head>
        <style>
     h1
     {
        color: rgb(200, 40, 40);
     }
     p
     {
        color: darkmagenta;
     }
   </style>
</head>
<body>
   <h1>Element Selector Example</h1>
   <p>This is a Paragraph1.</p>
   <p>This is a Paragraph2.</p>
</body>
```

Output

CSS Class Selector

- The CSS Class selector is one of the most helpful selectors of all the selectors.
- CSS class is a selector to assign class name either one or group of element and apply specific styles.
- **We can use the same name class numbers of time with any html element which helps us in reusing the styles and avoids unnecessary repetition.**
- Using class selector you can easily modify element styles.
- CSS class selector define using class attribute with value (user define class name).
- CSS class selector identify using **(".") dot sign** concatenate with user define class name.
- **Class name cannot start with number.**

CSS class advantages

One or more CSS class assign to a every single element separate by single space.

Another advantages is assign same class one or more element with same styles.

➢ **same class name with multiple element syntax is like,**

Syntax:

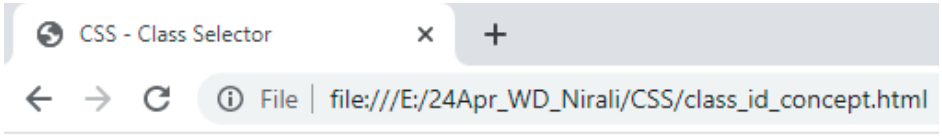
```
.classname
{
        Propertyname:value;
}
```

Example

```
<head>
<style>
      .para1
      {   color: red;  }
      .para
      {   color: dodgerblue; }
   </style>
</head>
<body>
   <h1 class="para1">Class Selector Example</h1>
   <p class="para">This is a Paragraph1.</p>
   <p class="para">This is a Paragraph2.</p>
   <p class="para1">This is a Paragraph3.</p>
</body>
```

This style applies to all the HTML elements having an attribute value for the class as 'para' and 'para1'. An element having the same class attribute value helps us in reusing the styles and avoids unnecessary repetition.

Output



Class Selector Example

This is a Paragraph1.

This is a Paragraph2.

This is a Paragraph3.

- **CSS Class allows you to set a particular style for any HTML elements using CSS class. If you assign same class multiple element with assign unique style each element, syntax is like,**

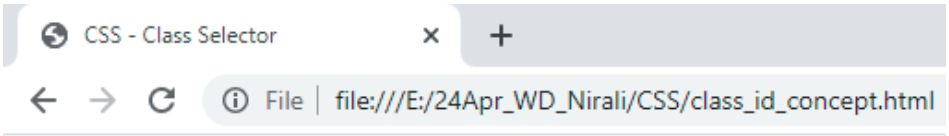
Syntax

```
element.classname
{
        Propertyname:value;
}
```

Example

```
<style>
      p.para1
      {
         color: red;
      }
   </style>
</head>
<body>
   <h1 class="para1">Class Selector Example</h1>
   <p class="para1">This is a Paragraph3.</p>
</body>
</html>
```

Output



Class Selector Example

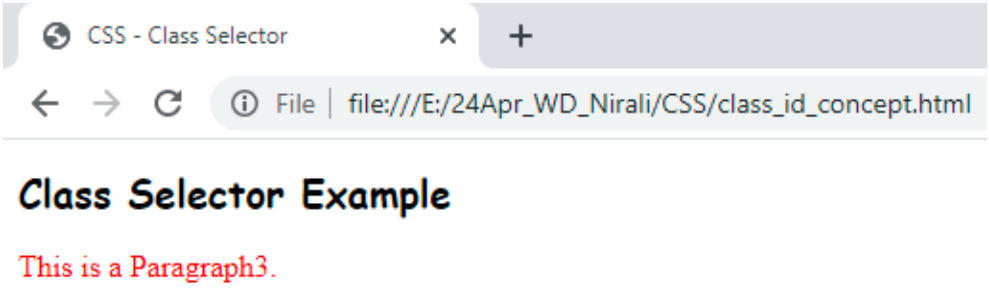
This is a Paragraph3.

- **One or more CSS class assign to a every single element separate by single space.**

Example

```
<head>
<style>
      .para1
      {  color: red; }
      .sample
      {  font-size: 20px; }
      .test
      {  font-family: cursive; }
</style>
</head>
<body>
   <h1 class="para  sample  test">Class Selector Example</h1>
   <p class="para1">This is a Paragraph3.</p>
</body>
```

Output



In above example, three classes (para, sample, test) style apply to H1 element.

CSS ID Selector

- The id selector selects the id attribute of an HTML element to select a specific element. **An id is always unique within the page so it is chosen to select a single, unique element.**
- CSS ID selector define using id attribute with value (user define id name).
- CSS ID selector identify using **hash("#")** concatenate with user define id name.

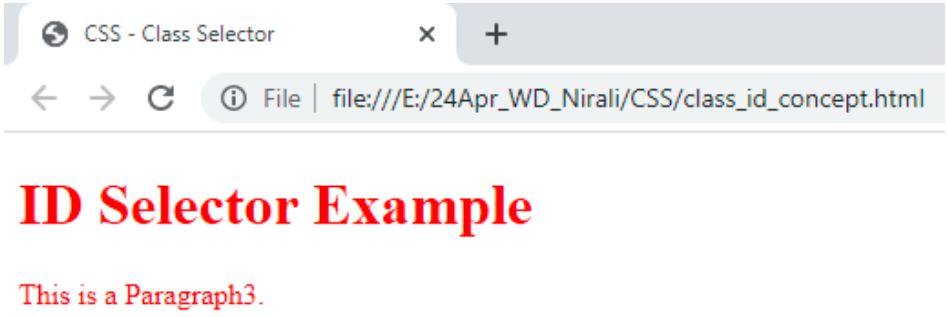
Syntax

```
#idname
{
        Propert:value;
}
```

Example

```
<head>
<style>
      #para
      {
         color: red;
      }
      #para1
      {
         color: red;
      }
</style>
</head>
<body>
   <h1 id="para">ID Selector Example</h1>
   <p id="para1">This is a Paragraph3.</p>
</body>
```

Output



- **CSS class Vs CSS ID**

  This difference is important to know where is use CSS ID and CSS Class. Because both work are same. CSS ID is a use for "unique identifier an element". It means CSS ID selector can be used only one time in a document while class selector can be used multiple times in a document. Class and ID have not any strong rules to where is use CSS ID or CSS Class.

CSS Universal Selector

- The universal selector is used as a wildcard character. It selects all the elements on the pages.
- We declare a universal selector with the help of an **(*) asterisk** at the beginning of the curly brace.

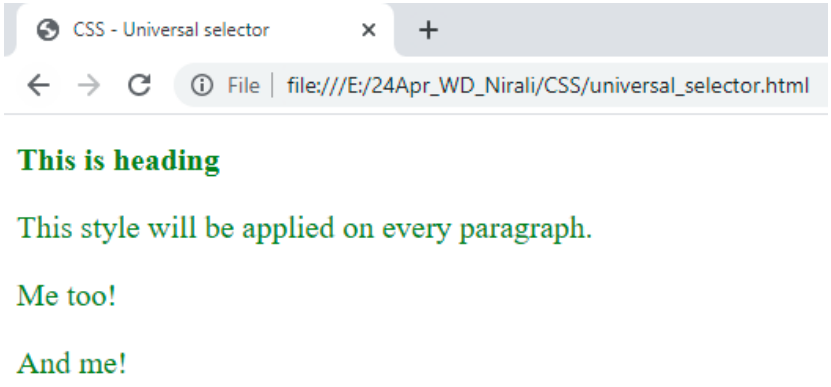
Syntax

```
*
{
        Property:value;
}
```

Example

```
<head><style>
      * { color: green;  font-size: 20px;  }
   </style> </head>
<body>
   <h2>This is heading</h2>
   <p>This style will be applied on every paragraph.</p>
   <p id="para1">Me too!</p>
   <p>And me!</p> </body>
```

Output



CSS Group Selector

- The grouping selector is used to select all the elements with the same style definitions.
- Grouping selector is used to minimize the code. **Commas (,)** are used to separate each selector in grouping.

Syntax

```
Selector1, selector2, selector3
{
        Property:value;
}
```

Example

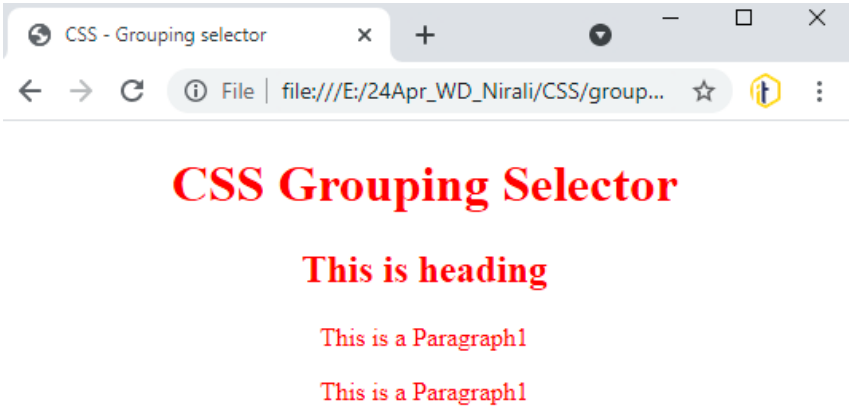
Let's see the CSS code without group selector.

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

As you can see, If you would like to apply same CSS properties to multiple elements then It can be grouped in following ways:

```
<head>
   <style>
      h1,h2,p {
        color: red;
        text-align: center;
      }
   </style>
</head>
<body>
   <h1>CSS Grouping Selector</h1>
   <h2>This is heading</h2>
   <p>This is a Paragraph1</p>
   <p>This is a Paragraph1</p>
</body>
```

Output



CSS Attribute Selector

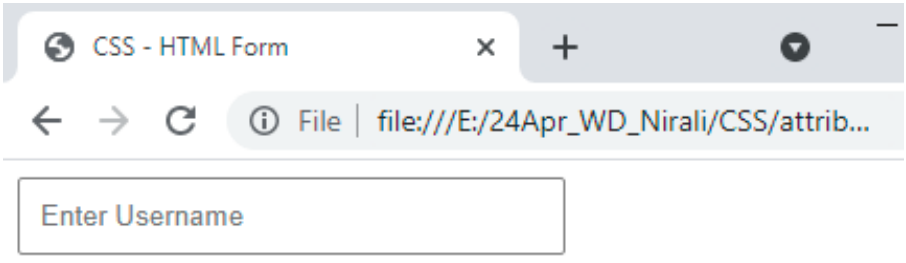
- The CSS Attribute selector styles content according to the attribute and the attribute value mentioned in the square brackets. No spaces can be present ahead of the opening square bracket.

Example

- Attribute selector with Value

```
<head>
<style>
      .formContainer
      {
         width: 50%;
         float: left;
      }
      input[type="text"]
      {
         width: 100%;
         padding: 10px;
         box-sizing: border-box;
         border-radius: 2px;
         border: 1px solid grey;
         margin-bottom: 10px;
         outline: none;
      }
</style>
</head>
<body>
   <div class="formContainer">
      <input type="text" placeholder="Enter Username" >
   </div>
</body>
```

Output



In the Above Example, CSS code would be a match for the <input type=”text”>. Similarly, if the value of the attribute ‘type’ changes, the Attribute selector would not match it. For example, the selector would not match the attribute if the value of ‘type’ changed from ‘text’ to ‘submit’.

- If the attribute selector is declared with only the attribute, and no attribute value, then it will match to all the HTML elements with the attribute 'type', regardless of whether the value is 'text' or 'submit'.

Example

```
<head>
<style>
      input[type]
      {
         width: 100%;
         padding: 10px;
         box-sizing: border-box;
         border-radius: 2px;
         border: 1px solid grey;
         margin-bottom: 10px;
         outline: none;
      }
</style>
</head>
```

This will help us to target the attribute only, regardless of the element. In our example, it will target based on the attribute 'type', regardless of the element 'input'. CSS Selectors help us to simplify our code and enable us to utilize and reuse the same CSS code for various HTML elements. They help us in styling specific segments and portions of our webpage.

CSS Descendant Selector

- The CSS descendant selector is used to match the descendant elements of a particular element. The word Descendant indicates nested anywhere in the HTML Document. It can be a direct child or deeper than five levels, but it will still be referred to as a descendant.
- The Descendant combinator is represented using a **single space**. It combines two selectors in which the first selector represents an ancestor (parent, parent's parent, etc.), and the second selector represents descendants. The elements matched by the second selector are selected if they have an ancestor element that matches the first selector.
- **In simple terms,** The descendant selector matches all elements that are descendants of a specified element.
- Descendant Selector includes elements that are not direct elements too like a child, grandchild and etc.

Syntax

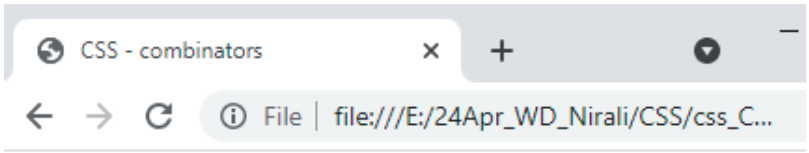
```
element element
{
        Property:value;
}
```

Example

```
<head>
<style>
div p
      {
         color: red;
      }
</style>
</head>
<body>
<div>
      <p>This is a Paragraph1</p>
      <span><p>This is a Paragraph3</p></span>
      <p>This is a Paragraph4</p>
</div>
      <h2>hedaing Text</h2>
       <p>This is a Paragraph6</p>
       <p>This is a Paragraph7</p>
</body>
```

This example selects all <p> elements inside elements:

Output



This is a Paragraph1

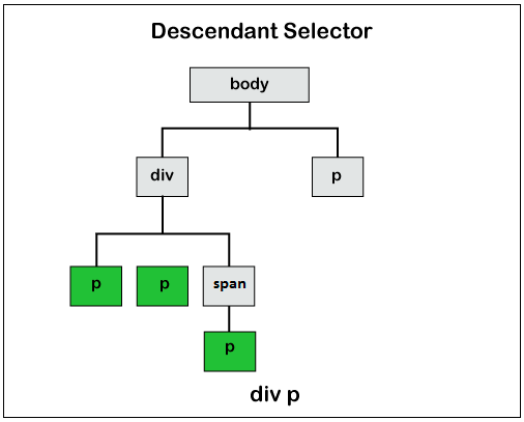
This is a Paragraph3

This is a Paragraph4

hedaing Text

This is a Paragraph6

This is a Paragraph7

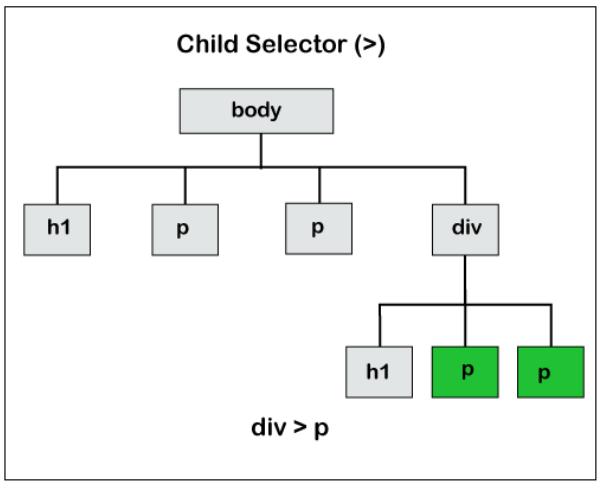


CSS Child Selector

- It selects the direct descendant of the parent. This combinator only matches the elements that are the immediate child in the document tree. It is stricter as compared to the descendant selector because it selects the second selector only when the first selector is its parent.
- The Chile combinator is represented using a Greater than **(>)** Symbol.
- In simple terms, The child selector selects all elements that are the children of specified element.
- The parent element must always be placed at the left of the **">"**. If we remove the greater than **(>)** symbol that designates this as a child combinator, then it will become the descendant selector.

Syntax

```
element > element
{
        Property:value;
}
```

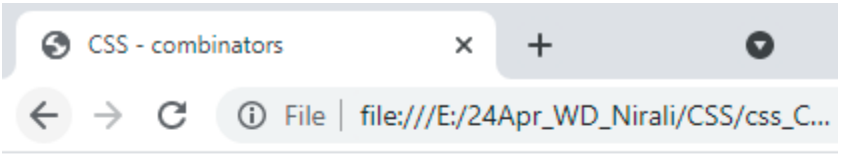


Example

```
<head>
<style>
Div > p
      {
         color: red;
      }
</style>
</head>
<body>
<div>
      <p>This is a Paragraph1</p>
      <span><p>This is a Paragraph3</p></span>
      <p>This is a Paragraph4</p>
</div>
      <h2>hedaing Text</h2>
       <p>This is a Paragraph6</p>
       <p>This is a Paragraph7</p>
</body>
```

This example selects all <p> elements that are children of a element.

Output



This is a Paragraph1

This is a Paragraph3

This is a Paragraph4

hedaing Text

This is a Paragraph6

This is a Paragraph7

CSS 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 Adjacent Sibling combinator is represented using a Greater than **(+)** Symbol.

Syntax

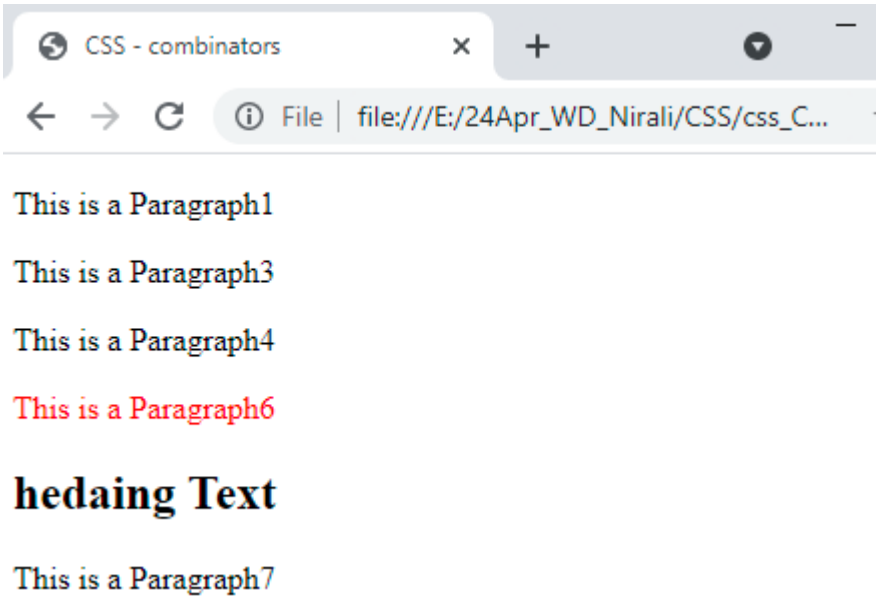
```
element > element
{
        Property:value;
}
```

Example

