More CSS Concepts

OVERFLOW-WRAP

If there is an overflow of text in an element, it can be fixed using the **overflow-wrap** property which allows the element to break the ***breakable word*** in order to wrap it in the next line.

Syntax:

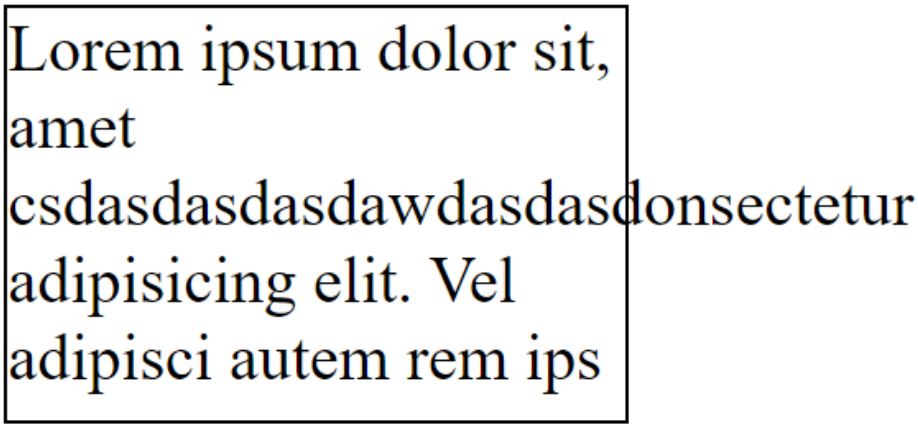
```
overflow-wrap: break-word;
overflow-wrap: normal;
overflow-wrap: inherit;
overflow-wrap: initial;
```

- **break-word:** It allows words to be broken
- **normal**: The browser will break words according to the normal rules. It's the default value.
- **initial**: This will set this property to its default value
- **inherit**: The element will inherit this property from its parent element.

For eg.,

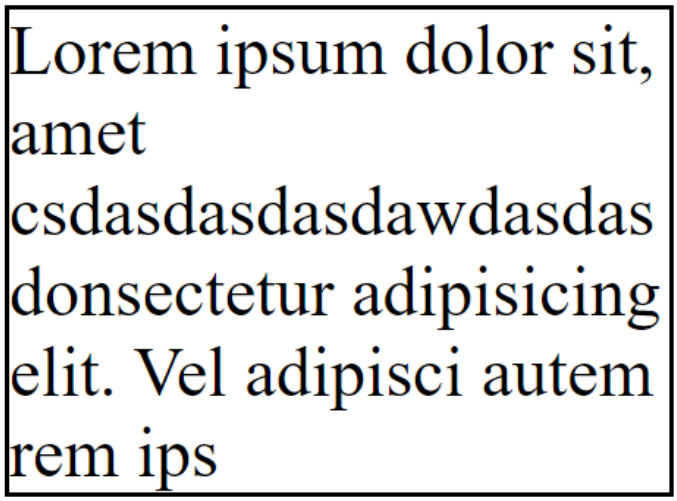
```
<head>

<style>
    p {
        width: 150px;
        height: 100px;
        border: 1px solid black;
      }
</style>

</head>

<body>

<p>
    Lorem ipsum dolor sit, amet csdasdasdasdawdasdasdonsectetur adipisicing elit.
Vel adipisci autem rem ips
</p>

</body>
```

Output:



As you can see that the text is ***overflowing out of the container*** and it's not looking good.
This can be fixed by using **overflow-wrap:**

```
width: 150px;
height: 100px;
border: 1px solid black;
overflow-wrap: break-word;
```



CSS GRADIENTS

With **CSS gradients**, you can create ***beautiful transitions between two colors or more***.
Generally, gradients are used as ***backgrounds for your HTML page***.
Gradients are of two types:

- **Linear Gradients:** Linear gradients are color bands that progress in straight lines. ie., up, down, left, right, and diagonally
- **Radial Gradients:** They radiate from a central point; they are defined by their center.

