Coding Bootcamp

CSS

Selectors

Learning Objectives

By the end of this lesson, you will be able to:

- Apply the CSS syntax to write stylesheets for styling web pages
- Identify the selectors and tags to apply styles effectively for enhancing the webpage layout
- Differentiate between the types of selectors based on their usage for optimal CSS application
- Utilize CSS pseudo-classes and pseudo-elements for advanced styling to create dynamic effects

Learning Objectives

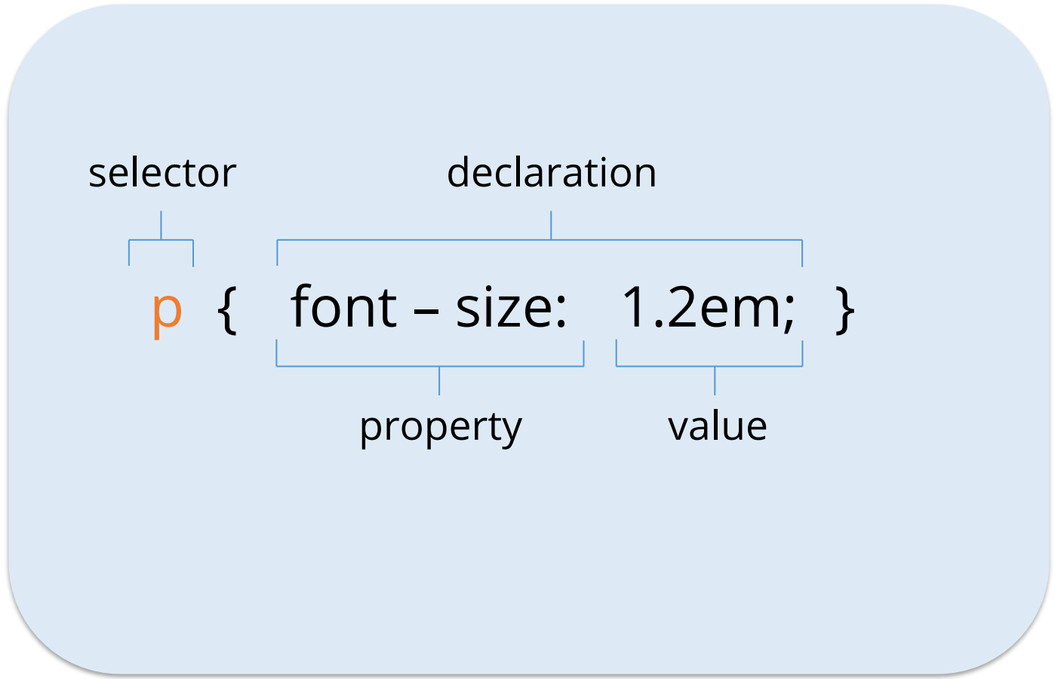
By the end of this lesson, you will be able to:

- Incorporate the concept of animation within CSS for adding movement to elements
- List the types of animations along with their syntaxes for implementing various effects
- Implement CSS media queries and classify their types for achieving responsive design

CSS Syntax

CSS Syntax

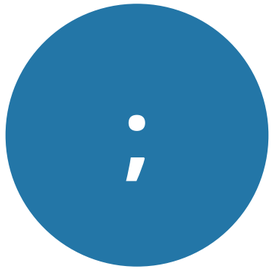
CSS controls the presentation and layout of web pages. The stylistic rules defined in a CSS file are applied to HTML elements to enhance their appearance.



CSS Syntax

A selector has to identify the HTML element to be styled.

Declarations are separated by a semicolon.