```
<head>
<style>
div+p
{
    color: red;
}
</style>
</head>
<body>
   <div>
      <p>This is a Paragraph1</p>
      <span><p>This is a Paragraph3</p></span>
      <p>This is a Paragraph4</p>
   </div>
   <p>This is a Paragraph6</p>
   <h2>hedaing Text</h2>
   <p>This is a Paragraph7</p>
</body>
```

This example selects the first <p> element that are placed immediately after elements

Output



CSS General Sibling Selector

- The general sibling selector selects all elements that are next siblings of a specified element.
- It is useful when we have to select the siblings of an element even if they are not adjacent directly.
- The General Sibling combinator is represented using a tlide **(~)** Symbol.

Syntax

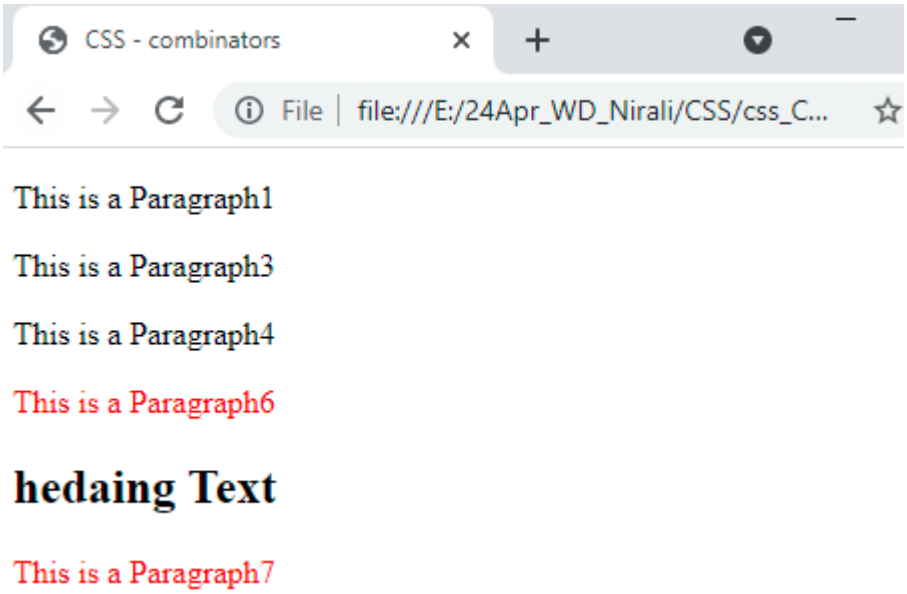
```
element ~ element
{
        Property:value;
}
```

Example

```
<head>
<style>
div+p
{
    color: red;
}
</style>
</head>
<body>
   <div>
      <p>This is a Paragraph1</p>
      <span><p>This is a Paragraph3</p></span>
      <p>This is a Paragraph4</p>
   </div>
   <p>This is a Paragraph6</p>
   <h2>hedaing Text</h2>
   <p>This is a Paragraph7</p>
</body>
```

This example selects all <p> elements that are next siblings of elements:

Output



CSS Pseudo class selectors

- A pseudo-class can be defined as a keyword which is combined to a selector that defines the special state of the selected elements. It is added to the selector for adding an effect to the existing elements based on their states. For example, The **":hover"** is used for adding special effects to an element when the user moves the cursor over the element.
- The names of the pseudo-class are not case-sensitive.
- A pseudo-class starts with a **colon (:)**

  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,etc.

Syntax

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

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

Syntax

```
classname:pseudo-class
{
        Property:value;
}
```

Here is the List of pseudo classes:

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children

:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last

		child
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

CSS Pseudo element selector

- A pseudo-class can be defined as a keyword which is combined to a selector that defines the special state of the selected elements. Unlike the pseudo-classes, the pseudo-elements are used to style the specific part of an element, whereas the pseudo-classes are used to style the element.
- As an example, a pseudo-element can be used to style the first letter or the first line of an element. The pseudo-elements can also be used to insert the content after or before an element.
- A pseudo-class starts with a **double colon notation (::).** In CSS3, the double colon replaced the single colon notation for pseudo-elements. It was an attempt

from W3C to differentiate between the pseudo-elements and pseudo-classes. So, it is recommended to use **double colon notation (::pseudo-element)** instead of using single-colon notation **(:)**.

Syntax

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

➢ Pseudo-elements and CSS Classes

Pseudo-classes can be combined with CSS classes:

Syntax

```
Classname::pseudo-element
{
        Property:value;
}
```

Here is the List of pseudo elements:

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::marker	::marker	Selects the markers of list items
::selection	p::selection	Selects the portion of an element that is selected by a user

❖ CSS Properties

CSS Text Properties

- CSS text formatting properties is used to format text and style text.
- CSS text formatting include following properties:
  - **color:** This property is used to set the color of a text.
  - **direction:** This property is used to set the text direction.
  - **letter-spacing:** This property is used to add or subtract space between the letters that make up a word.
  - **word-spacing:** This property is used to add or subtract space between the words of a sentence.
  - **text-indent:** This property is used to indent the text of a paragraph.
  - **text-align:** This property is used to align the text of a document.
  - **text-decoration:** This property is used to underline, overline, and strikethrough text.
  - **text-transform:** This property is used to capitalize text or convert text to uppercase or lowercase letters.
  - **white-space:** This property is used to control the flow and formatting of text.

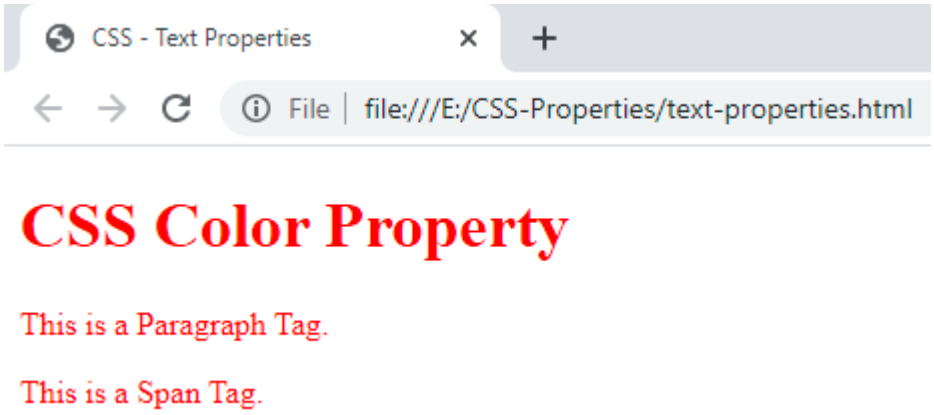
➢ Color Property

- color property is used to set the color of the text.
- **color can be set by using the name like "red", hex value like "#ff0000" or by its RGB value like "rgb(255, 0, 0)".**
- The default text color for a page, you can define in the body selector.

Example

```
<head> <style>
      h1{ color: red; }
      p { color: rgb(255, 0, 0); }
      span { color: #ff0000; }
   </style></head>
<body>
   <h1>CSS Color Property</h1>
   <p>This is a Paragraph Tag.</p>
   <span>This is a Span Tag.</span></body>
```

Output



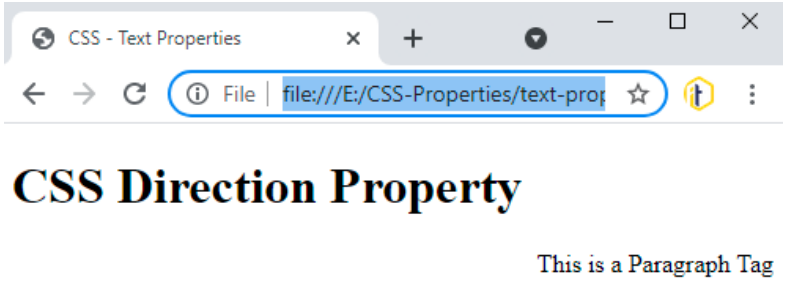
➢ Direction Property

- Text direction property is used to set the direction of the text.
- **The direction can be set by using rtl : right to left, ltr : left to right .**
- **Left to right** is the **default direction** of the text.

Example

```
<head>
<style>
      p
      {
         direction: rtl;
      }
   </style>
</head>
<body>
   <h1>CSS Direction Property</h1>
   <p>This is a Paragraph Tag</p>
</body>
</html>
```

Output



➢ Letter Spacing Property

- This property is used to specify the space between the characters of the text.
- **Possible values are normal or a number(px,rem,em,pt,%) specifying space.**
- **Normal** is the **default** value of letter spacing property.

Example

```
<head>
<style>
      p
      {
         letter-spacing: 4px;
      }
   </style>
</head>
<body>
   <h1>CSS Letter Spacing Property</h1>
   <p>This is a Paragraph Tag</p>
</body>
```

Output

➢ Word Spacing Property

- Word spacing is used to specify the space between the words of the line.
- **Possible values are normal or a number(px,rem,em,pt,%) specifying space.**
- **Normal** is the **default** value of word spacing property.

Example

```
<head><style>
      p{
         word-spacing:10px; }
   </style>
</head>
<body>
   <h1>CSS Word Spacing Property</h1>
   <p>This is a Paragraph Tag</p>
</body>
```

Output

➢ Text Indent Property

- Text indentation property is used to indent the first line of the paragraph.
- **The size can be in px, rem, pt, em, %.**

Example

```
<head><style>
      p {
         text-indent: 40px;
      }
   </style>
</head>
<body>
   <h1>CSS Text Indent Property</h1>
   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quia atq
ue laborum minima inventore, aliquam ullam eius magnam temporibus
nobis, accusamus quam eveniet ducimus, facere illo facilis quisquam ab
 animi. Sit!</p>
</body>
```

Output

➢ Text Align Property

- Text alignment property is used to set the horizontal alignment of the text.
- **The text can be set to left, right, centered and justified alignment.**
- In justified alignment, line is stretched such that left and right margins are straight.

Example

```
<head>
<style>
      .test{text-align: left;}
      .test1{text-align: center;}
      .test2{text-align: right;}
      .test3{text-align: justify;}
   </style>
</head>
<body>
   <h1>CSS Text Align Property</h1>
   <p class="test">This ia a Paragraph1.</p>
   <p class="test1">This ia a Paragraph2.</p>
   <p class="test2">This ia a Paragraph3.</p>
   <p class="test3">Lorem ipsum dolor sit amet consectetur adipisi
cing elit. Quia atque laborum minima inventore, aliquam ullam eius
magnam temporibus nobis, accusamus quam eveniet ducimus, facer
e illo facilis quisquam ab animi. Sit!</p>
</body>
```

Output

➢ Text Decoration Property

- Text decoration is used to add or remove decorations from the text.
- **Text decoration can be underline, overline, line-through or none.**
- The value text-decoration: none; is often used to remove underlines from links

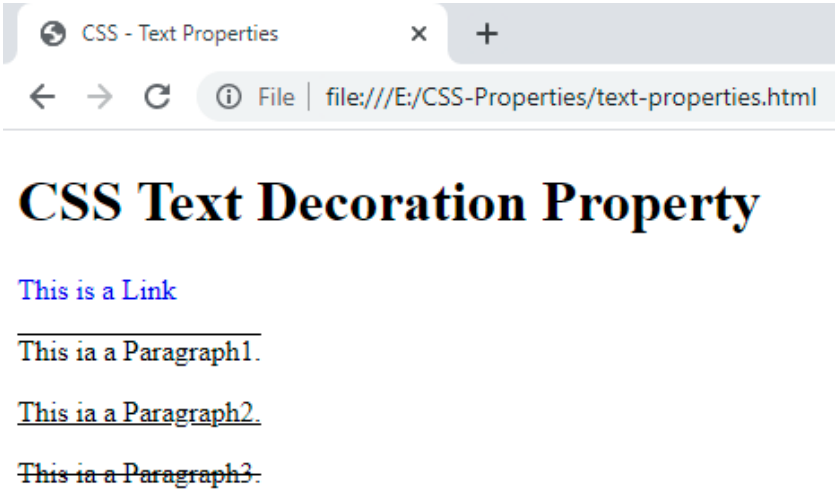
NOTE: In CSS3, the text-decoration property is a shorthand property for text-decoration-line, text-decoration-color, and text-decoration-style, but this is currently not supported in any of the major browsers.

It is not recommended to underline text that is not a link, as this often confuses the reader.

Example

```
<head><style>
   a{text-decoration: none;}
   .test{text-decoration: overline;}
   .test1{text-decoration: underline;}
   .test2{text-decoration: line-through;}
</style>
</head><body>
   <h1>CSS Text Decoration Property</h1>
   <a href="#">This is a Link</a>
   <p class="test">This ia a Paragraph1.</p>
   <p class="test1">This ia a Paragraph2.</p>
   <p class="test2">This ia a Paragraph3.</p></body>
```

Output



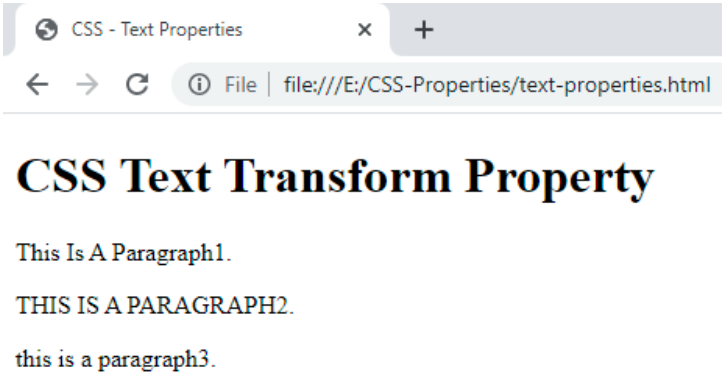
➢ Text Transform Property

- Text transformation property is used to change the case of text, uppercase or lowercase.
- **Text transformation can be uppercase, lowercase, capitalize or none .**
- Capitalize is used to change the first letter of each word to uppercase.
- **None** is the **default** value of Text Transform property

Example

```
<head>
<style>
      .test{text-transform: capitalize;}
      .test1{text-transform: uppercase;}
      .test2{text-transform: lowercase;}
   </style>
</head>
<body>
   <h1>CSS Text Transform Property</h1>
   <p class="test">This ia a Paragraph1.</p>
   <p class="test1">This ia a Paragraph2.</p>
   <p class="test2">This ia a Paragraph3.</p>
</body>
```

Output



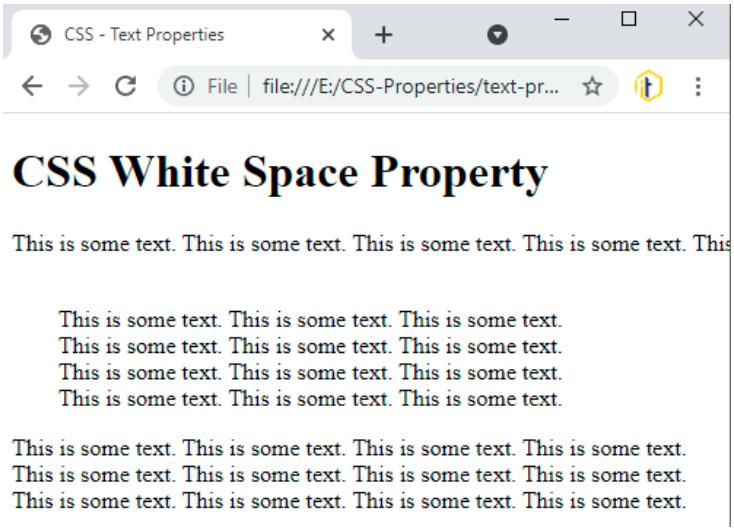
➢ White Space Property

- The white-space property specifies how white-space inside an element is handled.
- **White space can be no-wrap, pre, normal .**
- **normal:** Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary. This is **default** value.
- **no-wrap:** Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a
 tag is encountered.
- **pre:** Whitespace is preserved by the browser. Text will only wrap on line breaks. Acts like the <pre> tag in HTML

Example

```
<head><style> .test{white-space: nowrap;}.test1{white-space: pre;}
      .test2{white-space: normal;}</style></head><body>
   <h1>CSS White Space Property</h1>
   <p class="test">
      This is some text. This is some text. This is some text.
      This is some text. This is some text. This is some text.
      This is some text. This is some text. This is some text.
      This is some text. This is some text. This is some text.</p>
   <p class="test1">
      This is some text. This is some text. This is some text.
      This is some text. This is some text. This is some text.
      This is some text. This is some text. This is some text.
      This is some text. This is some text. This is some text.</p>
   <p class="test2">
      This is some text. This is some text. This is some text.
      This is some text. This is some text. This is some text.
      This is some text. This is some text. This is some text.
     This is some text. This is some text. This is some text.</p></body>
```

Output



➢ Line Height Property

- This property is used to set the space between the lines.
- **The size can be in px, rem, pt, em, %.**
- **Normal** is the **default** value of Line Height property

Example

```
<head>
<style>
      .test{line-height: 40px;}
   </style>
</head>
<body>
   <h1>CSS Line Height Property</h1>
   <p class="test">
      Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo r
atione soluta et corrupti ut corporis consectetur pariatur, saepe impedit.
Perspiciatis sunt quia velit impedit possimus atque consectetur molestias
. Quam, dolorem?
   </p>
</body>
```

Output

CSS Font Properties

- CSS Font property is used to control the look of texts. **Fonts** set a theme for your webpage. Fonts also help you connect with your audience. By the use of CSS font property you can change the text size, style and more. We can also resize our font using percentage.
- CSS Font property include following properties:
  - **font-family:** This property is used to change the face of a font.
  - **font-style:** This property is used to make a font italic or oblique.
  - **font-variant:** This property is used to create a small-caps effect.
  - **font-weight:** This property is used to increase or decrease how bold or light a font appears.
  - **font-size:** This property is used to increase or decrease the size of a font.
  - **font:** This property is used as shorthand to specify a number of other font properties.

Font Family Property

- It is used to set the font type of an HTML element.
- CSS font family can be divided in two types:
  - **I. Generic family:** It includes Serif, Sans-serif, cursive, fantasy and Monospace.
  - **II. Font family:** It specifies the font family name like Arial, New Times Roman etc.
- **Serif**: Serif fonts include small lines at the end of characters. Example of serif: Times new roman, Georgia etc.

- **Sans-serif**: A sans-serif font doesn't include the small lines at the end of characters. Example of Sans-serif: Arial, Verdana etc.

- Fonts may be named specifically or a generic font family name can be used. Start with the font you want, and always end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.
- **When multiple font names are specified using commas,** they are read in a descending order looking for a first match. Not all fonts are supported in all the browsers and devices, hence we should provide comma separated fonts, so that, most device/browsers are covered. The five generic font names are: serif, sans-serif, cursive, Fantasy, Monospace

Note: Font may not render the same way in all the browsers.

If a font name contains white-space, it must be quoted. Single quotes must be used when using the "style" attribute in HTML.

- **Default** value of font-family property is **depends on the browser.**

Example

```
<head>
<style>
      .test
      {  font-family:"Comic Sans MS", cursive, sans-serif;}
      .test1
      { font-family: Arial;}
   </style>
</head>
<body>
   <p class="test">
      This text is rendered in either Comic Sans MS, cursive, or the
      default sans-
serif font depending on which font  you have at your system.
    </p>
    <p class="test1">
      This is a Paragraph.
    </p>
</body>
```

Output

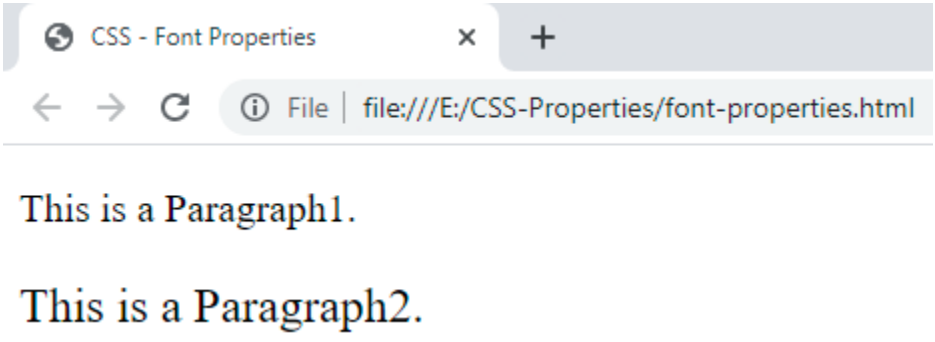
➢ Font Size Property

- It is used to set the font size of an HTML element. The font-size can be set in different ways like in "pixels, percentage, em, rem, points or we can set values like larger | smaller | xx-small | x-small | small | medium | large | x-large | xx-larger" etc.
- **Medium(16px)** is the **default** value of font-size property.
- px pixel value differs for devices with different screen resolution.
- Another widely used unit is em, If we set the font-size of the text in the body as 1em and set the font-size of the h1 heading text as 3em. Then no matter which device, the browser will make sure that the heading text is always 3 times the size of the body text.

Example