Linear Gradients:

This is the most basic type of gradient as here, you just **need to mention two colors.**

For eg.,

```
  div {
        width: 50px;
        height: 50px;
        background: linear-gradient(black, red);
    }
```

Output:

You can also create horizontal ones:

```
div {
    width: 50px;
    height: 50px;
    background: linear-gradient(to right, black, red);
}
```

You can also create **diagonal-running gradients** by ***specifying both horizontal and vertical starting positions:***

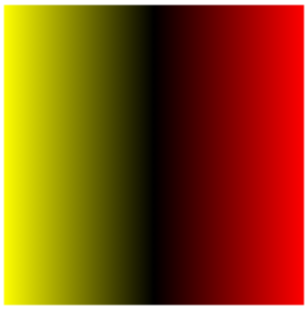
```
div {
    width: 50px;
    height: 50px;
    background: linear-gradient(to top right, black, red);
}
```

To have a **greater amount of control** on the direction, you can specify ***the angle in degrees,*** where a **positive angle means clockwise direction** and a **negative value means anticlockwise direction*:***

```
div {
    width: 50px;
    height: 50px;
    background: linear-gradient( 60deg, black, red);
}
```

You can also specify more than two colors:

```
div {
    width: 50px;
    height: 50px;
    background: linear-gradient( to right, yellow, black, red);}
```



You can use as many as colors you want. As you can see in the above example, ***by default, all the colors are equally spaced along the length of the gradient.***

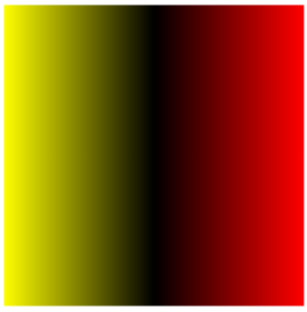
If needed, you can **change the location of each color** and **hard code the values to achieve the desired effect**.

If you give the values in percentages then ***0% means the starting point*** and ***100% means the ending point.*** You can also use values outside this range and if you don't specify the position for a color, it'll be automatically assigned to it.

For eg.,

```
div {
    width: 50px;
    height: 50px;
    background: linear-gradient( yellow 0%, black 50%, red 100%);
}
```

Will produce this same effect:



There are ***repeating-linear-gradients*** as well, which will take the same values as linear gradients, but they'll **repeat the gradient effect repeatedly, so as to cover the entire space of the container.**

For eg.,

```
div {
    width: 50px;
    height: 50px;
    background: repeating-linear-gradient( yellow, black 5%, red 20%);
}
```

Output:

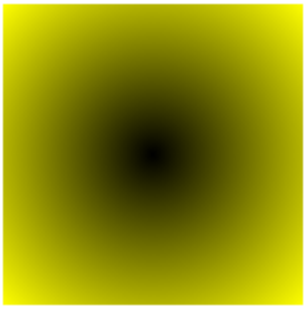
Radial Gradients:

Similar to the linear gradients, you can also create radial ones. The only difference is that they start to radiate from the element's center, ending towards the edges.
All the rules of the linear gradient are also applicable to the radial one.

For eg.,

```
div {
     width: 50px;
     height: 50px;
     background: radial-gradient(red, black, yellow);
}
```

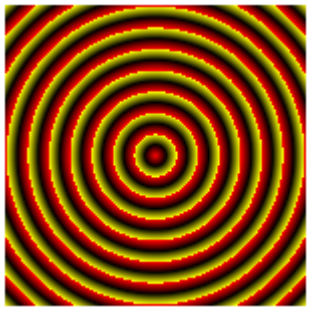
Output:



Similarly, you can create repeating-radial-gradients:

```
div {
    width: 50px;
    height: 50px;
    background: repeating-radial-gradient(red 0%, black 5%, yellow 10%);
}
```

Output :



Extra:

Read this to get more information on CSS gradients:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Images/Using_CSS_gradients

LINK RELATED PSEUDO CLASSES

You probably know about ***:hover*** and ***:nth-child*** pseudo-classes. We'll discuss some more pseudo-classes related to hyperlinks!

To style and design what happens with hyperlinks as they are clicked, CSS has these pseudo-classes depending on the current state of a link:

- **:link -** This denotes an unvisited link.
- **:active -** This state comes just after a link is clicked.
- **:visited -** This denotes links that have already been visited.
- **:target -** It's used to style a link's target element.

- **:link**

  This pseudo-class is used for unvisited links. By default, the color of unvisited links is blue.