{ Name: Value ; Name: Value }

Separates name and value

Separates each attribute statement

CSS Syntax

The CSS syntax is written as:

```
selector {
property: value;
}
```

CSS Syntax: Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
color: green;
text-align: left;
font-size: 20px;
}
</style>
</head>
<body>
<p>Thanks for reaching us! </p>
<p>These are basic syntax of CSS</p>
<p>This is a paragraph</p>
</body>
</html>
```

CSS selector

Selectors

Selectors

The selector instructs the browser on which HTML elements to apply the CSS property values.

A selector is an HTML tag that tells the browser where to apply the style.

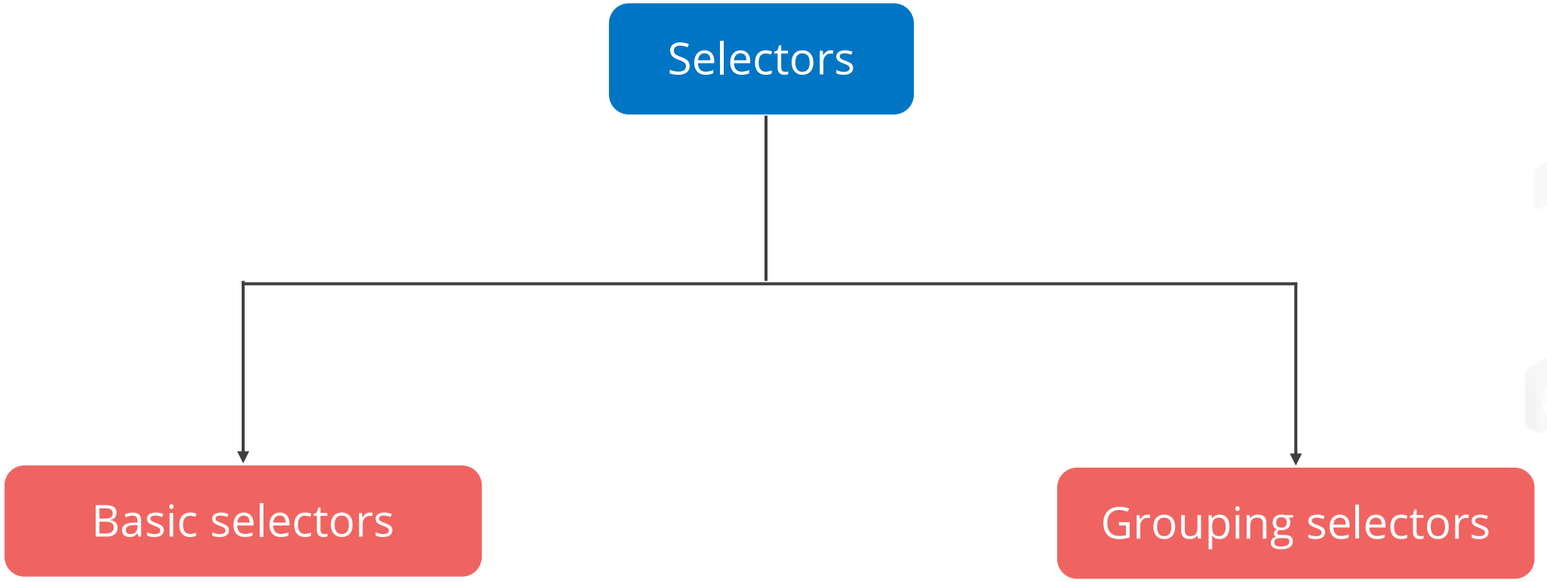
Selectors

The CSS syntax is written as:

```
selector {
property: value
}
```

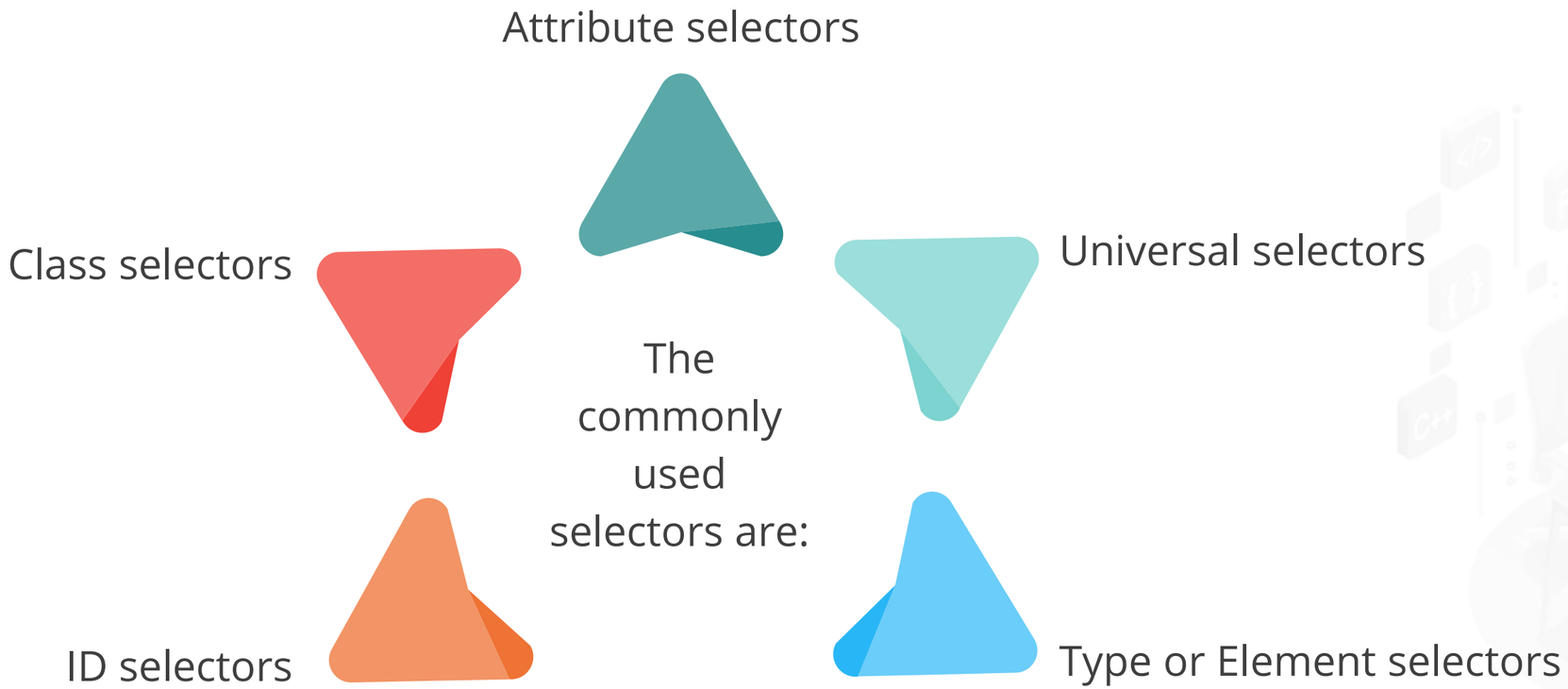
Selectors: Classification

Selectors are classified into two categories.



Basic Selectors

Basic Selectors



Universal Selectors

They match the name of any element type and apply to all the document's elements.
The syntax can be written as:

```
* {
  property: value;
}

//Target all the elements in the document.
```

Type or Element Selectors

They only match the element name in the document and select all the elements with the same name as the provided element. The syntax can be written as:

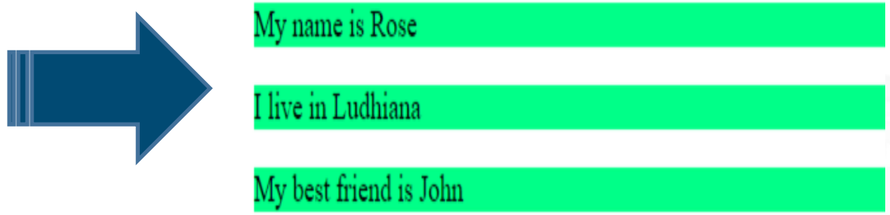
```
element {
  property: value;
}

//Targets all the h1 and h2 in the document.
```

Type or Element Selectors: Example

CSS code:

```
<!DOCTYPE html>
<html>
<head>
<style>
P {
Background-color: rgb (0,255,136);
}
</style>
<title>Element Selector Example</title>
</head>
<body>
<h1>Element Selector</h1>
<div>
<p id=”name”>My name is Rose</p>
<p id=” city”>I live in Ludhiana</p>
</div>
<p>My best friend is John</p>
</body>
</html>
```

Output:



ID Selectors

They apply styles to a single, unique element in an HTML document. Each element can have only one ID. The syntax is as follows:

```
#id_value {
  /* style properties */
}

//Targets unique element that matches the
id_value
```

ID Selectors: Example

```
#Custom {
color: #deb887;
font-size: 20px;
text-align left;
}
```

ID Selectors: Example

HTML file:

```
<!DOCTYPE html>
<html>
<head>
<title>ID Selectors</title>
<link rel=" stylesheet" type="text/css" href="
style.css"/>
</head>
<body>
<p id=" color">This is colorful</p>
<p>This is colorless</p>
</body>
</html>
```

Output:

This is colorful

This is colorless

Class Selectors

They apply styles to elements that have a specific class attribute. Class selectors allow you to apply the same styles to multiple elements by assigning them the same class attribute. The syntax is:

```
.class_value {
/* style properties*/
}
Or
t1{
}
Or
p.t1{
}
```

Class Selectors: Example

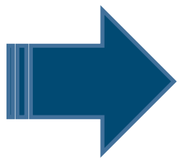
Style CSS file:

```
. custom {
color: #75a6e7;
font-size: 20px;
text-align: left;
}
```

HTML file:

```
<!DOCTYPE html>
<html>
<head>
<title>Class Selectors</title>
<link rel=" stylesheet" type="text/css" href="
style.css"/>
</head>
<body>
<p class=" custom">This is customized</p>
<p>This is not customized</p>
</body>
</html>
```

Output:



This is customized

This is not customized

Attribute Selectors

They select elements based on the presence or value of a given attribute. This allows for more specific styling of elements that share the same tag but have different attributes. The syntax is:

```
Input [type=”text”]
{
color: black;
}
```

Grouping Selectors

Grouping Selectors

They apply the same set of styles to multiple elements simultaneously. The syntax is as follows:

```
element1, element2 {
/*declarations*/
}
```


The syntax for grouping selectors involves listing multiple selectors separated by commas, followed by a declaration block.

Grouping Selectors: Example

CSS code:

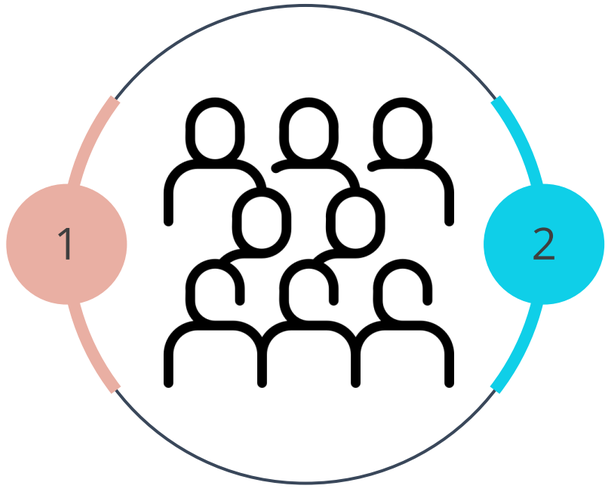
```
<!DOCTYPE html>
<html>
<head>
  <style>
    p, h1 {
      text-align: center;
      background-color: black;
      color: orange;
      width: 50%;
    }
  </style>
  <title>Grouping Selectors</title>
</head>
<body>
  <p>This is a paragraph</p>
  <h1>This is the heading</h1>
</body>
</html>
```

Output:

Types of Grouping Selectors

Grouping selectors are classified as:

Descendant selectors:
Targets all descendants of a specified element, regardless of their nesting level



Child selectors:
Targets only direct children of a specified element

Descendant Selectors

They match an element that is a descendant of another element. The syntax and example are as follows:

Syntax

```
ancestor descendant {
  /* style properties */
}
```

Example

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

Child Selectors

They match an element that are the children of a specified element. The syntax and example are as follows:

Syntax

```
parent > child {
  /* style properties */
}
```

Example

```
li > a {
  color: blue;
}

Or

body > p {
  color: black;
}
```

Adding ID, Class, and Descendant Selectors

Problem Statement:

You have been asked to develop the HTML file with id, class, and descendant selectors.

Outcome:

By completing this task, you will be able to create web pages using ID, class, and descendant selectors in HTML and CSS.

Note: Refer to the demo document for detailed steps:
03_Adding_Id_Class_Descendant_Selector

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Create the index.html and styles.css files
2. Add id, class, and descendant selectors to the HTML file
3. View the results in your local browser

Grouping Selectors

Problem Statement:

You have been asked to develop an HTML file with grouped selectors.

Outcome:

By completing this task, you will understand how to apply the same styles to multiple elements efficiently using grouped selectors.

Note: Refer to the demo document for detailed steps:
04_Grouping_Selector

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Create the index.html and styles.css files
2. Add grouped selectors with styles.css
3. View the results in your local browser

TECHNOLOGY

CSS Pseudo Classes

simplilearn

Pseudo Classes

They are keywords that define a special state for an element. The syntax is as follows:

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

CSS classes are used with pseudo-classes.

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

Link

It assists in including a special style to the unvisited link.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>Link Pseudo-Class Example</title>
  <style>
    a:link {
      color: blue;
    }
  </style>
</head>
<body>
  <h3>Link Pseudo-Class Example</h3>
  <a href="">Unvisited Link</a>
</body>
</html>
```

Visited

It assists in adding a special style to the visited link.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>Visited Pseudo-Class Example</title>
  <style>
    a:visited {
      color: green;
    }
  </style>
</head>
<body>
  <h3>Visited Pseudo-Class Example</h3>
  <a href="https://www.example.com">Click the
link</a>
</body>
</html>
```

Hover

It adds a special style to an element when one moves the cursor over it.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>Hover Pseudo-Class Example</title>
  <style>
    a:hover {
      color: orange;
    }
  </style>
</head>
<body>
  <h3>Hover Pseudo-Class Example</h3>
  <a href="#">Hover here</a>
</body>
</html>
```

Active Class

It helps to add a special style to the only active element.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>Active Pseudo-Class Example</title>
  <style>
    a:active {
      color: pink;
    }
  </style>
</head>
<body>
  <h3>Active Pseudo-Class Example</h3>
  <a href="#">Click Here</a>
</body>
</html>
```

Focus Class

It helps to add a special style to an element which has focus on it.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>Focus Pseudo-Class Example</title>
  <style>
    a:focus {
      color: purple;
    }
  </style>
</head>
<body>
  <h3>Focus Pseudo-Class Example</h3>
  <a href="#">Click Here</a>
</body>
</html>
```

First Child

It helps to add a special style to an element that is the first child of another element.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>First Child Pseudo-Class Example</title>
  <style>
    div > p:first-child {
      text-indent: 20px;
    }
  </style>
</head>
<body>
  <div>
    <p>The first paragraph in the div. It will be indented.</p>
    <p>The second paragraph is not the first child in the div. So, it will not be
indented.</p>
  </div>
  <p>This paragraph is outside the div and will not be affected.</p>
  <div>
    <h3>First child example</h3>
    <p>This paragraph inside the div tag will not be affected.</p>
  </div>
</body>
</html>
```

Lang

It adds a special style to the elements.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>Lang Pseudo-Class Example</title>
  <style>
    :lang(en) {
      quotes: '‘' '’' '‘' '’';
    }

    :lang(fr) {
      quotes: '«' '»' '‹' '›';
    }
  </style>
</head>
<body>
  <h3>Lang Pseudo-Class Example</h3>
  <p>Here is a quote in English: <q lang="en">This is an English quote.</q></p>
  <p>Here is a quote in French: <q lang="fr">Ceci est une citation
française.</q></p>
</body>
</html>
```

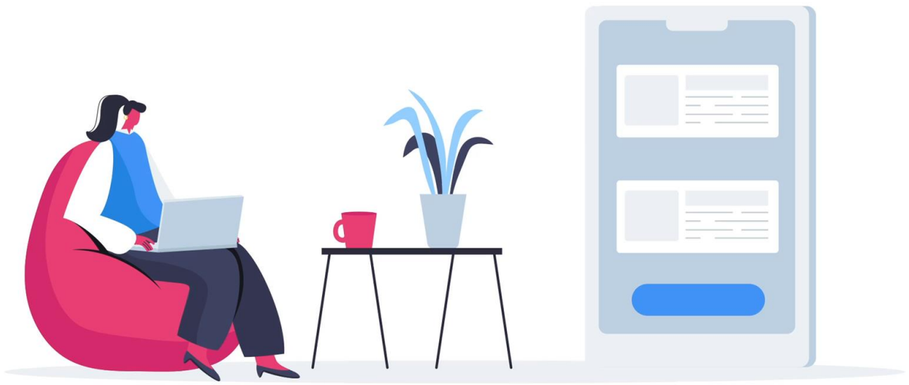
TECHNOLOGY

CSS Pseudo Elements

simplilearn

CSS Pseudo Elements

These are employed to apply special effects to a certain selectors.



The syntax is:

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

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

First Line

It adds special styles to the first line of the text in a selector.

Example: CSS code

```
<html>
   <head>
      <style type = "text/css">
         p:first-line { text-decoration: underline; }
         p.noline:first-line { text-decoration: none; }
      </style>
   </head>
   <body>
      <p class = "noline">
         Example for Firstline element
      </p>

      <p>
This example shows how to use: first-line pseudo element to give effects to
the first line of any HTML element.
      </p>
   </body>
</html>
```

First Letter

It adds a unique style to the first letter of the text in a selector.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>First Letter Pseudo-Element Example</title>
  <style>
    p:first-letter {
      font-size: 5em;
    }

    p.normal:first-letter {
      font-size: 10px;
    }
  </style>
</head>
<body>
  <p class="normal">
    The first character of this statement is normal with a font size of 10 px;
  </p>
  <p>
    This example shows how to use the ::first-letter pseudo-element to design the first
character of an HTML element.
  </p>
</body>
</html>
```

Before

It inserts content before an element.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>Before Pseudo-Element Example</title>
  <style>
    p::before {
      content: url(/images/bullet.gif);
    }
  </style>
</head>
<body>
  <p>This can be preceded by a bullet.</p>
  <p>This can be preceded by a bullet.</p>
  <p>This can be preceded by a bullet.</p>
</body>
</html>
```

After

It inserts content after an element.

Example: CSS code

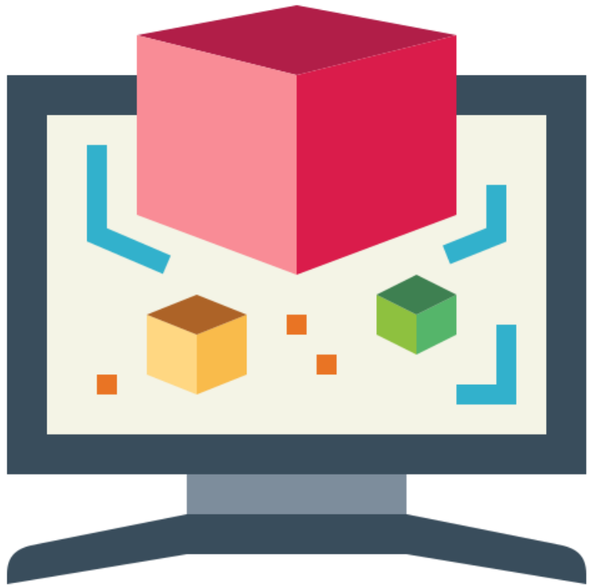
```
<!DOCTYPE html>
<html>
<head>
  <title>After Pseudo-Element Example</title>
  <style>
    p::after {
      content: url(/images/bullet.gif);
    }
  </style>
</head>
<body>
  <p>This line can be succeeded by a bullet.</p>
  <p>This line can be succeeded by a bullet.</p>
  <p>This line can be succeeded by a bullet.</p>
</body>
</html>
```

TECHNOLOGY

DIV and SPAN Elements

CSS Animations

The animation property of CSS can be used with many other CSS properties, including background color, color, height, width, images, videos, and audio.

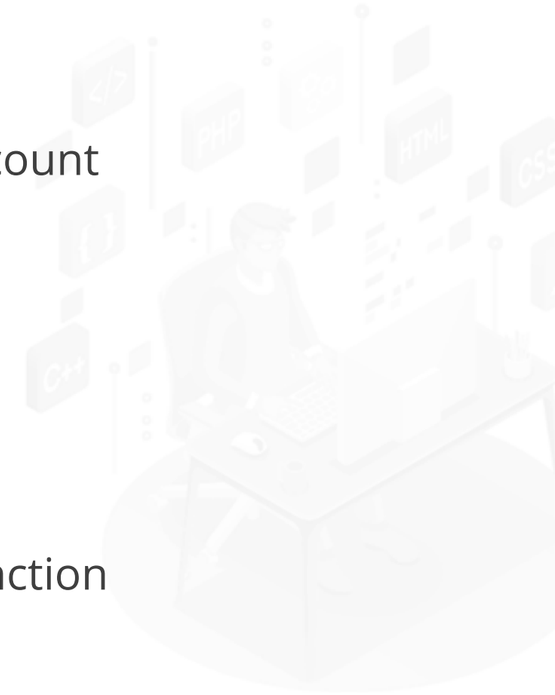


The keyframe stores the element's style or animation at specific points in time.

CSS Animations

With CSS, one can animate HTML elements without any JavaScript. Some of the animation properties are:

1. @keyframes
2. animation-name
3. animation-duration
4. animation-delay
5. animation-iteration-count
6. animation-direction
7. animation-timing-function
8. animation-fill-mode



@keyframes

The @keyframes rule in CSS defines the keyframes of an animation. Keyframes specify the styles that an element should have at specific moments during the animation.

Example

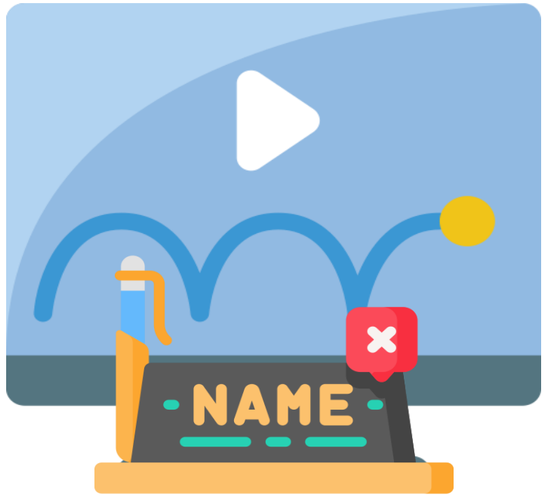
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
width: 200px;
height: 200px;
background-color: orange;
animation-name: anim;
animation-duration: 5s;
}
@keyframes anim {
from {background-color: green;}
to {background-color: purple;}
}
```

Example

```
</style>
</head>
<body>
<h2>CSS Animation</h2>
<div></div>
<p>Adding keyframes are must for the
CSS animation. The keyframe decides the
style or animation of the element at
certain times. </p>
</body>
</html>
```

animation-name

It specifies the name of the **@keyframes** animation that should be applied to an element.



It links the defined keyframes to an element, specifying which animation sequence should be applied.

animation-duration

It specifies how long the animation should take to complete.

If there is no animation-duration property specified, no animation will occur without applying the duration property.

By default, its value is zero seconds.

animation-delay

It specifies a delay for the start of the animation.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
  <title>Animation Properties</title>
  <style>
    div {
      width: 100px;
      height: 100px;
      background-color: red;
      position: relative;
      animation-name: example;
      animation-duration: 5s;
      animation-delay: 2s;
    }

    @keyframes example {
      0% { background-color: red; left: 0px; top: 0px; }
      25% { background-color: blue; left: 100px; top: 0px; }
      50% { background-color: orange; left: 100px; top: 100px; }
      75% { background-color: green; left: 0px; top: 100px; }
      100% { background-color:
grey; left: 0px; top:
0px; }
    }
</style>
</head>
<body>
  <h3>Animation
Properties</h3>
  <p></p>
  <div></div>
</body>
</html>
```

animation-iteration-count

It specifies the number of times the animation should run.

Example: CSS code

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
 width: 100px;
  height: 100px;
 background-color: red;
position: relative;
animation-name: test;
animation-duration: 5s;
animation-iteration-count: 3;
}
@Keyframes example {
  0% {background-color: red; left:0px; top:0px;}
25% {background-color: blue; left:100px; top:0px;}
50% {background-color: orange; left:100px; top:100px;}
75% (background-color: green; left:0px; top:100px;}
100% {background-color: grey; left:0px; top:0px;}
}
```

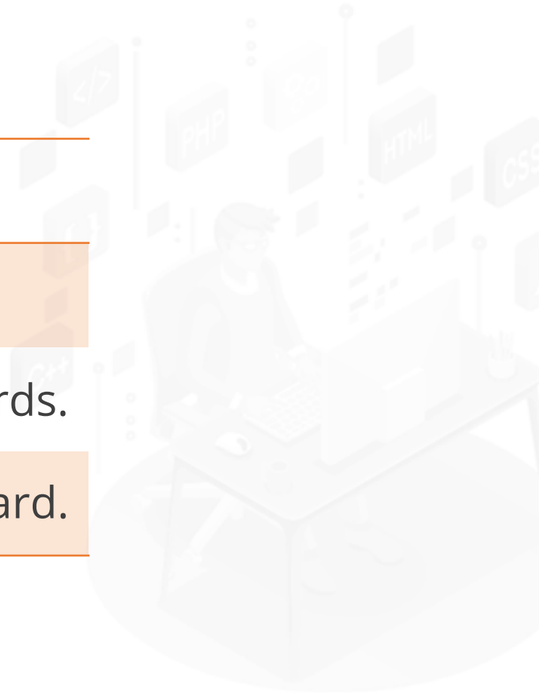
```
</style>
</head>
<body>
<h3>Animation iteration
count</h3>
<p>
 </p>
<div></div>
</body>
</html>
```

animation-direction

It specifies whether an animation should play forward, backward, or in alternate cycles.

It can have the following values:

normal	Animation plays by default.
reverse	Animation plays in the reverse direction.
alternate	Animation plays forward first and then backwards.
alternate-reverse	Animation plays backwards first and then forward.



animation-direction: Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: green;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
  animation-direction: reverse;
}

@keyframes example {
  0% { background-color: red; left: 0px; top: 0px; }
  25% { background-color: blue; left: 100px; top: 0px; }
  50% { background-color: orange; left: 100px; top: 100px; }
  75% { background-color: green; left: 0px; top: 100px; }
  100% { background-color: grey; left: 0px; top: 0px; }
}
```

```
</style>
</head>
<body>
<h3>Animation Direction</h3>
<p></p>
<div></div>
</body>
</html>
```

animation-timing-function

It specifies how an animation progresses through the duration of each cycle.

It can have the following values:

ease	Specifies the flow of the animation
linear	Specifies the animation with the same speed
ease-in	Specifies the animation with a slow start
ease-out	Specifies the animation with the slow end
ease-in-out	Specifies the animation with the slow start and slow end
cubic-Bezier (n, n, n, n)	Defines values in the cubic-bezier function

animation-timing-function: Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 400px;
  height: 200px;
  background-color: green;
  font-weight: bold;
  position: relative;
  animation: anim 5s infinite;
}
#div1{animation-timing-function: ease;}
#div2{animation-timing-function: linear;}
#div3{animation-timing-function: ease-in;}
#div4{animation-timing-function: ease-out;}
#div5{animation-timing-function: ease-in-out;}
@keyframes anim {
  from { left: 0px; }
  to { left: 400px; }
}
```

```
</style>
</head>
<body>
<h3>Animation Timing
Function</h3>
<p>Different timing functions
applied to animations:</p>
<div id="div1">ease</div>
<div id="div2">linear</div>
<div id="div3">ease-in</div>
<div id="div4">ease-out</div>
<div id="div5">ease-in-out</div>
</body>
</html>
```

animation-fill-mode

It specifies a style for the target element when the animation stops.

It can have the following values:

none	Styles are not applied to the element before and after the execution.
forwards	The element keeps the original style values set by the last keyframes.
backwards	The element fetches the style values that are set by the first keyframes.
both	The animation follows the rules for both forward and backward.

animation-fill-mode: Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: green;
  position: relative;
  animation-name: anim;
  animation-duration: 3s;
  animation-fill-mode: forwards;
}

@keyframes anim {
  from { top: 0px; }
  to { top: 100px; background-color: red; }
}
```

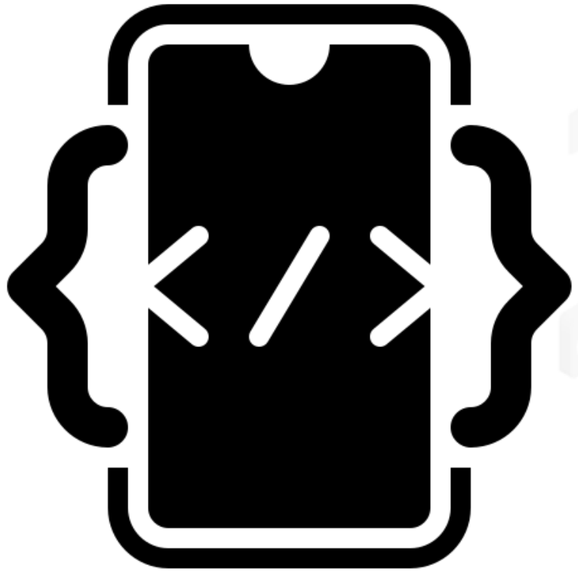
```
</style>
</head>
<body>
<h3>Animation Fill Mode</h3>
<p>This example demonstrates
the  animation-fill-mode:
forwards  property.</p>
<div></div>
</body>
</html>
```

TECHNOLOGY

CSS Media Queries

Media Queries

A media query contains media type and can contain one or more expressions, which resolve to either true or false.



They allow you to apply a special CSS for mobiles or adjust the layout for the print.

Media Queries

Media queries check:

- Height and width of the viewport
- Height and width of the device
- Resolution
- Orientation

The syntax is:

```
@Media not | only mediatype and
(expressions) {
CSS-Code.
}
```

There are different stylesheets for different media.

```
@Media not | only mediatype and (expressions) {
CSS-Code. <link rel=" stylesheet" media="
mediatype and |not|only (expressions) "href="
style.css">
}
```

Media Queries: Types

Some of the CSS media query types are:

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for desktops, laptops, tablets, mobiles, and more
speech	Used for the screen readers to read the page aloud

Media Query with Minimum Width: Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: green;
}

@media screen and (min-width: 480px) {
  body {
    background-color: red;
  }
}
</style>
</head>
<body>
<h3>Media Query Min-width</h3>
<p>This example changes the background color based on the viewport
width.</p>
</body>
</html>
```

Media Query with Maximum Width: Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: red;
}

@media screen and (max-width: 992px) {
  body {
    background-color: brown;
  }
}
</style>
</head>
<body>
<h3>Media Query Max-width</h3>
<p>This example changes the background color based on the viewport
width.</p>
</body>
</html>
```

Adding Text Styling, Borders, Background Images

Problem Statement:

You have been asked to develop the HTML file with text styling, borders, and background images.

Outcome:

By completing this task, you will improve your web design skills by applying advanced CSS techniques for a better user experience.

Note: Refer to the demo document for detailed steps:
05_text_Styling_Borders_Backgroung_Images

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. On the HTML folder, create files with .html and .css extension
2. Create and save text styling, borders, and background images
3. Use the Chrome browser to open the saved file

Styling Form

Problem Statement:

You have been asked to develop the HTML file using a styled form.

Outcome:

By completing this task, you will learn how to design user-friendly and visually appealing forms

Note: Refer to the demo document for detailed steps:
06_Styling_Form

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. On the HTML folder, create a file with a .html extension
2. Create and save the code for a styled form
3. Use the Chrome browser to open the saved file

Adding Links, Blocks, and Float

Problem Statement:

You have been asked to develop an HTML file with links, blocks, and float.

Outcome:

By completing this task, you will understand how to effectively create navigable links and structure web page layouts using block elements and the float property.

Note: Refer to the demo document for detailed steps:
07_Links_Blocks_Floats

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. In the HTML folder, create a file with a .html and .css extension
2. Create and save code for links, blocks, and float
3. Use the Chrome browser to open the saved files

Key Takeaways

- The Selector instructs the browser on which HTML elements to apply the CSS property values.
- Universal Selectors match the name of any element type and apply to all the document's elements.
- Element selectors select all elements with the same name as the provided element.
- With Group Selectors, multiple items can be selected and styled at once.
- With keyframes, the animation transitions to the newest style from the current style.
- A media query contains media type and one or more expressions that resolve to either true or false.

TECHNOLOGY

Thank You

simplilearn