```
<style>
      .test
      {
        font-size: 20px;
      }
      .test1
      {
         font-family: 1.5em;
      }
   </style>
</head>
<body>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
</body>
```

Output



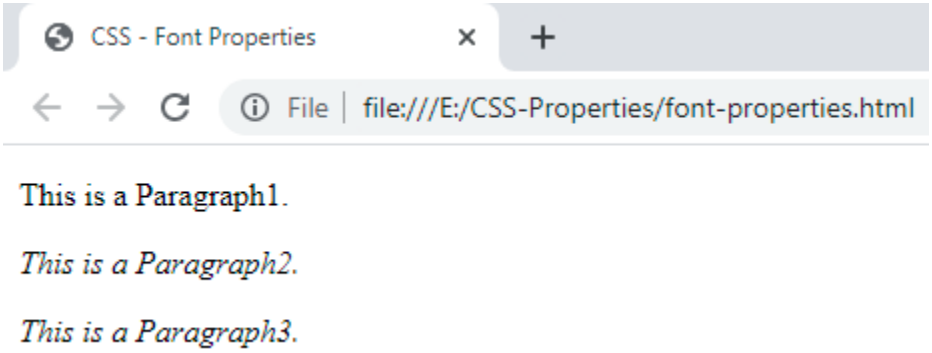
➢ Font Style Property

- CSS Font style property defines what type of font you want to display.
- **Possible values are italic, oblique, or normal.**
- **Normal** is a **default** value of font-style property.

Example

```
<head><style>
      .test{ font-style: normal;}
      .test1{ font-style: italic;}
      .test2{ font-style: oblique;}
   </style>
</head>
<body>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
   <p class="test2">This is a Paragraph3.</p>
</body>
```

Output



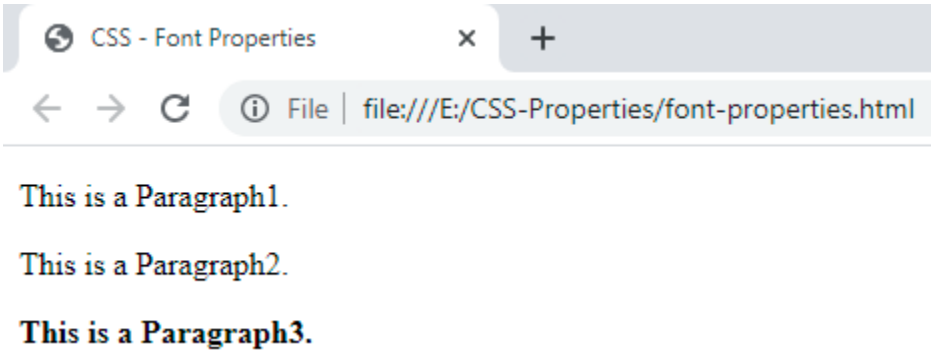
➢ Font weight Property

- font weight property defines the weight of the font and specify that how bold a font is.
- **The possible values of font weight are normal, bold, bolder, lighter or number (100, 200….. upto 900).**
- Most browsers would interpret **100-500** as normal text, **600-900** as bold. Relative values such as *lighter* or *bolder* will increase or decrease according to the parent element's font weight.
- **Normal** is a **default** value of font weight property.

Example

```
<head>
<style>
      .test{font-weight: normal;}
      .test1{ font-weight: 400;}
      .test2{font-weight: bold;}
   </style>
</head>
<body>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
   <p class="test2">This is a Paragraph3.</p>
</body>
```

Output



➢ Font variant Property

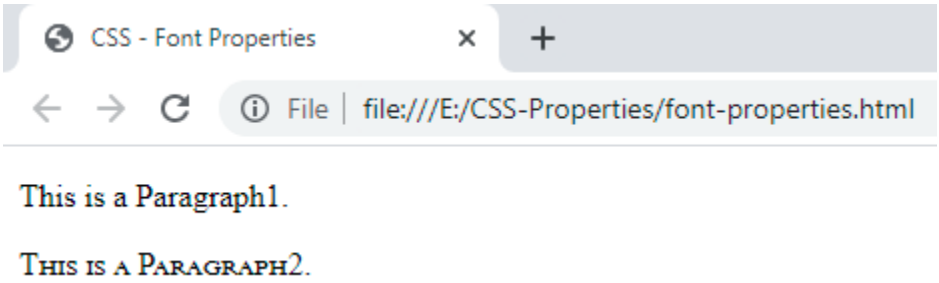
- CSS font variant property specifies how to set font variant of an element.
- **Possible values are small-caps and normal.**
- **Normal** is a **default** value of font-variant property.

- In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Example

```
<head>
<style>
      .test{font-variant: normal;}
      .test1{ font-variant: small-caps;}
   </style>
</head>
<body>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
</body>
```

Output



➢ Font Property

- To shorten the code, it is also possible to specify all the individual font properties in one property.
- The font property is a shorthand property for:
    - font-style
    - font-variant
    - font-weight
    - font-size/line-height
    - font-family

NOTE: The font-size and font-family values are required. If one of the other values is missing, their default value are used.

Example

```
<head>
   <style>
      .test{font: 20px Arial, sans-serif;}
      .test1{font: italic bold 12px/30px Georgia, serif;}
   </style>
</head>
<body>
      <h1>The font Property</h1>
      <p class="test">This is a Paragraph1.</p>
      <p class="test1">This is a Paragraph2.</p>
</body>
```

Output

In the above example,
First Paragraph, the font size is set to 20 pixels, and the font family is Arial.
Second Paragraph, the font is set to italic and bold, the font size is set to 12 pixels, the line height is set to 30 pixels, and the font family is Georgia.

CSS Border Properties

- The CSS border properties are used to specify the style, color and size of the border of an element.
- CSS Border property include following properties:
  - **border-style:** This property specifies whether a border should be solid, dashed line, double line, or one of the other possible values.
  - **border-width:** This property specifies the width of a border.
  - **border-color:** This property specifies the color of a border.

- **border**: This property is used as shorthand to specify a number of other font properties.
- **border-radius:** This property sets the rounded borders and provides the rounded corners around an element, tags, or div.
- **border-collapse:** This property is used to set the border of the table cells.
- **border-spacing:** This property is used to set the distance between the borders of the adjacent cells in the table.

➢ Border style Property

- The Border style property is used to specify the border type which you want to display on the web page.
- There are some border style values which are used with border-style property to define a border.

Value	Description
none	It doesn't define any border.
dotted	It is used to define a dotted border.
dashed	It is used to define a dashed border.
solid	It is used to define a solid border.
double	It defines two borders wIth the same border-width value.
groove	It defines a 3d grooved border. effect is generated according to border-color value.
ridge	It defines a 3d ridged border. effect is generated according to border-color value.
inset	It defines a 3d inset border. effect is generated according to border-color value.
outset	It defines a 3d outset border. effect is generated according to border-color value.

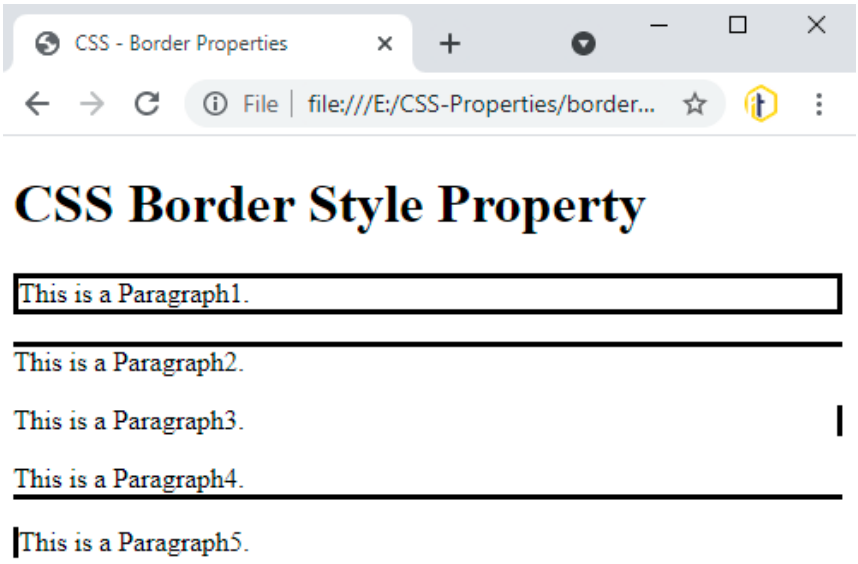
- You can individually change the style of the bottom, left, top, and right borders of an element using the following properties –
    - **border-bottom-style** changes the style of bottom border.
    - **border-top-style** changes the style of top border.
    - **border-left-style** changes the style of left border.

- **border-right-style** changes the style of right border.

Example

```
<head>
<style>
      .test{border-style: solid;}
      .test1{border-top-style: solid;}
      .test2{border-right-style: solid;}
      .test3{border-bottom-style: solid;}
      .test4{border-left-style: solid;}
   </style>
</head>
<body>
   <h1>CSS Border Style Property</h1>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
   <p class="test2">This is a Paragraph3.</p>
   <p class="test3">This is a Paragraph4.</p>
   <p class="test4">This is a Paragraph5.</p>
</body>
```

Output



➢ Border width Property

- The border-width property is used to set the border's width. It is set in px, pt, cm or em. You can also use the one of the three pre-defined values, thin, medium or thick to set the width of the border.

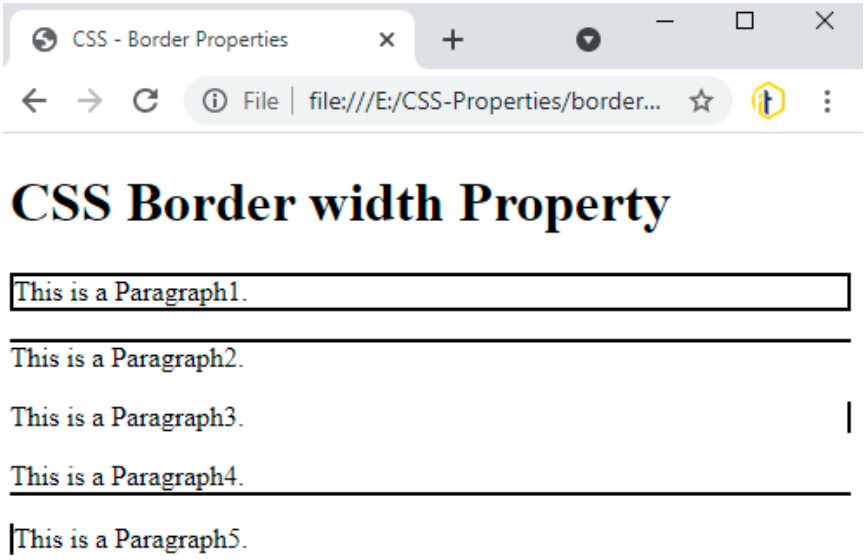
- You can individually change the width of the bottom, top, left, and right borders of an element using the following properties –
    - **border-bottom-width** changes the width of bottom border.
    - **border-top-width** changes the width of top border.
    - **border-left-width** changes the width of left border.
    - **border-right-width** changes the width of right border.

NOTE: The border-width property is not used alone. It is always used with other border properties like "border-style" property to set the border first otherwise it will not work. If you don't use border-style then border-width will be medium.

Example

```
<head>
<style>
      .test{border-style: solid; border-width: 2px;}
      .test1{border-top-style: solid; border-top-width: 2px;}
      .test2{border-right-style: solid; border-right-width: 2px;}
      .test3{border-bottom-style: solid; border-bottom-width: 2px;}
      .test4{border-left-style: solid; border-left-width: 2px;}
</style>
</head>
<body>
   <h1>CSS Border width Property</h1>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
   <p class="test2">This is a Paragraph3.</p>
   <p class="test3">This is a Paragraph4.</p>
   <p class="test4">This is a Paragraph5.</p>
</body>
```

Output



➢ Border Color Property

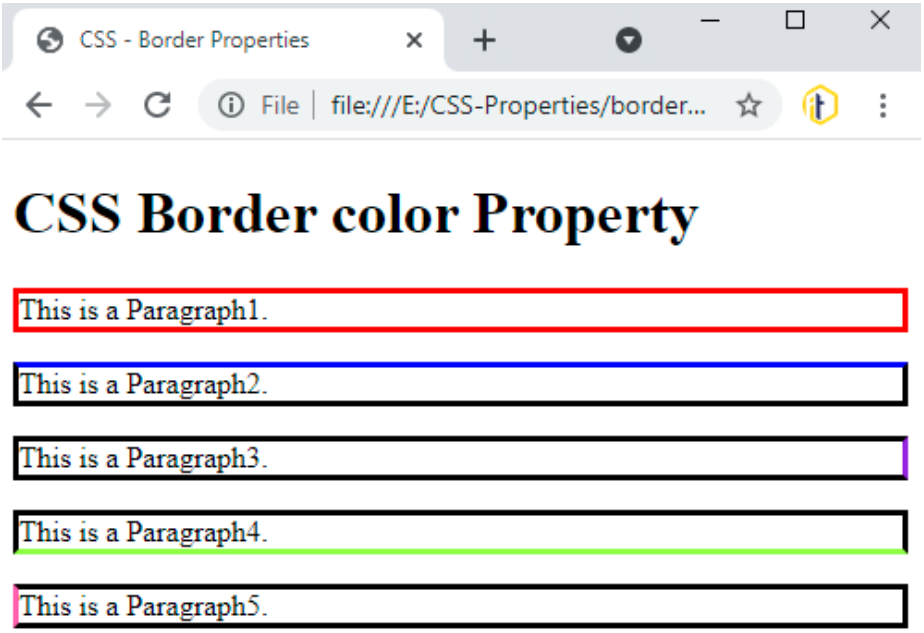
- The border-color property is used to set the color of the four borders.
- The color can be set by:
  1. Name: It specifies the color name. For example: "red".
  2. RGB: It specifies the RGB value of the color. For example: "rgb(255,0,0)".
  3. Hex: It specifies the hex value of the color. For example: "#ff0000".
- **There is also a border color named "transparent". If the border color is not set it is inherited from the color property of the element.**
- You can individually change the color of the bottom, left, top and right sides of an element's border using the properties –
  - **border-bottom-color** changes the color of bottom border.
  - **border-top-color** changes the color of top border.
  - **border-left-color** changes the color of left border.
  - **border-right-color** changes the color of right border.

NOTE: The border-color property is not used alone. It is always used with other border properties like "border-style" property to set the border first otherwise it will not work. If you don't use border-color property then border-color will be font color of the element.

Example

```
<head>
<style>
      .test{border-style: solid; border-color: red;}
      .test1{border-style: solid; border-top-color: blue;}
      .test2{border-style: solid; border-right-color:blueviolet;}
      .test3{border-style: solid; border-bottom-color: greenyellow;}
      .test4{border-style: solid; border-left-color: hotpink;}
   </style>
</head>
<body>
   <h1>CSS Border color Property</h1>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
   <p class="test2">This is a Paragraph3.</p>
   <p class="test3">This is a Paragraph4.</p>
   <p class="test4">This is a Paragraph5.</p>
</body>
```

Output



➢ CSS Border Property

- To shorten the code, it is also possible to specify all the individual border properties in one property.
- The border property is a shorthand property for the following individual border properties:
    - border-width
    - border-style **(required)**
    - border-color
- The following example shows how to use all the three properties into a single property. This is the most frequently used property to set border around any element.

Syntax

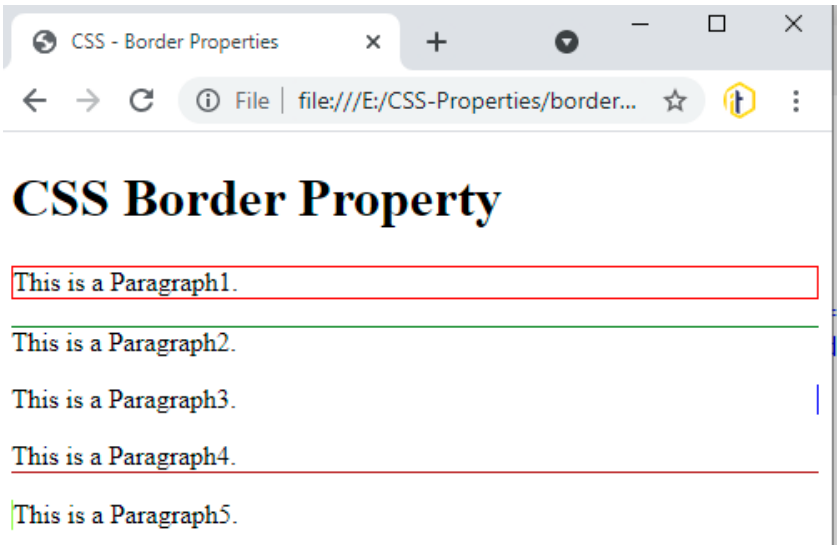
```
border: border-width border-style border-color;
```

- Same as above, you can set border shorthand individually,
    - **border-top**
    - **border-right**
    - **border-left**
    - **border-left**

Example

```
<head>
<style>
      .test{border:1px solid red;}
      .test1{border-top:1px solid green}
      .test2{border-right: 1px solid blue;}
      .test3{border-bottom: 1px solid brown;}
      .test4{border-left: 1px solid greenyellow;}
   </style>
</head>
<body>
   <h1>CSS Border Property</h1>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
   <p class="test2">This is a Paragraph3.</p>
   <p class="test3">This is a Paragraph4.</p>
   <p class="test4">This is a Paragraph5.</p>
</body>
```

Output



➢ Border Radius Property

- This CSS property sets the rounded borders and provides the rounded corners around an element, tags, or div. It defines the radius of the corners of an element.
- It is shorthand for **border top-left-radius, border-top-right-radius, border-bottom-right-radius** and **border-bottom-left-radius**. It gives the rounded shape to the corners of the border of an element. We can specify the border for all four corners of the box in a single declaration using the border-radius. The values of this property can be defined in percentage or length units.
- The border property is a shorthand property for the following individual border properties:

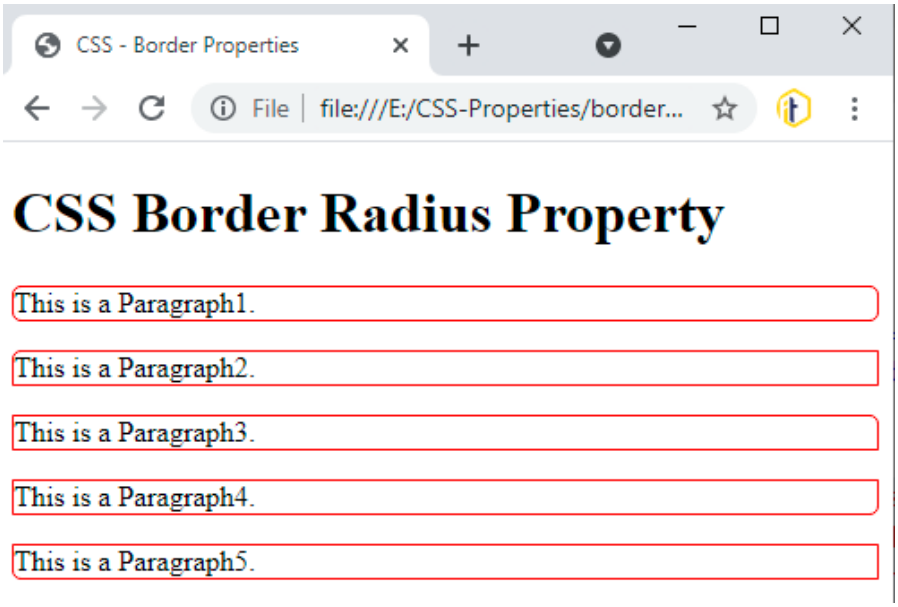
    - **border-top-left-radius** is used to set the border-radius for the top-left corner
    - **border-top-right-radius** is used to set the border-radius for the top-right corner
    - **border-bottom-right-radius i**s used to set the border-radius for the bottom-right corner
    - **border-bottom-left-radius** is used to set the border-radius for the bottom-left corner

Example