  For eg;

```
<p>
    take me to <a href="google.com">this</a> page.
</p>
```

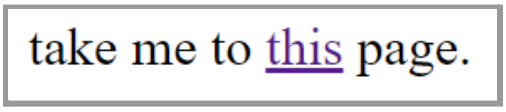
will show up as:

You can change this as per your style.

```
a:link {
    color: chartreuse;
}
```

- **:visited**

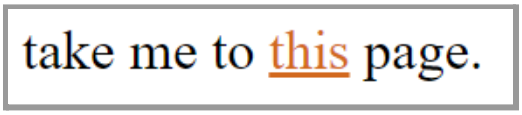
For links already visited, the default color is purple.
If we visit the link in the above example then the link will look like this:



We can use the **:visted** pseudo-class to change this:

```
a:visited {
    color: chocolate;
}
```

Output:



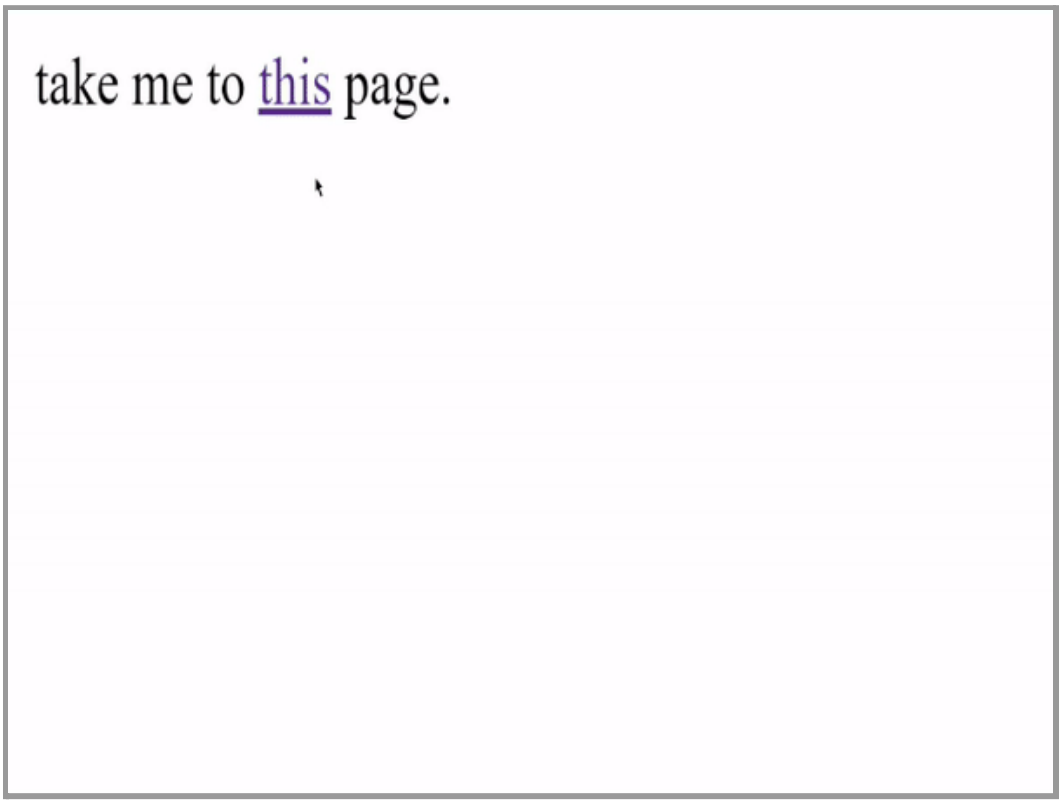
- **:active**

This state occurs when you click on a link. It is the very brief moment between someone clicking a link and the page being redirected to another

page. You can see this state easily if you click on a link and don't release the mouse button.

For eg., this code will set the color **red** for the ***active link***:

```
a:active {
    color: red;
}
```



As you can see, after clicking and holding the mouse button for a couple of seconds, the **link becomes active and its color changes to red**. As soon as the ***mouse button is released, the page is redirected to the specified website***.