```
<head>

<style>
      .test{border:1px solid red; border-radius: 5px;}
      .test1{border:1px solid red; border-top-left-radius: 5px;}
      .test2{border:1px solid red; border-top-right-radius: 5px;}
      .test3{border:1px solid red; border-bottom-right-radius: 5px;}
      .test4{border:1px solid red; border-bottom-left-radius: 5px;}
   </style>
</head><body>
   <h1>CSS Border Radius Property</h1>
   <p class="test">This is a Paragraph1.</p>
   <p class="test1">This is a Paragraph2.</p>
   <p class="test2">This is a Paragraph3.</p>
   <p class="test3">This is a Paragraph4.</p>
   <p class="test4">This is a Paragraph5.</p>
</body>
```

Output



If the border-radius property has four values:

- **border-radius: 3px 4px 5px 6px;**
    - left top border radius is 3px
    - right top border radius is 4px
    - right bottom border radius is 5px

- left bottom border radius is 6px

If the border-radius property has three values:

- **border-radius: 2px 3px 4px;**
  - left top border radius is 2px
  - right top and left bottom border radiuses are 3px
  - right bottom border radius is 4px

If the border-radius property has two values:

- **border-radius: 2px 3px;**
  - left top and right bottom border radiuses are 2px
  - right top and left bottom border radiuses are 3px

If the border-radius property has one value:

- **border-style: 2px;**
  - all four border radiuses are 2px

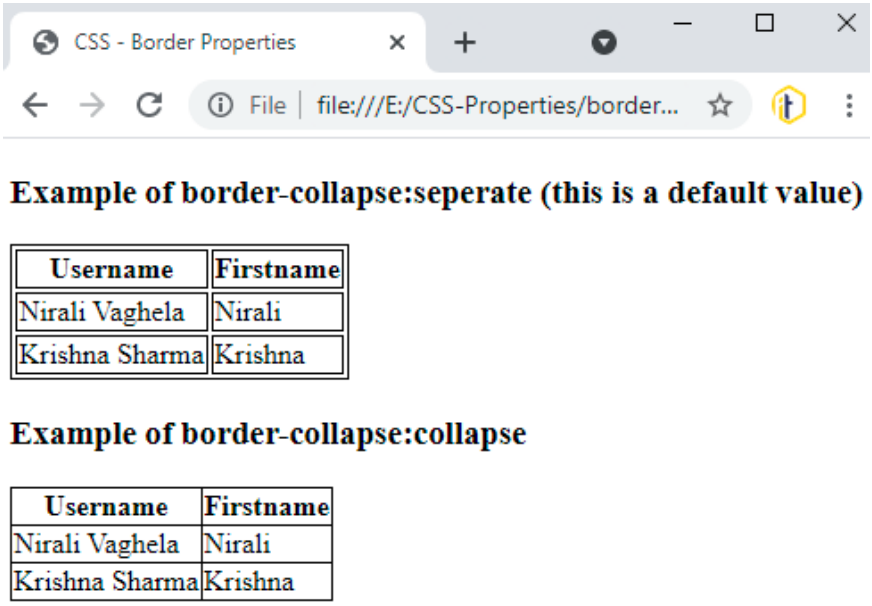
➢ Border Collapse Property

- This CSS property is used to set the border of the table cells and specifies whether the table cells share the separate or common border.
- This property has two main values that are **separate** and **collapse**.
- When it is set to the value **separate**, the distance between the cells can be defined using the **border-spacing** property. When the **border-collapse** is set to the value **collapse**, Borders are collapsed into a single border (border-spacing and empty-cells properties have no effect)
- **Separate** is a **default** value of border-collapse property.

Example

```
<style>
      .tbl,td,th,.tbl1
      {
         border: 1px solid black;
      }
      .tbl1
      {
         border-collapse: collapse;
      }
   </style>
</head>
<body>
   <h3>Example of border-
collapse:seperate (this is a default value)</h3>
   <table class="tbl">
      <tr>
         <th>Username</th>
         <th>Firstname</th>
      </tr>
      <tr>
         <td>Nirali Vaghela</td>
         <td>Nirali</td>
      </tr>
      <tr>
         <td>Krishna Sharma</td>
         <td>Krishna</td>
      </tr>
   </table>
   <h3>Example of border-collapse:collapse</h3>
   <table class="tbl1">
      <tr>
         <th>Username</th>
         <th>Firstname</th>
      </tr>
      <tr>
         <td>Nirali Vaghela</td>
         <td>Nirali</td>
      </tr>
      <tr>
         <td>Krishna Sharma</td>
         <td>Krishna</td>
      </tr>
   </table>
</body>
```

Output



➢ Border Spacing Property

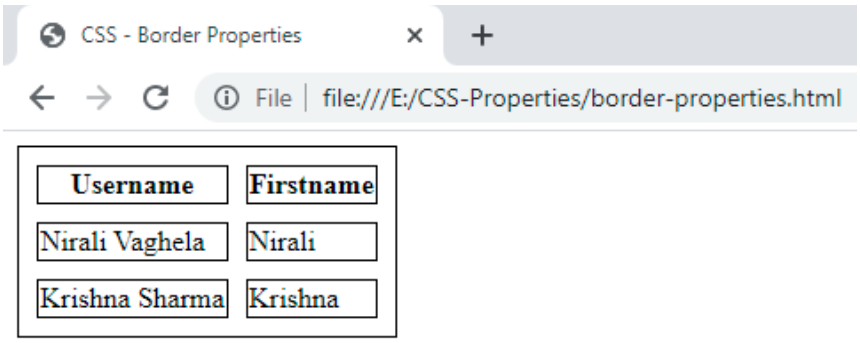
- This CSS property is used to set the distance between the borders of the adjacent cells in the table. It applies only when the **border-collapse** property is set to **separate**. There will not be any space between the borders if the **border-collapse** is set to **collapse**.
- It can be defined as one or two values for determining the vertical and horizontal spacing.
- When only one value is specified, then it sets both horizontal and vertical spacing.
- When we use the two-value syntax, then the first one is used to set the horizontal spacing (i.e., the space between the adjacent columns), and the second value sets the vertical spacing (i.e., the space between the adjacent rows).
- You can define value in px, pt, em, rem.

Example

- Example with one value

```
<head>
<style>
      .tbl,td,th{ border: 1px solid black;}
      .tbl{border-spacing: 10px;}
</style>
</head>
<body>
   <table class="tbl">
      <tr>
         <th>Username</th>
         <th>Firstname</th>
      </tr>
      <tr>
         <td>Nirali Vaghela</td>
         <td>Nirali</td>
      </tr>
      <tr>
         <td>Krishna Sharma</td>
         <td>Krishna</td>
      </tr>
   </table>
</body>
```

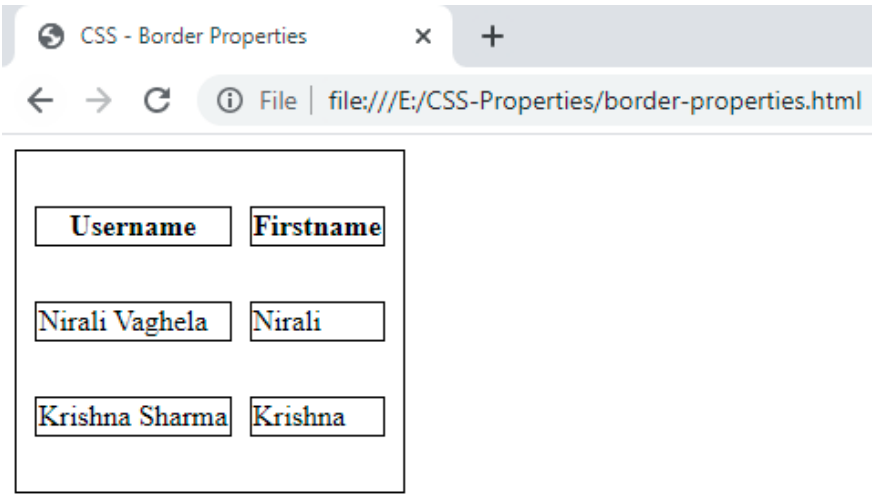
Output



- Example with two values

```
<head>
<style>
      .tbl,td,th{border: 1px solid black;}
      .tbl{border-spacing: 10px 30px;}
   </style>
</head>
<body>
   <table class="tbl">
      <tr>
         <th>Username</th>
         <th>Firstname</th>
      </tr>
      <tr>
         <td>Nirali Vaghela</td>
         <td>Nirali</td>
      </tr>
      <tr>
         <td>Krishna Sharma</td>
         <td>Krishna</td>
      </tr>
   </table>
</body>
```

Output



CSS Background Properties

- The CSS background properties are used to define the background effects for elements.
- CSS Border property include following properties:
  - **background-color:** This property is used to specify the background color of the element.
  - **background-image:** This property is used to set the background image of an element.
  - **background-repeat:** This property is used to control the repetition of an image in the background.
  - **background-position:** This property is used to define the initial position of the background image.
  - **background-attachment:** This property is used to control the scrolling of an image in the background.
  - **background-size:** This property is used to specifies the size of the background images.

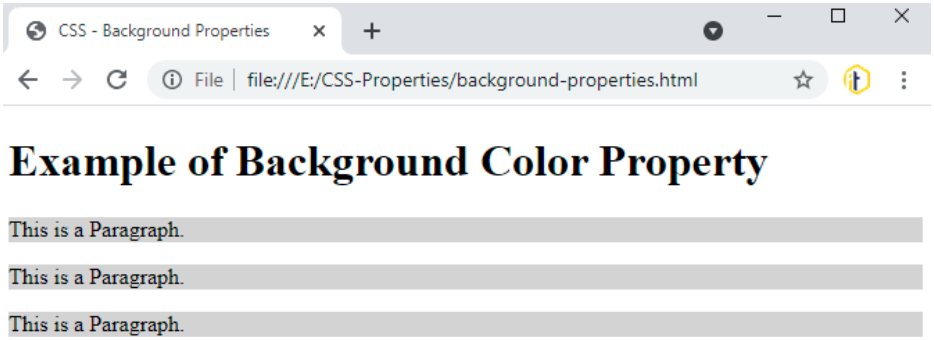
Background Color Property

- The background-color property sets the background color of an element.
- The background of an element is the total size of the element, including padding and border (but not the margin).
- **Transparent** is a **default** value of background-color property.
- The color can be set by:
  1. **Name:** It specifies the color name. For example: "red".
  2. **RGB:** It specifies the RGB value of the color. For example: "rgb(255,0,0)".
  3. **Hex:** It specifies the hex value of the color. For example: "#ff0000". (There are many websites to check hexa codes for colors, like ColorHexa)

Example

```
<head><style>
      .test{background-color: lightgray;}
      .test1{background-color: rgb(211, 211, 211);}
      .test2 {background-color: #d3d3d3;}
</style></head><body>
   <h1>Example of Background Color Property</h1>
   <p class="test">This is a Paragraph.</p>
   <p class="test1">This is a Paragraph.</p>
   <p class="test2">This is a Paragraph.</p></body>
```

Output



➢ Background Image Property

- The background-image property is used to set an image as a background of an element.
- By default the image covers the entire element, including padding and border (but not the margin).
- By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.
- The background image requires a URL to the source image specified with the url() syntax.
- None is the default value of background-image property

Example

```
<head>
<style>
      body{ background-image: url(sample.jpeg);}
</style>
</head>
<body>
   <h1>Example of Background Image Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output

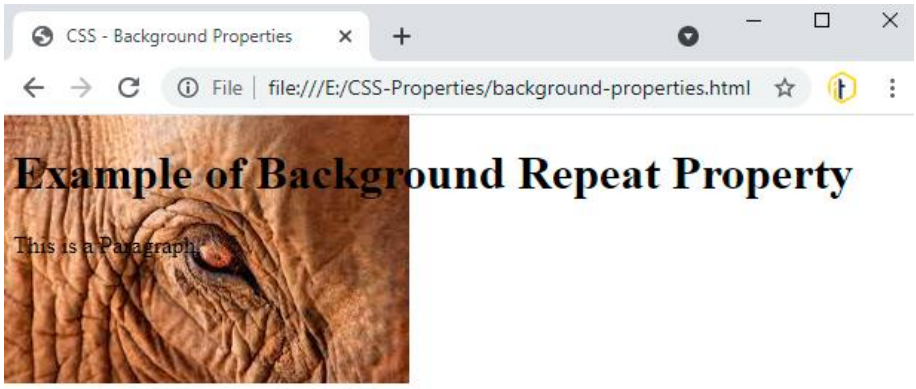
➢ Background Repeat Property

- The background-repeat property sets if/how a background image will be repeated.
- By default, a background-image is repeated both vertically and horizontally.
- Sometimes, it may so happen that the background image that you choose for your webpage may be smaller than the size of the element. In such a situation, you may want to repeat the image.
- The values for this property can be following:
  - **repeat** - repeats image both vertically and horizontally. This is the **default value** for background-repeat property.
  - **repeat-x** - repeats only horizontally
  - **repeat-y** - repeats vertically
  - **no-repeat** - doesn't repeat the background image.

Example

```
<head><style>
      body{
         background-image: url(sample.jpg);
         background-repeat: no-repeat;}
   </style>
</head>
<body>
   <h1>Example of Background Repeat Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output



The above code will make the image **sample.jpeg** as the webpage background, but the size of the image is smaller, it is not completely fill the webpage, and as the background-repeat property is set to no-repeat, hence it won't be repeated, leaving some area with no background.

➢ Background Position Property

- The background-position property is used to define the initial position of the background image.
- **Top left** is the **default value** of background-position property.
- You can set the following positions:
    - **center**
    - **top**
    - **bottom**
    - **left**
    - **Right**

- The position of the background image from the top left corner can be specified as a pixels value, percentage value or it can also be named values (top, right, bottom, left, center)

Example

- example of background-position property with name value

```
<style>
      body{
         background-image: url(sample.jpg);
         background-repeat: no-repeat;
         background-position: right top;}
</style>
</head>
<body>
   <h1>Example of Background Position Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

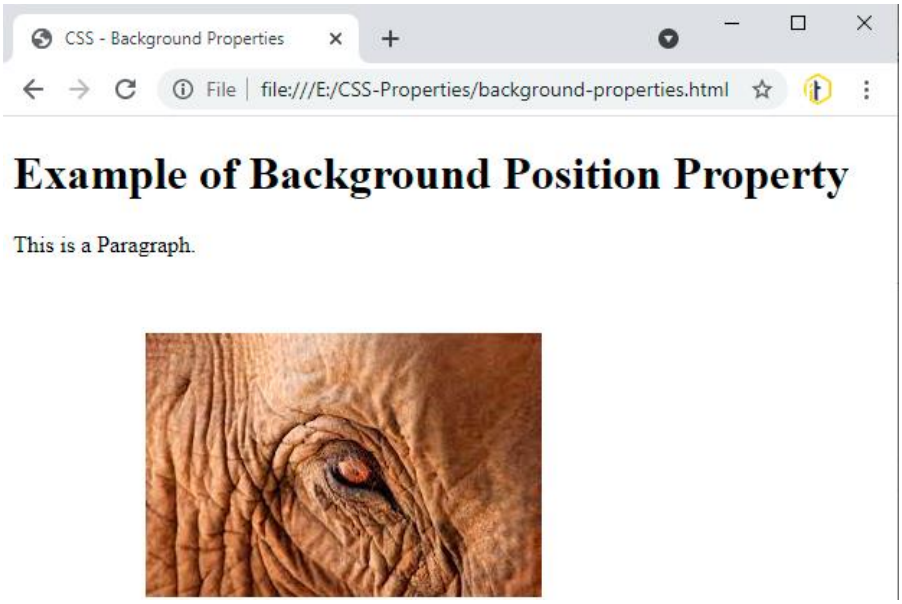
Output

Example

- example of background-position property with pixels value

```
<style>
      body{
         background-image: url(sample.jpg);
         background-repeat: no-repeat;
         background-position: right top;}
</style>
</head>
<body>
   <h1>Example of Background Position Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output



Example

- example of background-position property with percentage value

```
<style>
      body{
         background-image: url(sample.jpg);
         background-repeat: no-repeat;
         background-position: 10% 20%;}
</style>
</head>
<body>
   <h1>Example of Background Position Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

When we specify background-position in **pixels** or **percentage**, we specify 2 values, first value specifies distance from the left side, and the second value specifies the distance from the top.

➢ Background Size Property

- The background-size property specifies the size of the background images.
- You can set the following sizes:
    - **auto**
    - **contain**

- **center**

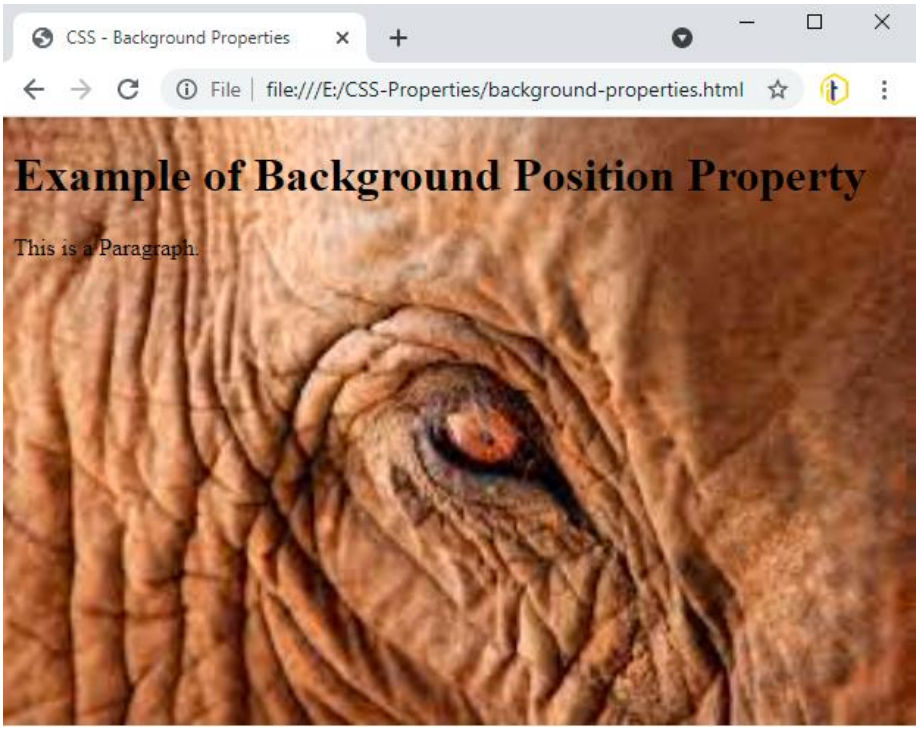
- The size of the background image can be specified as a one-value syntax (sets the width of the image (height becomes "auto"), the two-value syntax (first value: width of the image, second value: height), or it can also be named values (auto, contain, cover)

Example

- example of background-size with name value

```
<style>
      body{
         background-image: url(sample.jpg);
         background-repeat: no-repeat;
         background-size:cover;}
   </style>
</head>
<body>
   <h1>Example of Background Position Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output

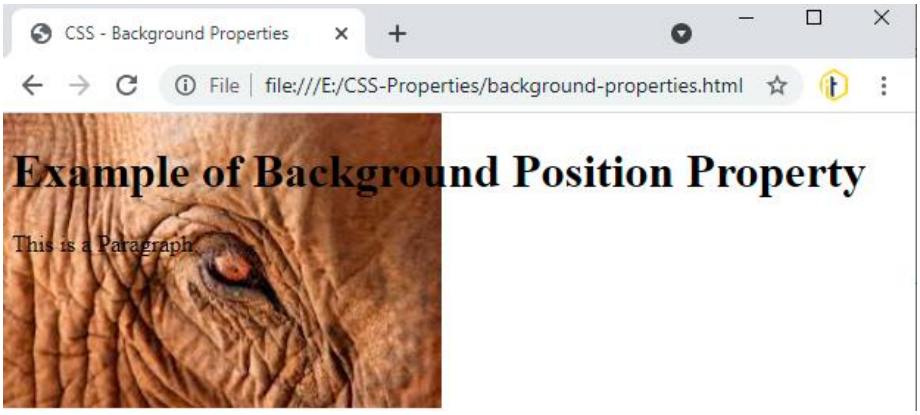


Example

- example of background-size with one value

```
<style>
      body{
         background-image: url(sample.jpg);
         background-repeat: no-repeat;
         background-size:300px;}
   </style>
</head>
<body>
   <h1>Example of Background Position Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output



In the above example width of the image is 300px and height is auto.

Example

- example of background-size with two value

```
<style>
      body{
         background-image: url(sample.jpg);
         background-repeat: no-repeat;
         background-size:100px 200px;}
   </style>
</head>
<body>
   <h1>Example of Background Position Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output

In the above example width of the image is 100px and height is 200px.

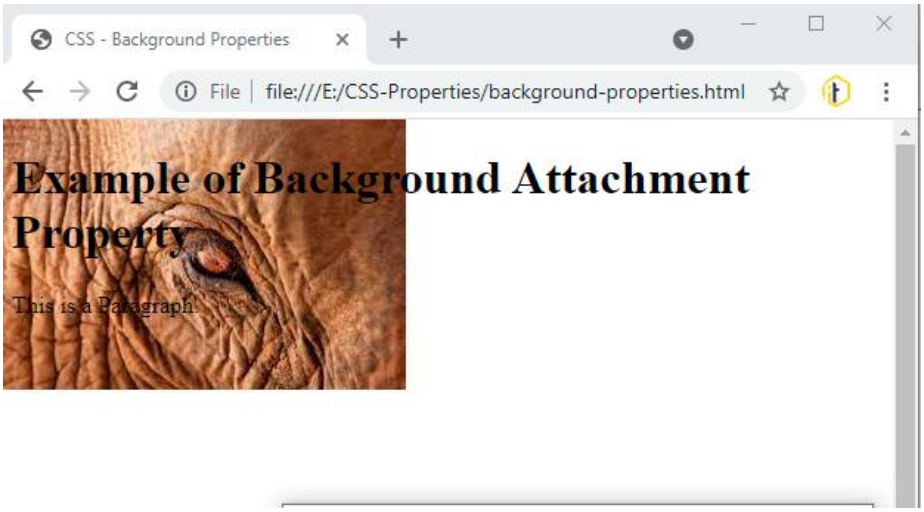
➢ Background Attachment Property

- This background-attachment property sets, whether the background image will scroll or not.
- **Scroll** is the **default** value of background-attachment property.
- **Possible values are scroll and fixed.**
- Value scroll, which sets the background to scroll with the associated content, typically text. The alternate value, fixed, makes the background static while associated content scrolls over the background.

Example

```
<style>
      body{
         background-image: url(sample.jpg);
         background-repeat: no-repeat;
         background-attachment: fixed;}
   </style>
</head>
<body style="height: 1200px;">
   <h1>Example of Background Attachment Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output



In above example, when you scroll up and down image will remain on the same place.

- **Sample of parallax scrolling effect**

Example

```
<style>
        .img_bg {
             background-image: url("Koala.jpg");
             min-height: 500px;
             background-attachment: fixed;
             background-position: center;
             background-repeat: no-repeat;
             background-size: cover;}
    </style>
</head>
<body style="height: 1200px;">
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ad
doloribus eaque, harum quo ipsa numquam accusantium molestiae
 quos quibusdam consequatur atque sapiente consequuntur id. Ob
caecati vel dolores eius rem delectus?</p>
    <div class="img_bg"></div>
    <div style="background-
color:yellow;">Lorem ipsum dolor sit amet consectetur adipisicing
elit. Ab exercitationem dignissimos numquam, nihil earum voluptas
 voluptatibus recusandae vitae impedit veniam, totam voluptatum,
 ex vero quis deleniti cumque unde sit dolorem.</div>
</body>
```

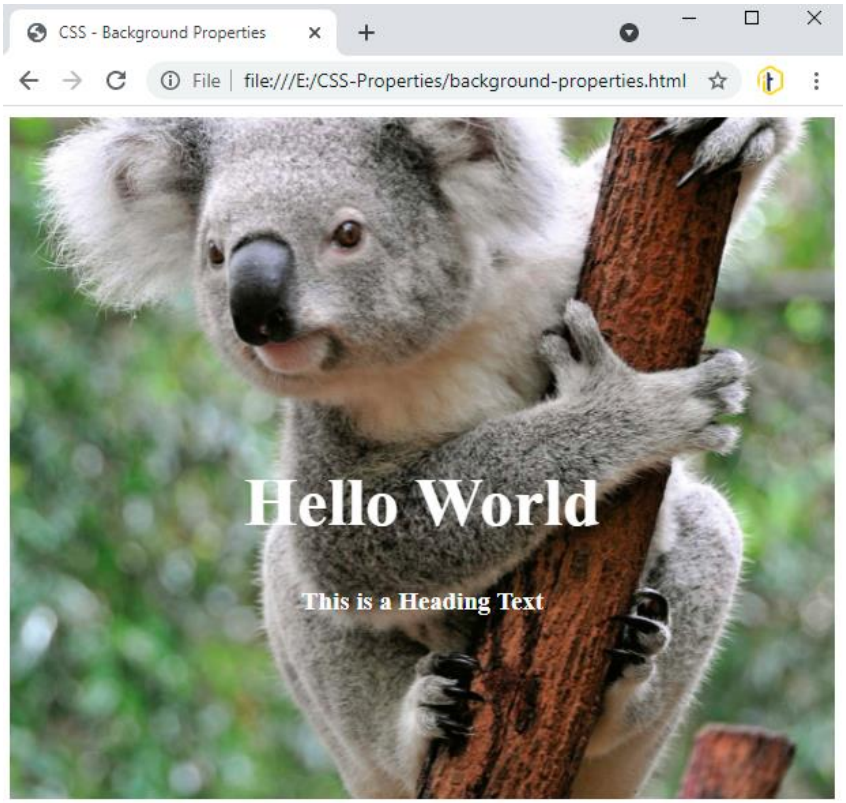
Output

- **Sample of banner (hero) image using different background images**

Example

```
<style>.banner-image {
        background-image: url(Koala.jpg);
        background-color: #cccccc;
        height: 500px;background-position: center;
        background-repeat: no-repeat;
        background-size: cover;}
  .banner-text {
        text-align: center;
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        color: white;}
   </style></head>
<body>
   <div class="banner-image">
      <div class="banner-text">
        <h1 style="font-size:50px">Hello World</h1>
        <h3>This is a Heading Text</h3></div></div></body>
```

Output



➢ Background Property

- To shorten the code, it is also possible to specify all the individual background properties in one property.
- The background property is a shorthand property for the following individual background properties:
    - background-color
    - background-image
    - background-position
    - background-size
    - background-repeat
    - background-origin
    - background-clip
    - background-attachment
- The order of the properties should not matter and any property that you don't define uses their own default values.

Syntax

```
background: bg-color bg-image position/bg-size bg-repeat bg-origin bg-clip bg-attachment;
```

Note: If one of the properties in the shorthand declaration is the bg-size property, you must use a / (slash) to separate it from the bg-position property, **e.g. background:url(Koala.jpg) 10px 20px/50px 50px no-repeat;** will result in a background image, positioned 10 pixels from the left, 20 pixels from the top, and the size of the image will be 50 pixels wide and 50 pixels high.

CSS Height Property

- This CSS property sets the height of an element.
- The height of an element does not include padding, borders, or margins!
- Height accepts numeric value (like pixels, (r)em, percentages).
- If we set **the height to a numeric value** (like in px, %, etc.), the content can be overflow if it does not fit in the given height. We can manage the overflowing content by defining the **overflow** property.
- If **height: auto;** the element will automatically adjust its height to allow its content to be displayed correctly.
- **Possible values are auto, length.**
- **Auto** is the **default** value of height property.

Note: The **min-height** and **max-height** properties override the height property.

Example

- Example with numeric value (make sure we have to manage overflowed contents with overflow property)

```
<head>
<style>
      .test{
         height: 100px;
         border: 1px solid black;
      }
   </style>
</head>
<body>
   <h1>Example of CSS Height Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output

Example

- Example with auto (according to content height of element will adjust)

```
<head>
<style>
      .test{
         height: auto;
         border: 1px solid black;}
</style></head>
<body>
   <h1>Example of CSS Height Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output

CSS Minimum Height Property

- The min-height property set the minimum-height of the element's content box. It means that the height of the content box can be greater than the **min-height** value, but cannot be shorter. It sets the lower bound on the element's height.
- It will be applied when the content is smaller than the minimum height; otherwise, if the content is larger, this property has no effect. This property ensures that the value of height property cannot be less than the value of the **min-height** property. It does not allow negative values.
- **Possible values are none, length(px, %, em, rem)**
- **None** is the **default value** of min-height property.

Example

```
<head><style>
      .test{
         min-height: 100px;
         border: 1px solid black;}
   </style>
</head><body>
   <h1>Example of CSS min-height Property</h1>
   <p class="test">Lorem ipsum dolor sit amet consectetur adipisicing elit. In ventore a nemo soluta incidunt neque consequatur eum sit saepe vitae reicien dis consequuntur aliquam corrupti, fuga qui sunt.</p></body>
```

Output

CSS Maximum Height Property

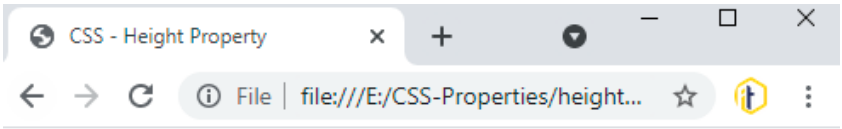
- The max-height property sets the maximum height of the element's content box. It means that the height of the content box can be smaller than the max-height value, but cannot be greater. It sets the upper bound on the element's height.
- When the content is larger than the maximum height, it will overflow. If the content is smaller than the **max-height**, this property does not affect. This property ensures that the value of height property cannot be greater than the value of the **max-height** property. It does not allow negative values.
- Sometimes it is useful to limit the element's height to a certain range.
- Possible values are none, length(px, %, em, rem)
- None is the default value of max-height Property.

Example

- Example, where the content is larger than the maximum height

```
<head><style>
      .test{
         max-height: 100px;
         border: 1px solid black;}
   </style></head>
<body>
   <h1>Example of CSS max-height Property</h1>
   <p class="test">Lorem ipsum dolor sit amet consectetur adipisicing elit. I
nventore a nemo soluta incidunt neque consequatur eum sit saepe vitae reici
endis consequuntur aliquam corrupti, fuga qui sunt.Lorem ipsum dolor sit am
et consectetur adipisicing elit. Inventore a nemo soluta incidunt neque conse
quatur eum sit saepe vitae reiciendis consequuntur aliquam corrupti, fuga qu
i sunt.</p></body>
```

Output



Example of CSS max-height Property

Lorem ipsum dolor sit amet consectetur adipisicing elit. Inventore a nemo soluta incidunt neque consequatur eum sit saepe vitae reiciendis consequuntur aliquam corrupti, fuga qui sunt. Lorem ipsum dolor sit amet consectetur adipisicing elit. Inventore a nemo soluta incidunt neque consequatur eum sit saepe vitae reiciendis consequuntur aliquam corrupti, fuga qui sunt.

Example

- Example, where the content is smaller than the maximum height then height will be your content's height.

```
<head><style>
      .test{
         max-height: 100px;
         border: 1px solid black;}
   </style></head><body>
   <h1>Example of CSS max-height Property</h1>
   <p class="test">Lorem ipsum dolor sit amet consectetur adipisicing elit. I
nventore a nemo soluta incidunt neque consequatur eum sit saepe vitae reici
endis consequuntur aliquam corrupti, fuga qui sunt.Lorem ipsum dolor sit am
et consectetur adipisicing elit. Inventore a nemo soluta incidunt neque conse
quatur eum sit saepe vitae reiciendis consequuntur aliquam corrupti, fuga qu
i sunt.</p></body>
```

Output

CSS Width Property

- The CSS width property is used to set the width of the content area of an element.
- It does not include padding borders or margins. It sets width of the area inside the padding, border, and margin of the element.
- **Possible values are auto, length(px, %, em, rem)**
- Normally give width in percentage so we open our webpage in any device and any resolution our html element will set properly and never overflow content in width.

- **Auto** is the **default** value of width property.

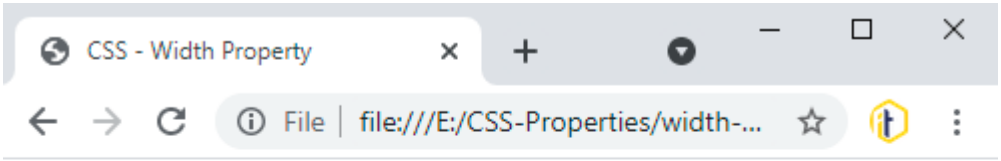
Note: The **min-width** and **max-width** properties override the width property.

Example

- Example with numeric value

```
<head>
<style>
      .test{
         width: 30%;
         border: 1px solid black;}
   </style>
</head>
<body>
   <h1>Example of Width Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output



Example of Width Property

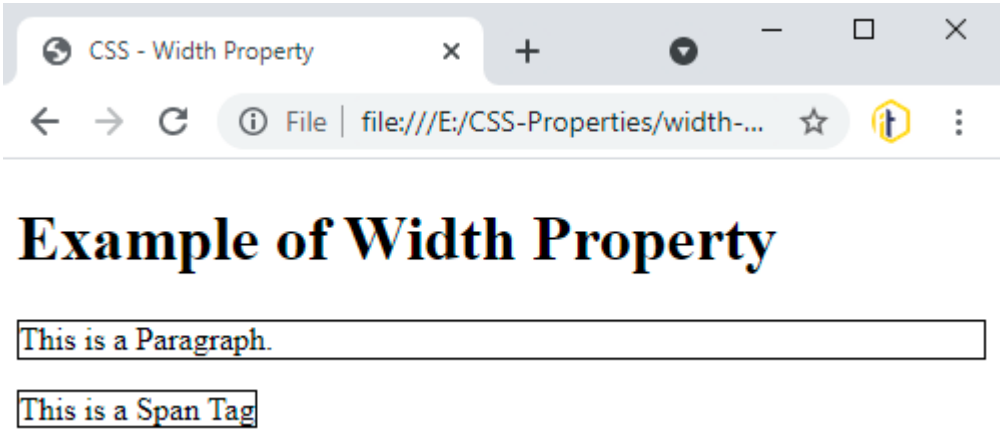
This is a Paragraph.

Example

- Example with auto value. (if Block element has auto value then width will be 100% and if inline element has auto value then width will be according to the content)

```
<head><style>
      .test,.test1{
         width: auto;
         border: 1px solid black;}</style></head>
<body>
   <h1>Example of Width Property</h1>
   <p class="test">This is a Paragraph.</p>
   <span class="test1">This is a Span Tag</span></body>
```

Output



CSS Minimum Width Property

- This min-width is used to set the minimum width of the element's content box. It means that the width of the content box can be greater than the **min-width** value, but cannot be shorter. It sets the lower bound on the element's width.
- It will be applied when the content is smaller than the minimum width; otherwise, if the content is larger, this property has no effect. This property ensures that the value of CSS **width** property cannot be less than the value of **min-width** property. It does not allow negative values.
- **Possible values are none, length(px,%, em, rem)**
- **None** is the **default** value of min-width property.

Example

```
<head>
<style>
      .test{
        min-width:20%;
         border: 1px solid black;
         display: inline-block;}
   </style></head>
<body>
   <h1>Example of Minimum Width Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output

CSS Maximum Width Property

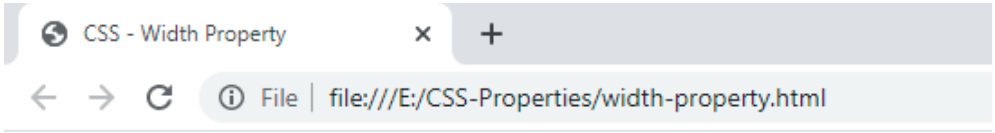
- The **max-width** property in CSS is used to set the maximum width of the element's content box. It means that the width of the content box can be smaller than the **max-width** value, but cannot be greater. It sets the upper bound on the element's width.
- When the content is larger than the maximum width, then it will automatically change the height of the element. If the content is smaller than the **max-width**, this property has no effect. This property ensures that the value of width property cannot be greater than the value of **max-width** property. It does not allow negative values.
- **Possible values are none, length(px,%, em, rem)**
- **None** is the **default** value of min-width property.

Example

- Example with length, when the content is smaller than the max-width.

```
<head>
<style>
      .test{
         max-width:20%;
         border: 1px solid black;
         display: inline-block;}
   </style>
</head>
<body>
   <h1>Example of Maximum Width Property</h1>
   <p class="test">This is a Paragraph.</p>
</body>
```

Output



Example of Maximum Width Property

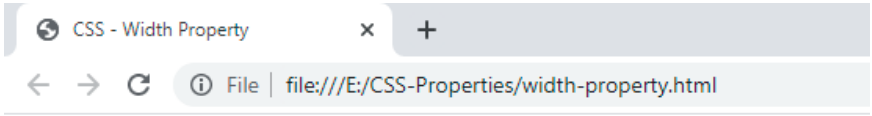
This is a Paragraph.

Example

- Example with length, when the content is greater than the max-width.

```
<head>
<style>
      .test{
         max-width:20%;
         border: 1px solid black;
         display: inline-block;}
   </style>
</head>
<body>
   <h1>Example of Maximum Width Property</h1>
   <p class="test">Lorem ipsum dolor sit amet consectetur, adipisicing elit. F
ugiat consequuntur sunt excepturi sequi possimus maiores nam, error, exerci
tationem amet nostrum explicabo quasi aliquid inventore molestiae dignissim
os, necessitatibus unde incidunt? Unde.</p>
</body>
```

Output



Example of Maximum Width Property

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Fugiat consequuntur sunt excepturi sequi possimus maiores nam, error, exercitationem amet nostrum explicabo quasi aliquid inventore molestiae dignissimos, necessitatibus unde incidunt? Unde.

CSS Overflow Property

- There are times, when the content inside an element is more, which leads to content overflowing outside the element. CSS overflow property allows us to handle the overflowing content either by hiding it or by adding a scroll bar to the block-level element.
- We know that every single element on a page is a rectangular box and the size, positioning and behavior of these boxes are controlled via CSS.
- Let's take an example: If you don't set the height of the box, it will grow as large as the content. But if you set a specific height or width of the box and the content inside cannot fit then what will happen. The CSS overflow property is used to overcome this problem.
- **Possible values are visible, hidden, scroll, auto.**
- **visible** is a **default value** of **overflow property.**
- **visible:** This is the default value. In this case, no action is taken on the overflowing content and it is visible outside the element as well.
- **hidden:** In this case, the overflowing content, which is outside the element becomes invisible i.e. it is hidden.
- **scroll:** In this case, The overflow is clipped, and a scrollbar is added to see the rest of the content
- **auto:** If the content is less, this acts like visible but as the content starts to overflow outside, this property starts acting like scroll and adds a scroll bar to the element. In short, Similar to scroll but it adds scrollbars only when necessary.

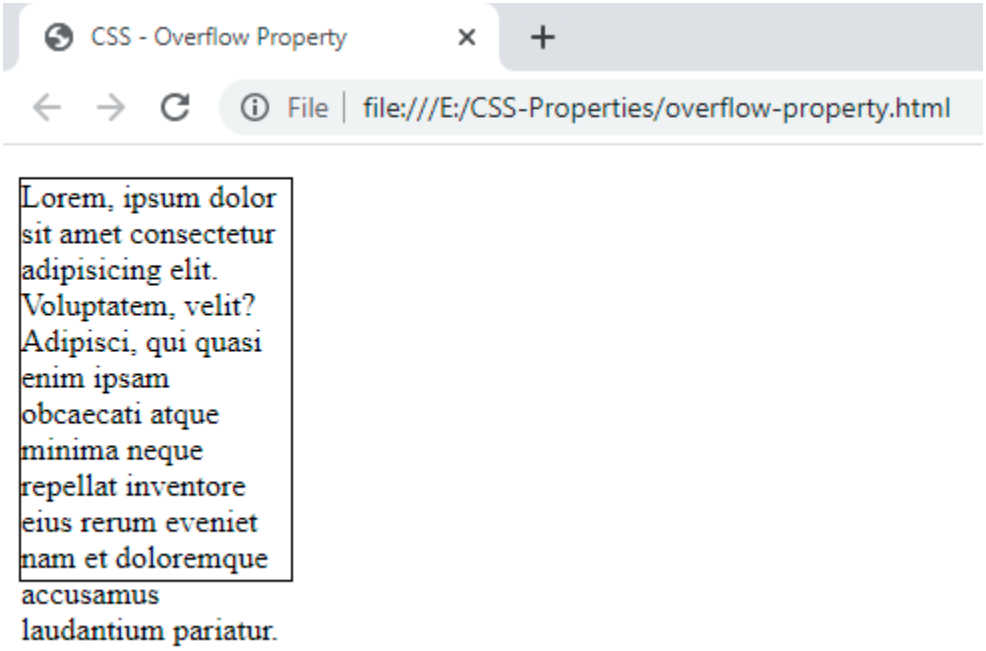
NOTE: The overflow property only works for block elements with a specified height.

Example

- Example with visible value (default value)

```
<head><style>
      .test{
         width: 10%;
         height: 200px;
         border: 1px solid black;}
   </style></head><body>
   <p class="test">Lorem, ipsum dolor sit amet consectetur adipisicing elit. Volu
ptatem, velit? Adipisci, qui quasi enim ipsam obcaecati atque minima neque repe
llat inventore eius rerum eveniet nam et doloremque accusamus laudantium pari
atur.</p>
</body>
```

Output

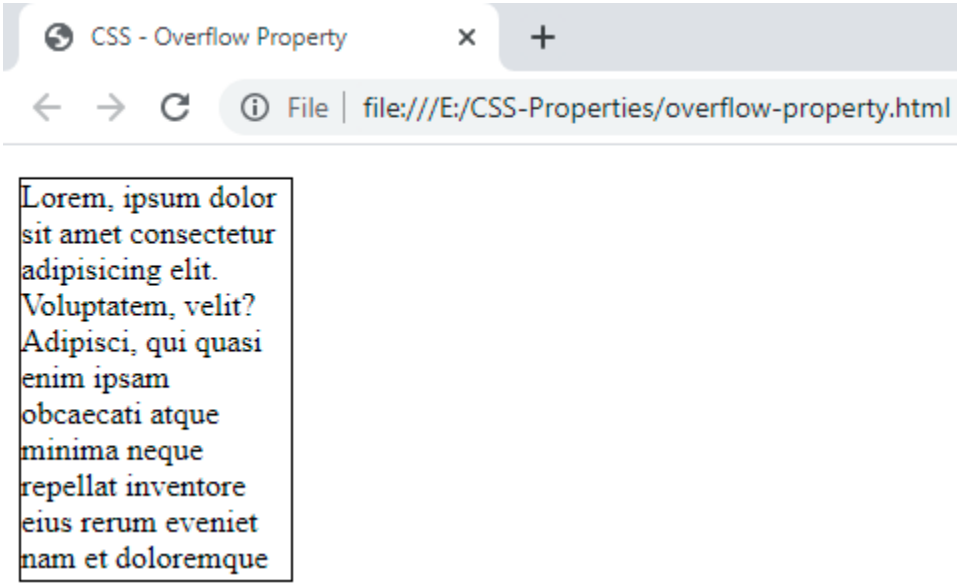


Example

- Example with hidden value.

```
<head>
    <style>
      .test{
          width: 10%;
          height: 200px;
          border: 1px solid black;
         overflow: hidden;}
    </style>
</head>
<body>
    <p class="test">Lorem, ipsum dolor sit amet consectetur adipisicing elit. Volu
ptatem, velit? Adipisci, qui quasi enim ipsam obcaecati atque minima neque repe
llat inventore eius rerum eveniet nam et doloremque accusamus laudantium pari
atur.</p>
</body>
```

Output

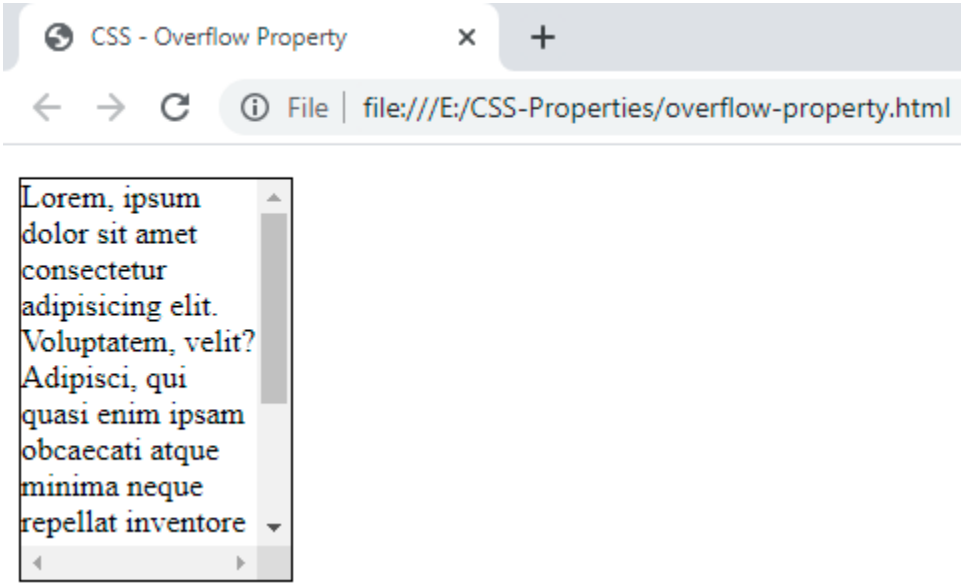


Example

- Example with scroll value. Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

```
<head><style>
      .test{
         width: 10%;
         height: 200px;
         border: 1px solid black;
        overflow: scroll;}
   </style>
</head>
<body>
   <p class="test">Lorem, ipsum dolor sit amet consectetur adipisicing elit. Volu
ptatem, velit? Adipisci, qui quasi enim ipsam obcaecati atque minima neque repe
llat inventore eius rerum eveniet nam et doloremque accusamus laudantium pari
atur.</p>
</body>
```

Output

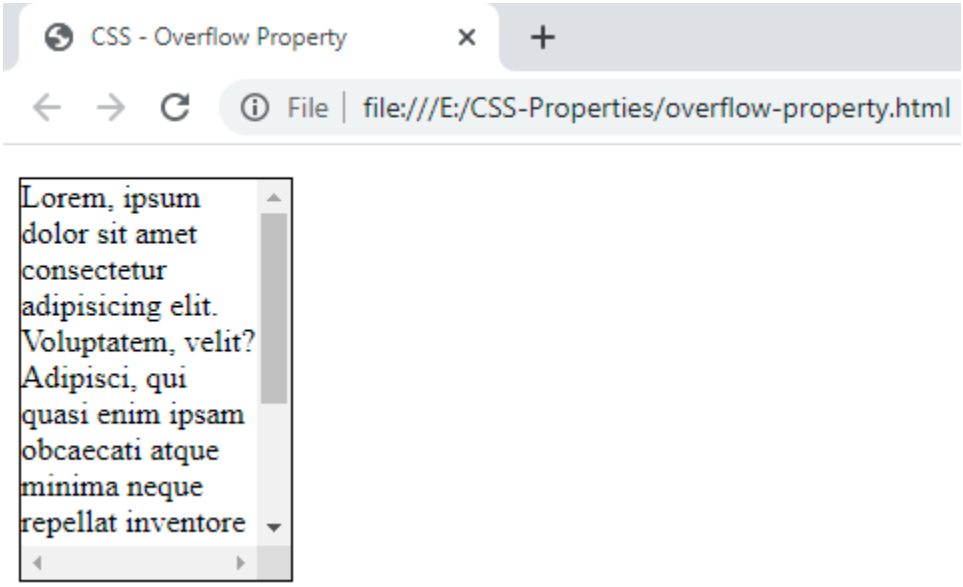


Example

- Example with auto value.

```
<head><style>
      .test{
         width: 10%;
         height: 200px;
         border: 1px solid black;
         overflow: auto;}
   </style>
</head>
<body>
   <p class="test">Lorem, ipsum dolor sit amet consectetur adipisicing elit. Volu
ptatem, velit? Adipisci, qui quasi enim ipsam obcaecati atque minima neque repe
llat inventore eius rerum eveniet nam et doloremque accusamus laudantium pari
atur.</p>
</body>
```

Output



HTML Div Tag

- The div tag is known as a Division tag.
- HTML tag is used to **divide** the **web page into different divisions or parts**. The tag basically **acts as a container** for other **HTML** elements.
- The tag is used for **grouping the HTML elements into sections** on a webpage, which is then styled with CSS or manipulated with JavaScript.
- The tag is easily styled by using the class or id attribute. Without css there is no meaning of Div tag, it is a container only.
- Any sort of content can be put inside the tag! We can manage contents very easily.
- We know that every tag has a specific purpose e.g. p tag is used to specify paragraph, <h1> to <h6> tag are used to specify headings but the tag is just like a container unit which is used to encapsulate other page elements and divides the HTML documents into sections.
- The div tag is generally used by web developers to group HTML elements together and apply CSS styles to many elements at once. For example: If you wrap a set of paragraph elements into a div element so you can take the advantage of CSS styles and apply font style to all paragraphs at once instead of coding the same style for each paragraph element.
- To more understand how tag is useful, let's take an example where we will develop a basic webpage structure, in which we will have a top **header**, a left **sidebar**, a **main body** part, and a **footer**. We will use the tag to create these parts on our webpage.

- Div tag is a **Block** element.
- Div tag can be instead, we can create div inside div.

Example

```
<style>
      .div1 {
          width:40%;
          border: 1px solid black;
          padding: 10px;
          box-sizing: border-box; }
       .div1 p{ color: red;}
   </style>
</head>
<body>
   <div class="div1">
       <p>This is an example of Division Tag.</p>
   </div>
</body>
```

Output

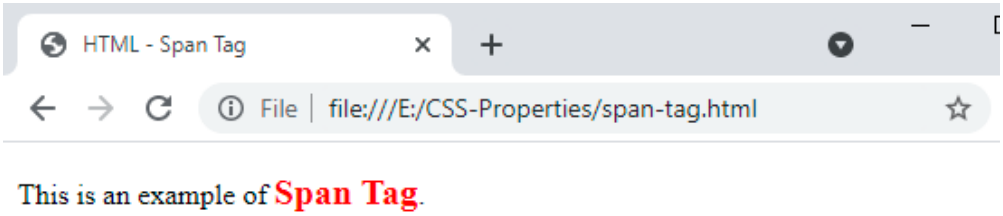
HTML Span Tag

- The tag is an inline container used to mark up a part of a text, or a part of a document.
- The tag is easily styled by CSS (using class and id attribute or inline style) or manipulated with JavaScript using the class or id attribute.
- The tag is much like the element, but is a block-level element and ** is an inline element.**

Example

```
<head><style>
      .span1
      {
         color: red;
         font-size: 20px;
         font-weight: bold;
      }
   </style>
</head>
<body>
   <p>This is an example of <span class="span1">Span Tag</span>.</p>
</body>
```

Output



CSS Margin Property

- CSS Margin property is used to define the space around elements. It is completely transparent and doesn't have any background color.
- We have the following properties to set an element margin.
    - **margin-bottom:** This Property specifies the bottom margin of an element.
    - **margin-top:** This Property specifies the top margin of an element.
    - **margin-left:** This Property specifies the left margin of an element.
    - **margin-right:** This Property specifies the right margin of an element.
    - **margin:** This Property specifies a shorthand property for setting the margin properties in one declaration.
- **Possible values are auto, length(px,pt,cm,%).**
- **Auto** is a **default** value of Margin Property.
- **Negative values are allowed to overlap content.**

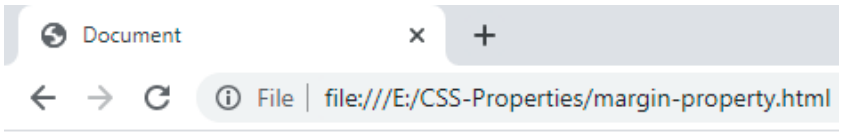
➢ Margin Top Property

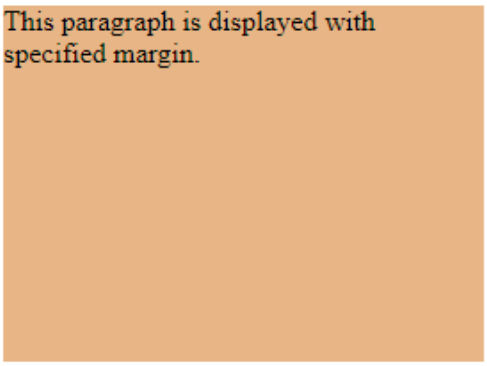
- The margin-top property sets the top margin of an element.

Example

```
<head><style>
      .div1
      {
         width: 20%;
         height: 200px;
         background-color: burlywood;
         margin-top: 100px;
      }
   </style>
</head>
<body>
   <div class="div1">
      This paragraph is displayed with specified margin.
   </div>
</body>
```

Output





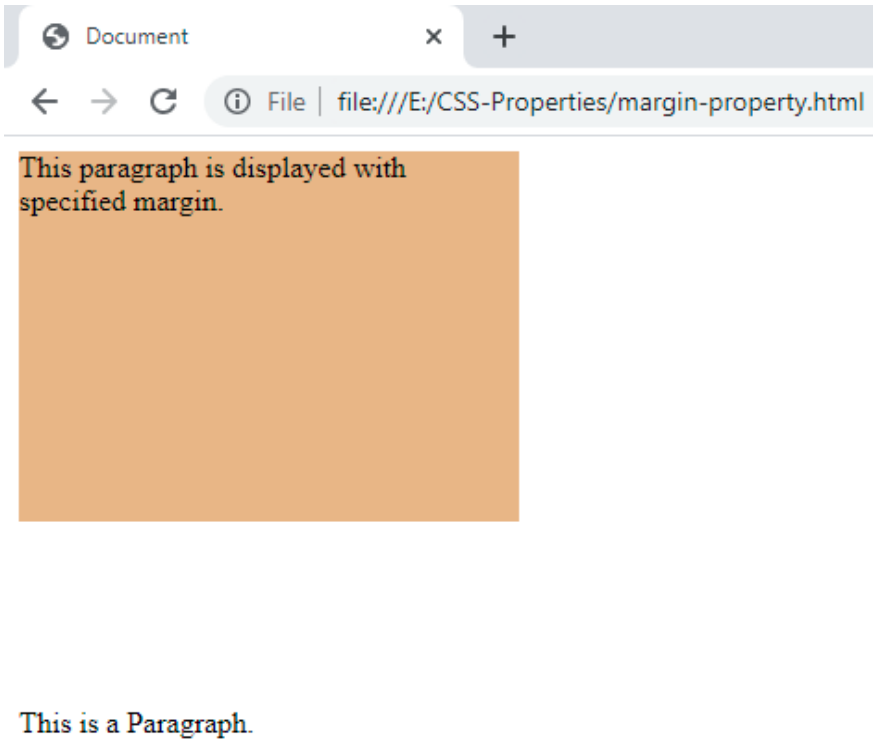
➢ Margin Bottom Property

- The margin-bottom property sets the bottom margin of an element.

Example

```
<head><style>
      .div1
      {
         width: 20%;
         height: 200px;
         background-color: burlywood;
         margin-bottom: 100px;
      }
   </style>
</head>
<body>
   <div class="div1">
      This paragraph is displayed with specified margin.
   </div>
   <p>This is a Paragraph.</p>
</body>
```

Output



➢

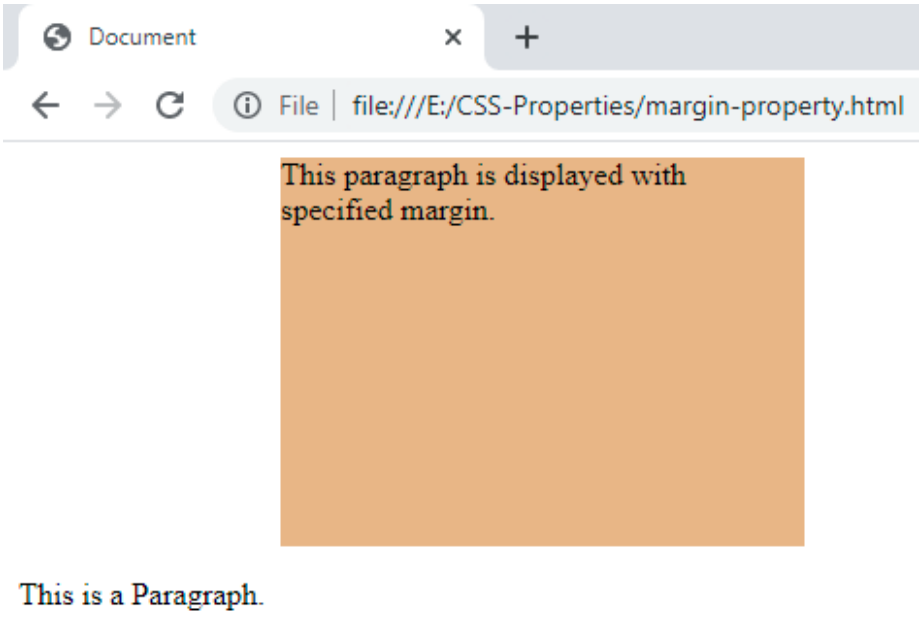
➢ Margin Left Property

- The margin-left property sets the left margin of an element.
- We usually give value in percentage (%). So we can calculate with width.

Example

```
<head><style>
    .div1
     {
        width: 20%;
        height: 200px;
        background-color: burlywood;
        margin-left: 10%;
     }
  </style>
</head>
<body>
  <div class="div1">
     This paragraph is displayed with specified margin.
  </div>
  <p>This is a Paragraph.</p>
</body>
```

Output



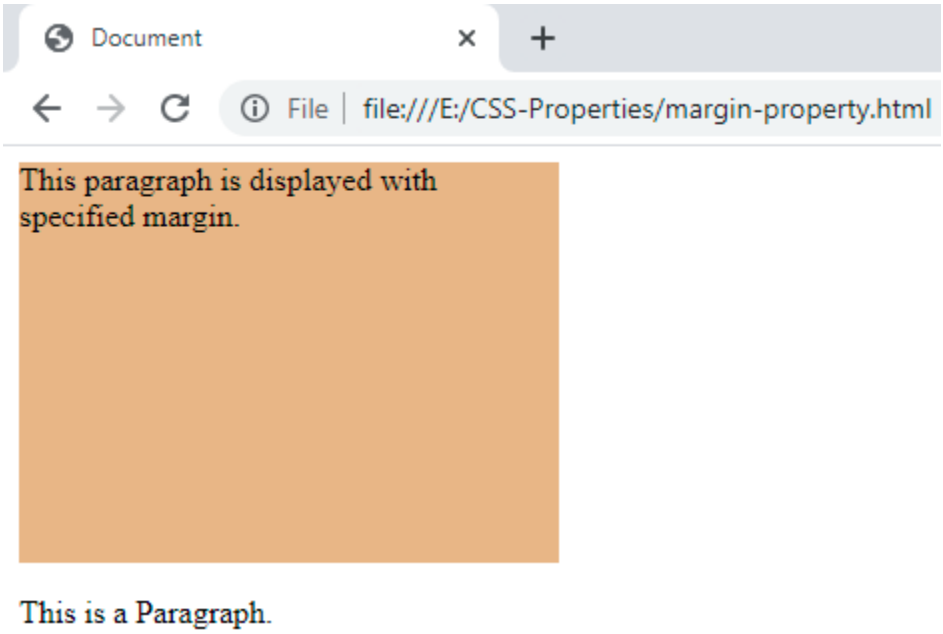
➢ Margin Right Property

- The margin-right property sets the right margin of an element.
- We usually give value in percentage (%). So we can calculate with width.

Example

```
<head><style>
    .div1
     {
        width: 20%;
        height: 200px;
        background-color: burlywood;
        margin-right: 10%;
     }
  </style>
</head>
<body>
  <div class="div1">
     This paragraph is displayed with specified margin.
  </div>
  <p>This is a Paragraph.</p>
</body>
```

Output



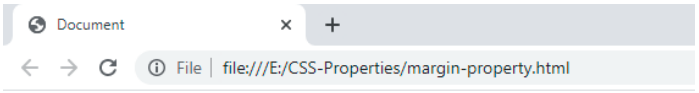
➢ Margin Property

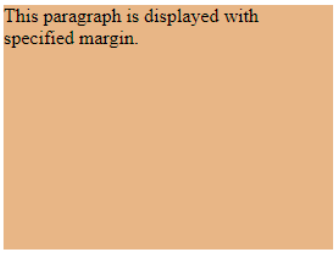
- To shorten the code, it is possible to specify all the margin properties in one property.
- The margin property is a shorthand property for the following individual margin properties:
  - **margin-top**
  - **margin-right**
  - **mrgin-bottom**
  - **margin-left**
- The following example shows how to use all the four properties into a single property. This is the most frequently used property to apply margin around any element.

Example

```
<head><style>
      .div1{
         width: 20%;
         height: 200px;
         background-color: burlywood;
         margin:50px 10%;}
</style></head>
<body>
   <div class="div1">
      This paragraph is displayed with specified margin.
   </div>
   <p>This is a Paragraph.</p>
</body>
```

Output





This is a Paragraph.

If the margin property has four values:

- margin: 10px 2% 15px 4%;
  - top margin is 10px
  - right margin is 2%
  - bottom margin is 15px
  - left margin is 4%

If the margin property has three values:

- margin: 10px 2% 15px;
  - top margin is 10px
  - right and left margins are 2%
  - bottom margin is 15px

If the margin property has two values:

- margin: 10px 2%;
  - top and bottom margins are 10px
  - right and left margins are 2%

If the margin property has one value:

- margin: 10px;
  - all four margins are 10px

  (we usually define margin-left and margin-right in %)

CSS Padding Property

- **CSS Padding property** is used *to* define the space between the element content and the element border.
- It is different from CSS margin in the way that CSS margin defines the space around elements. CSS padding is affected by the background colors. It clears an area around the content.
- We have the following properties to set an element padding.
  - **padding-bottom:** This Property specifies the bottom padding of an element.
  - **padding-top:** This Property specifies the top padding of an element.
  - **padding-left:** This Property specifies the left padding of an element.
  - **padding-right:** This Property specifies the right padding of an element.
  - **padding:** This Property specifies a shorthand property for setting the padding properties in one declaration.
- **Possible values are auto, length(px,pt,cm,%).**
- **Negative values are not allowed.**

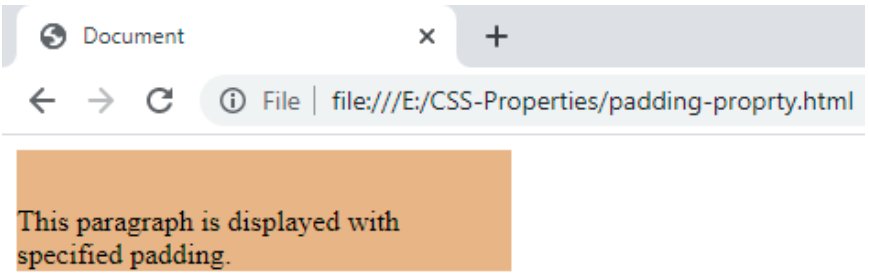
➢ Padding Top Property

- The padding-top property sets the top padding of an element.

Example

```
<head><style>
      .div1{
         width: 20%;
         padding-top: 30px;
         background-color: burlywood;}
 </style></head>
<body>
   <div class="div1">
      This paragraph is displayed with specified padding.
   </div>
</body>
```

Output



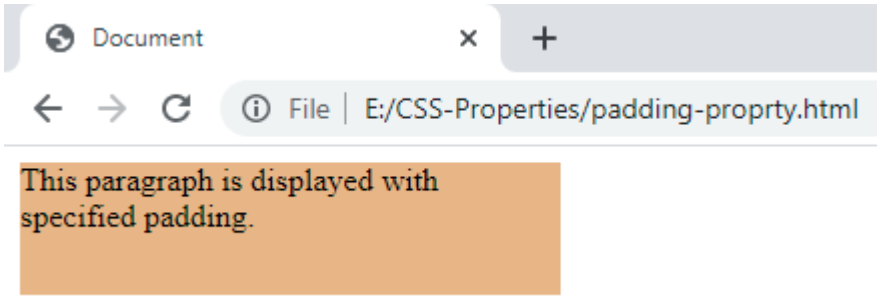
➢ Padding Bottom Property

- The padding-bottom property sets the bottom padding of an element.

Example

```
<head><style>
      .div1{
         width: 20%;
         background-color: burlywood;
         padding-bottom: 100px;}
   </style>
</head>
<body>
   <div class="div1">
      This paragraph is displayed with specified padding.
   </div>
</body>
```

Output



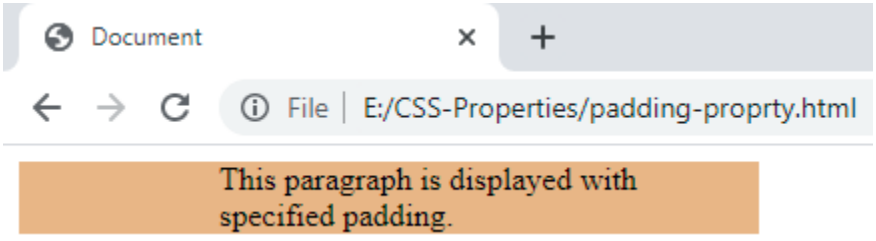
➢ Padding Left Property

- The padding-left property sets the left padding of an element.

Example

```
<head><style>
    .div1
     {
        width: 20%;
        background-color: burlywood;
        padding-left: 100px;
     }
  </style>
</head>
<body>
  <div class="div1">
     This paragraph is displayed with specified padding.
  </div>
 </body>
```

Output



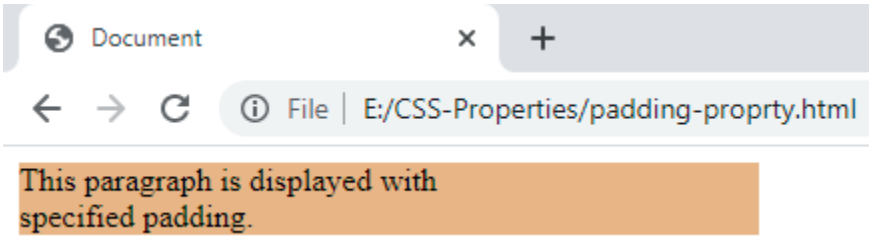
➢ Padding Right Property

- The padding-right property sets the right padding of an element.

Example

```
<head><style>
    .div1
     {
        width: 20%;
        background-color: burlywood;
        padding-right: 100px;
     }
  </style>
</head>
<body>
  <div class="div1">
     This paragraph is displayed with specified padding.
  </div>
</body>
```

Output



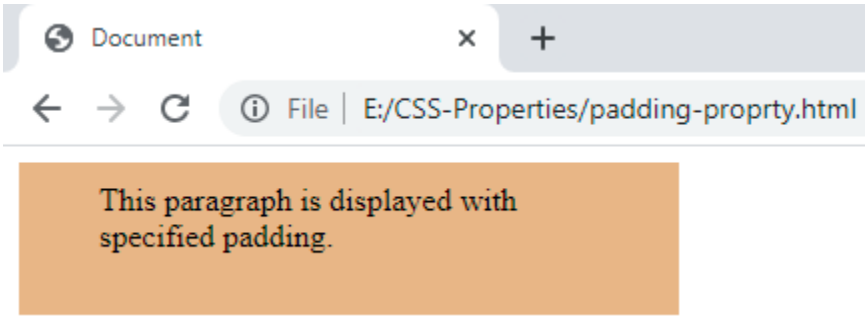
➢ Padding Property

- To shorten the code, it is possible to specify all the padding properties in one property.
- The padding property is a shorthand property for the following individual padding properties:
  - **padding-top**
  - **padding -right**
  - **padding -bottom**
  - **padding-left**
- The following example shows how to use all the four properties into a single property. This is the most frequently used property to apply padding in any element.

Example

```
<head><style>
      .div1{
         width: 20%;
         background-color: burlywood;
         padding:10px 20px 30px 40px;}
</style></head>
<body>
   <div class="div1">
      This paragraph is displayed with specified padding.
   </div>
 </body>
```

Output



If the padding property has four values:

- padding: 10px 20px 30px 40px;
  - top padding is 10px
  - right padding is 20px
  - bottom padding is 30px
  - left padding is 40px

If the padding property has three values:

- padding: 10px 20px 30px;
  - top padding is 10px
  - right and left paddings are 20px
  - bottom padding is 30px

If the padding property has two values:

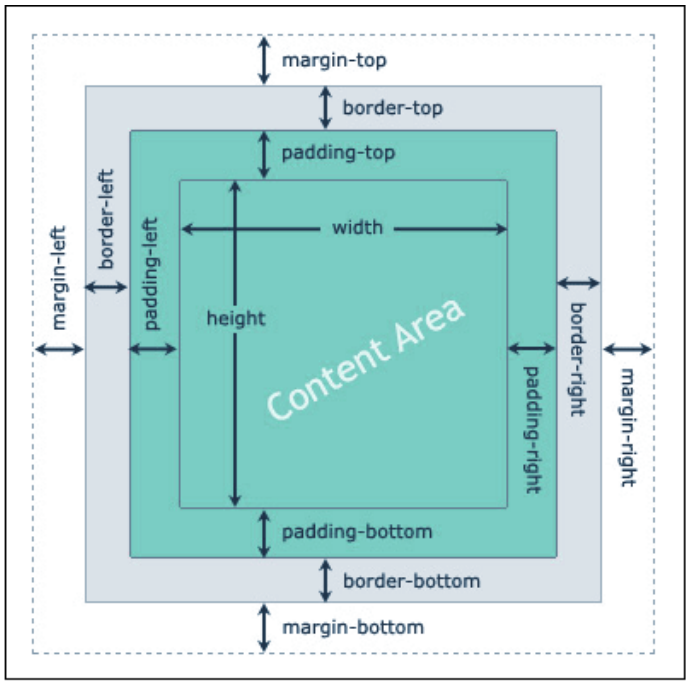
- padding: 10px 20px;
  - top and bottom paddings are 10px
  - right and left paddings are 20px

If the padding property has one value:

- padding: 10px;
  - all four paddings are 10px

CSS Box Model

- Each and every element created using any HTML tag, is a box.
- In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



- The CSS box model contains the different properties in CSS. These are listed below.
  - **Border:** A border that goes around the padding and content
  - **Margin:** This is the empty space outside the border. The margin is transparent
  - **Padding:** This is the empty space around the content (between border and content) inside the element. The padding is transparent
  - **Content:** The content of the box, where text and images appear

Width and Height of an Element

- When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the full size of an element, you must also add padding, borders and margins.
- The specific area that an element box may occupy on a web page is measured as follows:

Size of the box	Properties of CSS
Height	height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom
Width	width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

Example

```
<style>
      .div1
      {
         width: 30%;
         height: 200px;
         border: 1px solid black;
         margin:10px 10%;
         padding: 20px;
      }
   </style>
</head>
<body>
   <div class="div1">
      Example of CSS Model
   </div>
</body>
```

In the above example, HTML element with class as div1 must have a defined width of 30% and height of 200px. But the actual height and width occupied by this element will be different, as padding, border and margin must also be considered.

Actual Width: Width(30%) **+** left & right padding(20px + 20px) **+** left and right border(1px + 1px) **+** left and right margin(10% + 10%) = 50%+42px

Actual Height: Height(200px) **+** top & bottom padding(20px + 20px) **+** top and bottom border(1px + 1px) **+** top and bottom margin(10px + 10px) = 262px

CSS Box Sizing Property

- The CSS box-sizing is a way to calculate the total height and width of an HTML element. Generally, when we specify the height and width property to any HTML element along with padding and margin. The value of padding and border are added to the values of height and width of the element and the element takes more space than its actual size.
- In CSS2, the height and width of an element are calculated as:

  width + padding-left + padding-right + border-left + border-right = actual width
  height + padding-top + padding-bottom + border-top + border-bottom = actual height

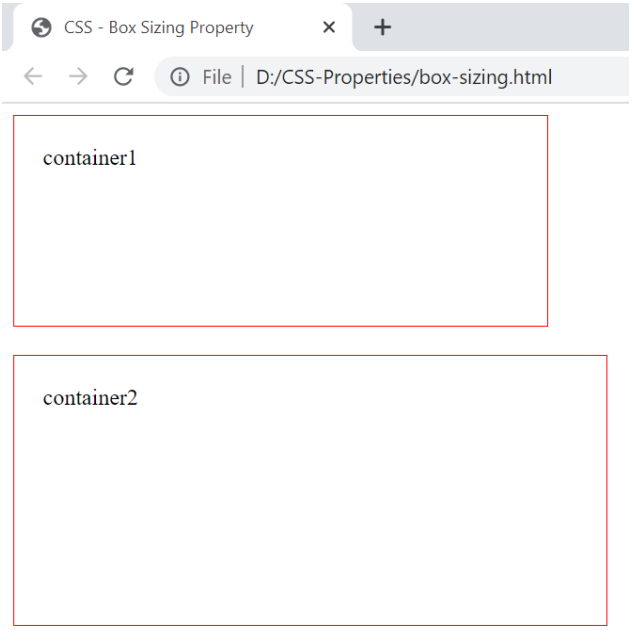
- The above formula specifies that when we set the padding and border of an HTML element, it acquires a little bit more space than the actual measurements you have set for that element. This is because the border and padding property of the element is not included in the specified height and width of the element.

Example

In this example, we have created two Boxes using the ** ** element. Then, we have applied the border and **padding** along with the height and width properties to both the boxes. In the second box, we have also applied the box-sizing: border-box; property. So, the **first box** gets **bigger in size than its actual size**, and the **second box**.

```
<head><style>
      .div1{
        width: 30%;
        height: 150px;
        padding: 20px;
        border: 1px solid red;
        margin-bottom: 20px;}
      .div2{
        width: 30%;
        height: 150px;
        padding: 20px;
        border: 1px solid red;
        box-sizing: border-box;}
</style></head><body>
   <div class="div1">container1</div>
   <div class="div2">container2</div>
</body>
```

Output



CSS Display Property

- CSS display is the most important property of CSS which is used to control the layout of the element. This property specifies how the HTML elements are going to display on a web page.
- Every HTML element has a default display value depending on what type of element it is. The **default display value** for most elements is **block** or **inline**.
- The web page considered every HTML element (div, p, heading) as a rectangular box and the CSS display property helps to set out the positioning of these boxes.
- Commonly used values for display property is inline, block, inline-block and none.
- A block-level element always starts on a new line and takes up the full width available (according to the browser width). , <h1>-<h6>, <p>, etc. are block-level elements.
- An inline element does not start on a new line and only takes up as much width as necessary. <a>, <img>, , etc are inline elements.
- As mentioned, every element has a default display value. However, you can override this.
- Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

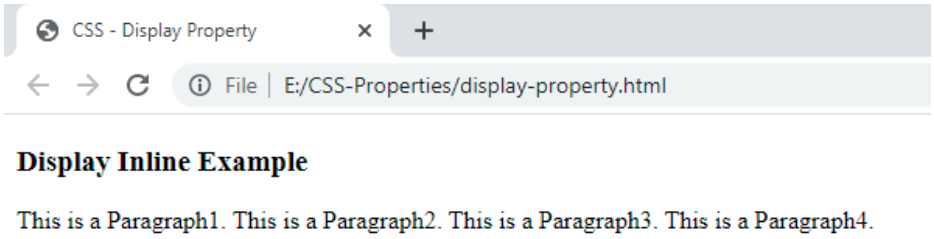
➢ Display: inline

- **The inline display** value sets out the elements as an inline element which means the elements are place side by side or within a line (horizontally). This property provides only as much space as any elements need.
- **The height, width, top-bottom margin, top-bottom padding property are not accepted by the HTML element whenever the display: inline property is specified for that element.**
- We usually use display: inline, when we would like to arrange Block-level elements in a single line or side by side.

Example

```
<style>p{display: inline;}</style></head>
<body>
   <h3>Display Inline Example</h3>
   <p>This is a Paragraph1.</p>
   <p>This is a Paragraph2.</p>
   <p>This is a Paragraph3.</p>
   <p>This is a Paragraph4.</p></body>
```

Output



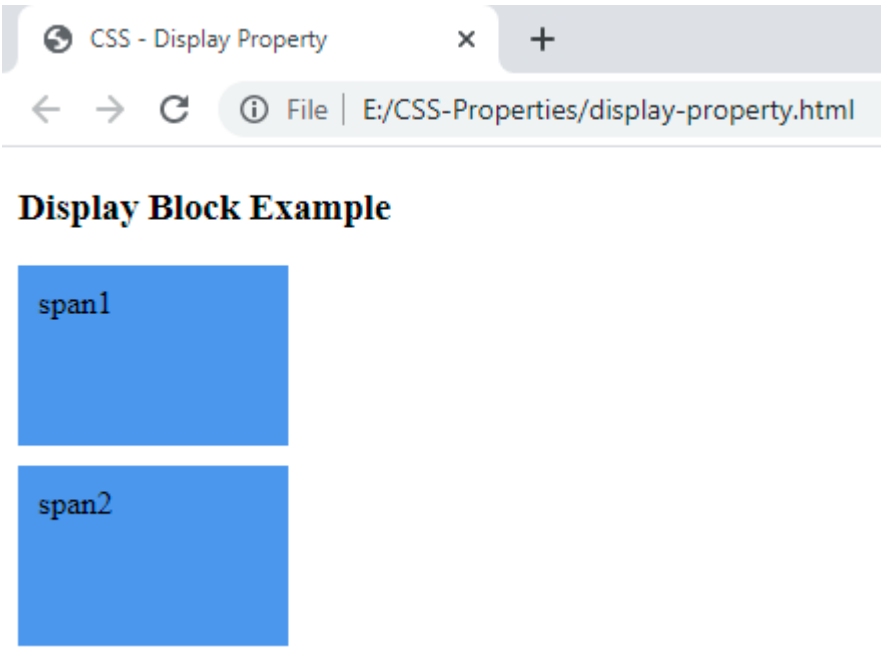
➢ Display: block

- **The CSS display block value** consider the full width of the screen. These values place the content of each element in a new line and allow them to act as a separate block. There are some HTML elements whose default display value is block these elements are , <p>, <ul>, etc.
- **The height, width, top-bottom margin, top-bottom padding property are accepted by the HTML element whenever the display: block property is specified for that element.**
- We usually use display: block, when we would like to arrange inline elements in a new line.

Example

```
<style>
      span
      {
         display: block;
         background-color: cornflowerblue;
         margin-bottom: 10px;
         padding:10px;
         box-sizing: border-box;
         width: 10%;
         height: 90px;
      }
   </style>
</head>
<body>
   <h3>Display Block Example</h3>
   <span>span1</span>
   <span>span2</span>
</body>
```

Output



➢ Display: inline-block

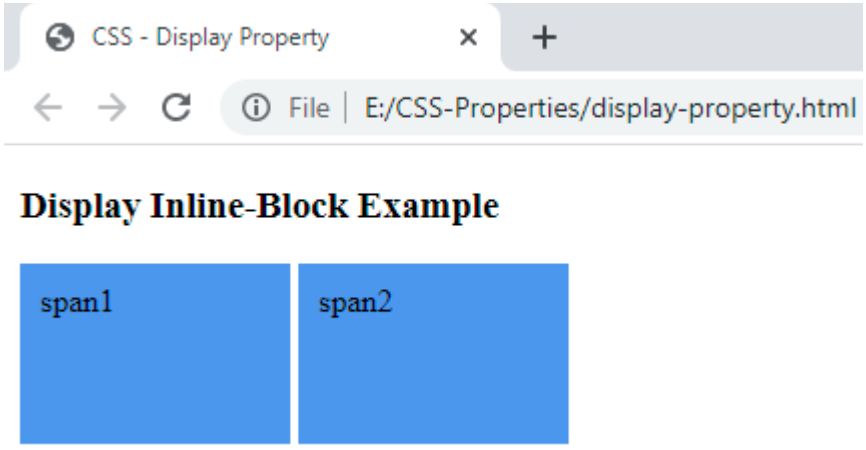
- **The inline-block** offers the property of both the values **inline** and **block**. This value is used whenever we want to place

our content side by side (inline) and also set the height, width, top-bottom margins and top-bottom paddings of the element.

Example

```
<style>
      span{
         display: inline-block;
         background-color: cornflowerblue;
         margin-bottom: 10px;
         padding:10px;
         box-sizing: border-box;
         width: 10%;
         height: 90px;
      }
   </style>
</head>
<body>
   <h3>Display Block Example</h3>
   <span>span1</span>
   <span>span2</span>
</body>
```

Output



➢ Display: none

- **The CSS display none value** removes the HTML element and also does not display any blank space at the place of the removed element. The removed elements exist in an HTML structure but not be display on a web page.

Example

```
<style>
      span
      {
         display: inline-block;
         background-color: cornflowerblue;
         margin-bottom: 10px;
         padding:10px;
         box-sizing: border-box;
         width: 10%;
         height: 90px;}
      .span2
      {
         display: none;
      }
   </style>
</head>
<body>
   <h3>Display None Example</h3>
   <span>span1</span>
   <span class="span2">span2</span>
   <span>span3</span>
</body>
```

Output

CSS Position Property

- Positioning of elements is extremely important. How you position the elements of your webpage are extremely important from the point of view of usability i.e., how comfortable a user feels using your website.
- CSS can help you a lot when it comes to positioning stuff on your website. The **CSS position property** is used to set position for an element. It is also used to place an element behind another and also useful for scripted animation effect.
- You can position an element using the top, bottom, left and right properties. These properties can be used only after position property is set first.
- **Possible values for position property are relative, absolute, fixed or sticky (css3).**
- **Static** is a **default** value for **position Property.**

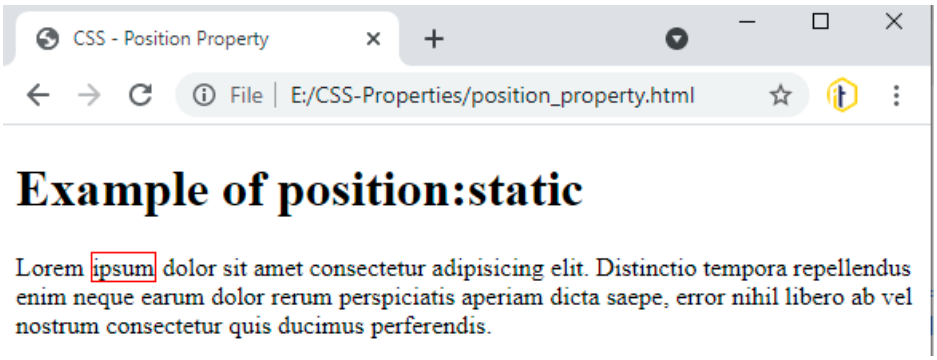
CSS Position Static Value

- **This is a by default position for HTML elements.** Explicitly specifying static is pretty rare, and can be used to get rid of any inherited position value.
- An element with position: static; is not positioned in any special way, it is always positioned according to the normal flow of the page.
- **It is not affected by the top, bottom, left and right properties.**

Example

```
<head><style>
      .span1
      {
         position: static;
         top: 100px;
         border: 1px solid red;
      }
   </style></head>
<body>
   <h1>Example of position:static</h1>
   <p>Lorem <span class="span1">ipsum</span> dolor sit amet con
sectetur adipisicing elit. Distinctio tempora repellendus enim neque ear
um dolor rerum perspiciatis aperiam dicta saepe, error nihil libero ab v
el nostrum consectetur quis ducimus perferendis.</p>
</body>
```

Output



➢ Position: relative

- **An element with position: relative;** is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.
- Setting an element's position to relative but not specifying any other attributes like top, bottom, left or right will have no particular effect on the element's position.

Example

```
<head><style>
      .span1{
         position: relative;
         top: 100px;
         left: 50px;
          border: 1px solid red;}
   </style></head>
<body>
   <h1>Example of position:relative</h1>
   <p>Lorem <span class="span1">ipsum</span> dolor sit amet con
sectetur adipisicing elit. Distinctio tempora repellendus enim neque ear
um dolor rerum perspiciatis aperiam dicta saepe, error nihil libero ab v
el nostrum consectetur quis ducimus perferendis.</p>
</body>
```

Output

➢ Position: absolute

- **An element with position: absolute;** allows you to place an element exactly where you want it to be placed. Positioning attributes top, bottom, left and right are used to determine the exact location.
- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However, if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- An absolute element does not leave a gap in the page where it would normally have been located.

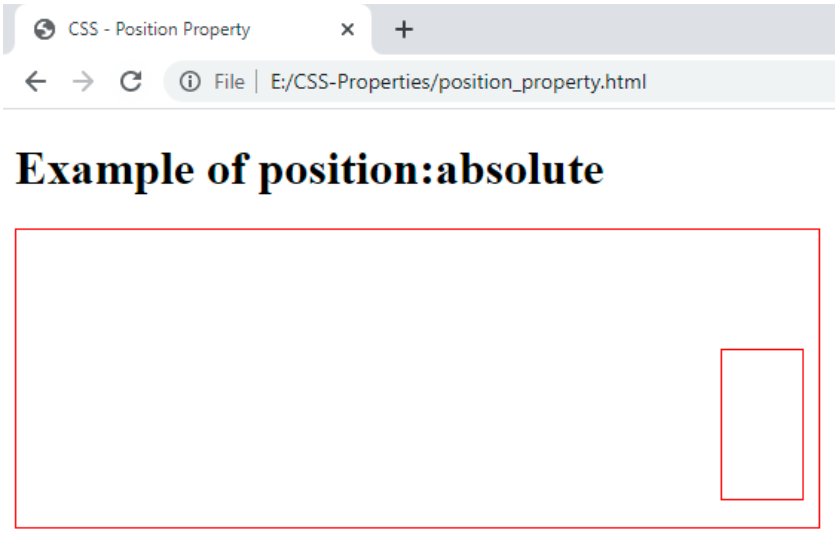
Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Example

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed)

```
<style>
      .outerContainer{
         width: 40%;
         height: 200px;
         position: relative;
         border: 1px solid red;}
      .innerContainer{
         position: absolute;
         width: 10%;
         height: 100px;
         border: 1px solid red;
         right: 10px;
         top: 80px;}</style></head>
<body>
   <h1>Example of position:absolute</h1>
   <div class="outerContainer">
      <div class="innerContainer"></div>
   </div>
</body>
```

Output



- if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

```
<style>
     .span1
     {
        position: absolute;
        top: 100px;
        left: 50px;
     }
   </style>
</head>
<body>
   <h1>Example of position:absolute</h1>
   <p>Lorem <span class="span1">ipsum</span> dolor sit amet con
sectetur adipisicing elit. Distinctio tempora repellendus enim neque ear
um dolor rerum perspiciatis aperiam dicta saepe, error nihil libero ab v
el nostrum consectetur quis ducimus perferendis.</p>
</body>
```

Output

➢ Position: fixed

- **An element with position: fixed;** is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
- A fixed element does not leave a gap in the page where it would normally have been located.

Example

```
<head><style>
      .span1{
         position: fixed;
         top: 150px;
         left: 0px;
         border: 1px solid red;}
   </style>
</head>
<body style="height: 1500px;">
   <h1>Example of position:fixed</h1>
   <p>Lorem <span class="span1">ipsum</span> dolor sit amet consec
tetur adipisicing elit. Distinctio tempora repellendus enim neque earum d
olor rerum perspiciatis aperiam dicta saepe, error nihil libero ab vel nostr
um consectetur quis ducimus perferendis.</p>
</body>
```

Output

CSS z-index Property

- When we play around with positioning so much, there will be situations when elements will overlap with each other. So, in situations of overlap, which element will appear on the top of the other?
- **The z-index** defines the stack order of an element. An element with a greater z-index will always be on top of elements with a lower z-index. An element can have a positive or a negative z-index value.
- **Auto** is a **default** value of **z-index property**.
- **Possible values are auto and integer (negative values are also allowed).**
    - **number:** It means that the element's stack level is set to the given value. It allows negative values.

- **auto:** It means that the order of the stack is equivalent to the parent, i.e., default.

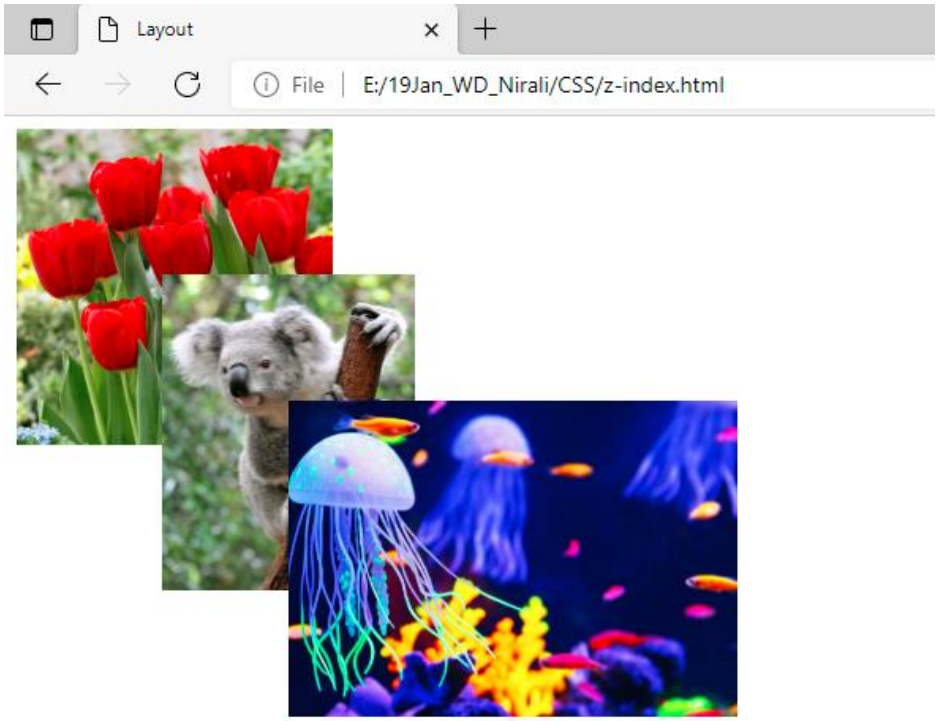
Note: z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of display: flex elements).

If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

Example

```
 <head><style>
.img1{
        position: absolute;
        top: 180px;
        left: 180px;
        z-index: 1000;}
 .img2{
        position: absolute;
        top: 100px;
        left: 100px;
        z-index: 20;}
 .img3{
        position: absolute;
        z-index: 10;}
   </style></head>
<body>
   <img src="../../5feb_WD_Nirali/images/Koala.jpg" alt="Koala" height
="200" width="200" class="img1">
   <img src="../../5feb_WD_Nirali/images/tulip.jpg" alt="Tulip" height="
200" width="200" class="img2">
   <img src="../../5feb_WD_Nirali/images/jellyfish.jpg" alt="jellyfish" hei
ght="200" width="200" class="img3">
</body>
```

Output



CSS Opacity Property

- **The CSS opacity property** is used to specify the transparency of an element on a scale of **0.0** to **1.0**. Using this property, we can set elements to be completely opaque (**visible**), fully transparent (**hidden**), or translucent (**partially visible**). It takes a numeric value lies between 0 and 1. Where 0 defines fully transparent and 1 defines completely visible. Opacity values between 0 and 1, such as 0.2, 0.4, 0.6, etc., change an element from transparent to opaque by gradually increasing the decimal value.
- **1** is a **default value** for **Opacity Property**.

Note: The older versions of IE use filter: alpha(opacity=x). Here x value varies from 0 to 100. Lower value produces the greater opacity.

When using the opacity property to add transparency to the background of an element, all of its child elements become transparent as well. This can make the text inside a fully transparent element hard to read. If you do not want to apply opacity to child elements, use RGBA color values instead.

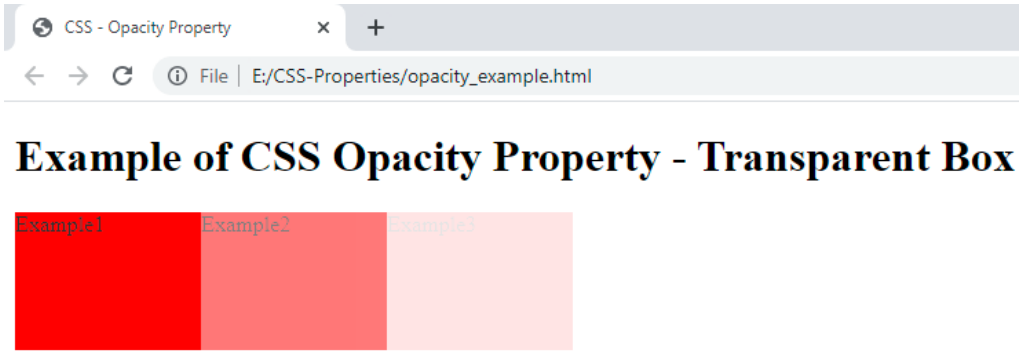
Example

➢ Transparent Box

- When using the **opacity property** to add transparency to the background of an element, all of its child elements inherit the same transparency. This can make the text inside a fully transparent element hard to read.

```
<head><style>
      .div{
         width: 10%;
         height: 100px;
         background-color: red;
         float: left;}
      .div1{opacity: 0.8;}
      .div2{opacity: 0.5;}
      .div3{opacity: 0.1;}
   </style></head>
<body>
   <h1>Example of CSS Opacity Property - Transparent Box</h1>
   <div class="div div1">Example1</div>
   <div class="div div2">Example2</div>
   <div class="div div3">Example3</div>
</body>
```

Output:

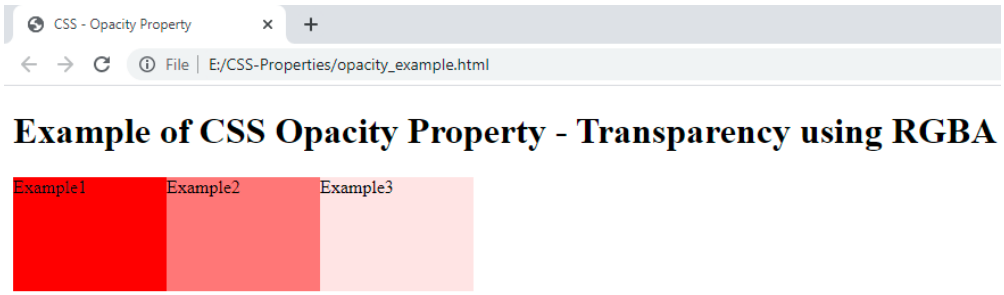


➢ Transparency using RGBA

- If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text.
- An RGBA color value is specified with: rgba(red, green, blue, *alpha*). The *alpha* parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
<head><style>
      .div{
         width: 10%;
         height: 100px;
         background-color: red;
         float: left;}
      .div1{ background-color: rgba(255, 0, 0, 0.8);}
      .div2{background-color: rgba(255, 0, 0, 0.5);}
      .div3{background-color: rgba(255, 0, 0, 0.1);}
   </style></head>
<body>
   <h1>Example of CSS Opacity Property - Transparent Box</h1>
   <div class="div div1">Example1</div>
   <div class="div div2">Example2</div>
   <div class="div div3">Example3</div>
</body>
```

Output



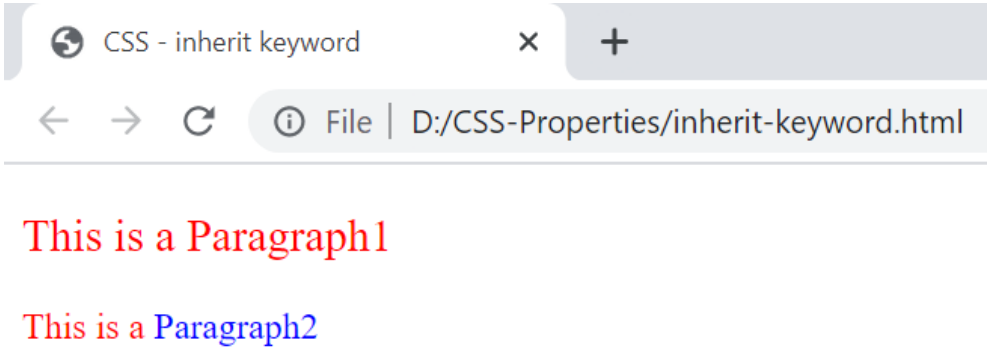
CSS inherit keyword

- **The inherit keyword** specifies that a property should inherit its value from its parent element.
- The inherit keyword can be used for any CSS property, and on any HTML element.

Example

```
<style>
      p{color: red;}
      .para{font-size: 20px;}
      span{color: blue;}
      .para span{color: inherit;}
   </style></head><body>
   <p class="para">This is a <span>Paragraph1</span></p>
   <p>This is a <span>Paragraph2</span></p></body>
```

Output



CSS initial keyword

- **The initial keyword** is used to set a CSS property to its default value.
- The initial keyword can be used for any CSS property, and on any HTML element.

Note: The initial keyword is not supported in Internet Explorer 11 and earlier versions, or in Opera 15 and earlier versions.

Example

```
<style>
      .div1  {
         color: red;
         border: solid;}
      h2{color: initial;}
</style></head>
<body>
   <div class="div1">
      <h2>Heading2</h2>
      <p>This is a Paragraph.</p>
      <span>This is a Span.</span>
   </div></body>
```

Output