- **:target**

The URLs can have URLs as a **#** followed by **an element's id,** which makes that element **a target element.**
You can use this to style a ***link's current active target element.***

Look at the HTML code below:

```
<a href="#p1">link1</a>
<a href="#p2">link2</a>
<p id="p1">I'm first</p>
<p id="p2">I'm second</p>
```

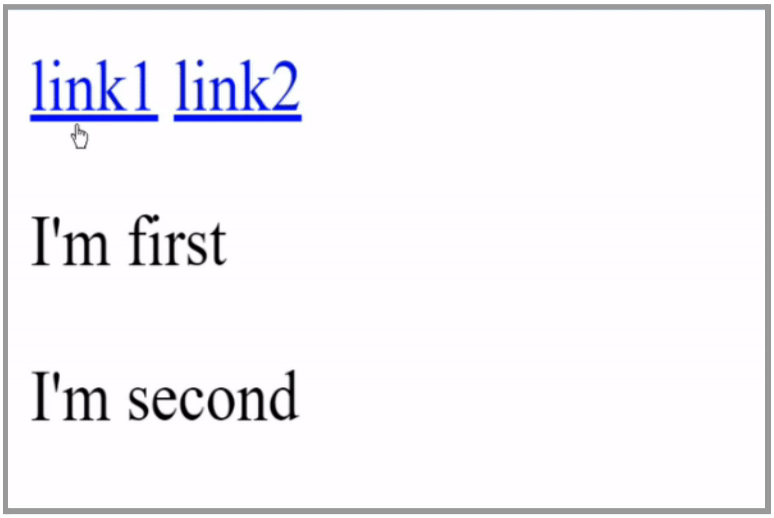
Here, we have two anchor tags followed by two <p> tags, and both the anchor tags ***contain the ids of both the <p> tags respectively, which makes both <p> tags targets for the respective links.***

Now, we can style these target elements, **when they become active.**

We can add styles like this, to style the target element, when the link is clicked:

```
#p1:target {
    background-color: darkslateblue;
}
#p2:target {
    background-color: deeppink;
}
```

Output:



As you can, as soon as the link becomes active, the styles are applied to the respective target elements.

PSEUDO ELEMENTS

If you want to style **specific parts of an element** like a ***line, or the first letter,or insert stuff before and after elements***, you can use something called **pseudo-elements.**

Syntax:

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

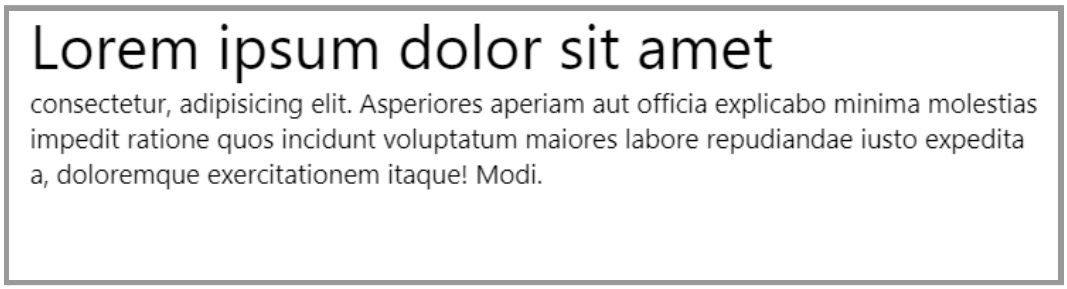
::first-line

This pseudo-element can be used to style the first line of an element.

For eg.,

```
<html>
<head>
   <style>
       p::first-line {
       font-size: 30px;
    }
   </style>
</head>
<body>
<p>
    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Asperiores aperiam aut officia explicabo minima molestias impedit ratione quos incidunt voluptatum maiores labore repudiandae iusto expedita a, doloremque exercitationem itaque! Modi.
</p>
</body>
</html>
```

Output:



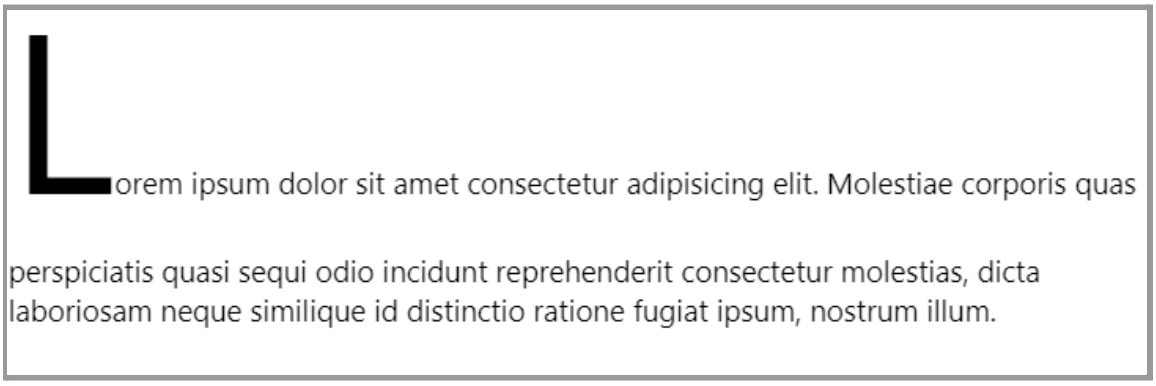
::first-letter

This pseudo-element can be used to add styling to the first letter of the text.

For eg.,

```
<html>
<head>
    <style>
        p::first-letter {
            font-size: 100px;
        }
    </style>
</head>
<body>
    <p>
Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae
corporis quas perspiciatis quasi sequi odio incidunt reprehenderit
consectetur molestias, dicta laboriosam neque similique id distinctio
ratione fugiat ipsum, nostrum illum.
    </p>
</body>
</html>
```

Output:



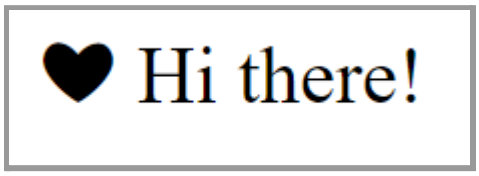
::before

You can use **::before** pseudo-element add content before an element's content. The content you want to add before the element will be **specified as a value to the content property.**

For eg.,

```
<html>
<head>
    <style>
        p::before {
            content: "❤";
            ;
        }
    </style>
</head>
<body>
    <p>
        Hi there!
    </p>
</body>
</html>
```

Output:



You can also add images like this:

```
<html>
<head>
    <style>
        h1::before {
            content:
url(https://dc-cdn.s3-ap-southeast-1.amazonaws.com/dc-Cover-67f1l9a958qafg
rmudoqel15k0-20160224121040.original.jpeg);
        }
    </style>
</head>
<body>
    <h1>My beautiful Cat</h1>
</body>
</html>
```

Output:

::after

Contrary to the ::before pseudo-element, the ::after pseudo-element is used to add content after an element's content.

For eg.,

```
<html>
<head>
    <style>
       h1::after {
       content:   url(https://i.ytimg.com/vi/3uazpaoz7tk/hqdefault.jpg);
      }
    </style>
</head>
<body>
    <h1>
        ♥♥ My cutipies ♥♥
    </h1>
</body>
</html>
```

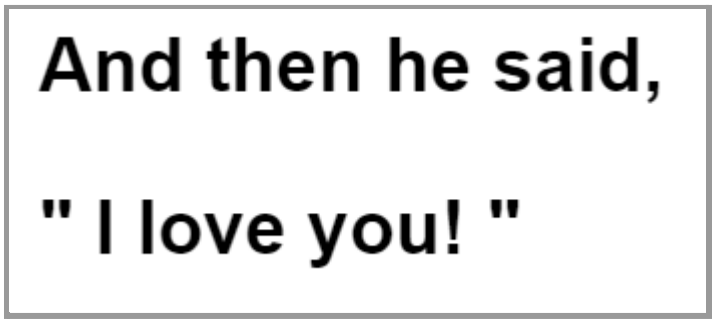
Output:

You can use the **::before** and **::after** pseudo-elements ***together to add content both before and after the element's content.***

For example, you can add quotation marks in a sentence using both **::before** and **::after** pseudo-elements:

```
<html>
<head>
    <style>
        p::before {
            content: ' " ';
        }

        p::after {
            content: ' " ';
        }
    </style>
</head>
<body>
    <h1>
        <span>And then he said,</span>
        <p>I love you!</p>
    </h1>
</body>
</html>
```

Output:



Extra:

To know more about pseudo-elements, follow this link:

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements