Front-end | CSS

Session 1

- Introduction
- Selectors
- How to
- Some Properties
- Advanced Selectors

by Mohammad Amin H.B. Tehrani - Reza Yazdani

www.maktabsharif.ir

Introduction

Intro

What is CSS?

CSS is the language we use to style a Web page.

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

```
<head>
   <meta charset="UTF-8">
   <title>Title</title>
   <style>
     body {
       background-color: rgb(150,150,250);
     }

     h1{
       color: #000086;
     }

     p {
       color: rebeccapurple;
     }
   </style>
</head>
<body>
```

Syntax

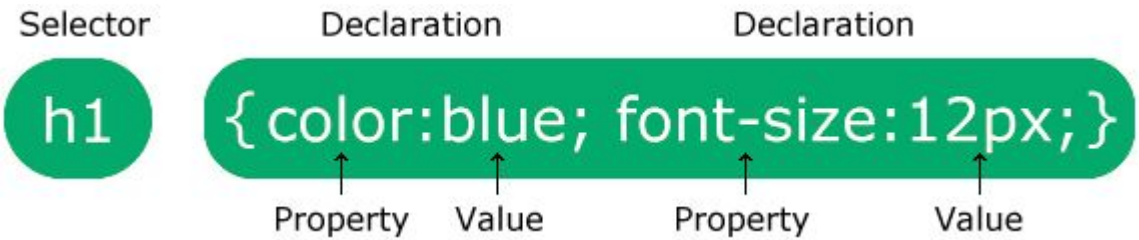
A CSS rule consists of a selector and a declaration block.

The **selector** points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.



Example

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>HTML/CSS Test</title>
   <style>
     body {background-color: rgb(150,150,250);}
     h1 {color: #000086;}
     p {color: rebeccapurple;}
   </style>
</head>
<body>
<h1>Hello world</h1>
<p>It's a test html page...</p>
</body>
</html>
```

Selectors

Intro

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- **Simple selectors (select elements based on name, id, class)**
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

Element Selector

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

tag_name { declaration }

```
p {
 font-size: 40px;
 color: darkgreen;
}

img {
 width: 100px;
 height: 150px;
}
```

```
<p>It's a test html page... </p>
<p>another paragraph</p>
<img src="/akbar.jpg">
```

It's a test html page...

another paragraph

id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

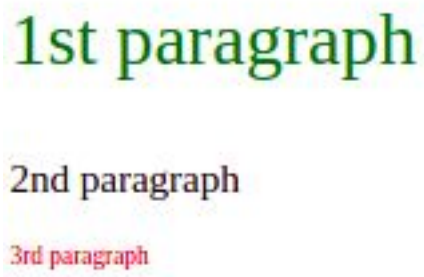
The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

#id { declarations; }

```
#p1 {
 font-size: 30px;
 color: darkgreen;
}
#p3 {
   font-size: 10px;
   color: red;
}
```

```
<p id="p1">1st paragraph</p>
<p>2nd paragraph</p>
<p id="p3">3rd paragraph</p>
```



Class selector

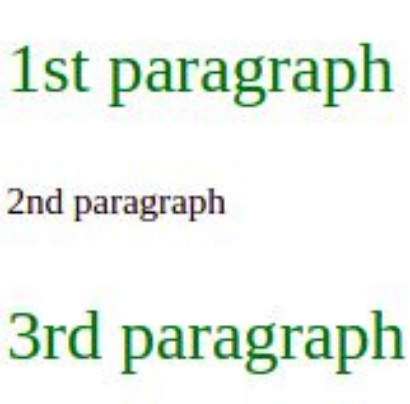
The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a **period (.)** character, followed by the class name.

.class { declarations; }

```
.my-class {
 font-size: 30px;
 color: darkgreen;
}
```

```
<p class="my-class">1st paragraph</p>
<p>2nd paragraph</p>
<p class="my-class">3rd paragraph</p>
```



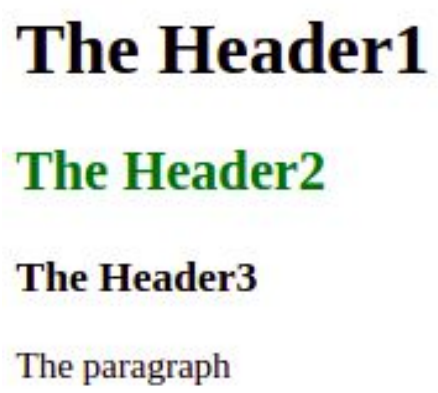
Element.Class selector

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

tag_name.class { declarations; }

```
h2.my-class {
    color: darkgreen;
}
```

```
<h1 class="my-class">The Header1</h1>
<h2 class="my-class">The Header2</h2>
<h3 class="my-class">The Header3</h3>
<p class="my-class">The paragraph</p>
```



Universal selector (*)

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

* { declarations; }

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

```
<h1>The Header</h1>
<p>The paragraph</p>
<a href="#">The link</a>
```

Grouping Selector

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

selector1, selector2, selector3, ... { declarations; }

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

h2 {
    text-align: center;
    color: red;
}

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

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

How to

Intro

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

How to

External css

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <link rel="stylesheet" href="mystyle.css">
   <title>Test html</title>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```
/* mystyle.css */
* {
   text-align: center;
   color: blue;
}
```

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

How to

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

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

       h1 {
           color: maroon;
           margin-left: 40px;
       }
   </style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

How to

InLine CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

How to

Cascading?

The name ***cascading*** comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

```css
/* cards.css */
.card-container {
   display: flex;
   flex-direction: row;
}
.card {
   background-color: white;
   box-shadow: 0 3px 5px 2px gray ;
   border-radius: 10px;
}

/* Update .card style */
.card {
   width: 150px;
   height: 250px;
   margin: 5px;
}

#primary_card {
   background-color: #0d0d5c;
}
```

```html
<head>
   <link rel="stylesheet" href="cards.css">
   <style>
      .card {
          text-align: center;
          padding: 10px;
      }

      #primary card {
          color: white;
      }

      p{
          opacity: 0.8;
          font-size: 12px;
      }
   </style>
</head>
<body style="background-color: rgba(64,86,141,0.57)">
```

1

2

3

How to

Cascading example

```css
#test-id {
   color: green;
}
.class-1 {
   color: red;
}
.class-2 {
   color: blue;
}
```

```html
<p class="class-1">Hello 1</p>
<p class="class-2">Hello 2</p>
<p class="class-1 class-2">Hello 3</p>
<p class="class-2 class-1">Hello 4</p>
<p id="test-id" class="class-2 class-1">Hello 5</p>
<p id="test-id" class="class-2 class-1">Hello 6</p>
<p id="test-id" class="class-1" style="color: black">Hello 6</p>
```

Hello 1

Hello 2

Hello 3

Hello 4

Hello 5

Hello 6

Hello 7

How to

!important rule

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!

```css
#test-id {
    color: green;
}

.class-1 {
    color: red!important;
}

.class-2 {
    color: blue;
}
```

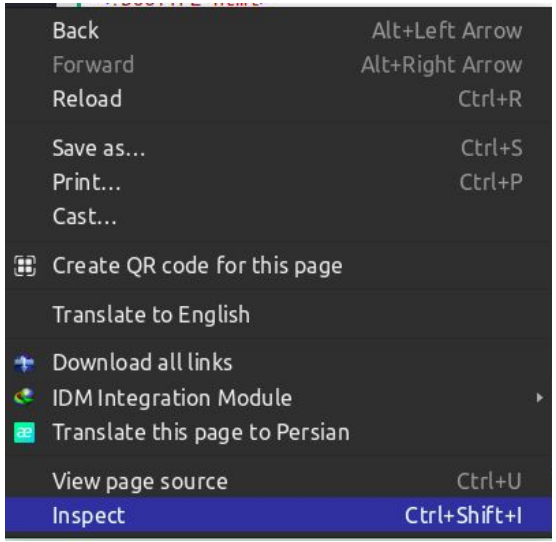
```html
<p id="test-id" class="class-1 class-2">Hello!</p>
```

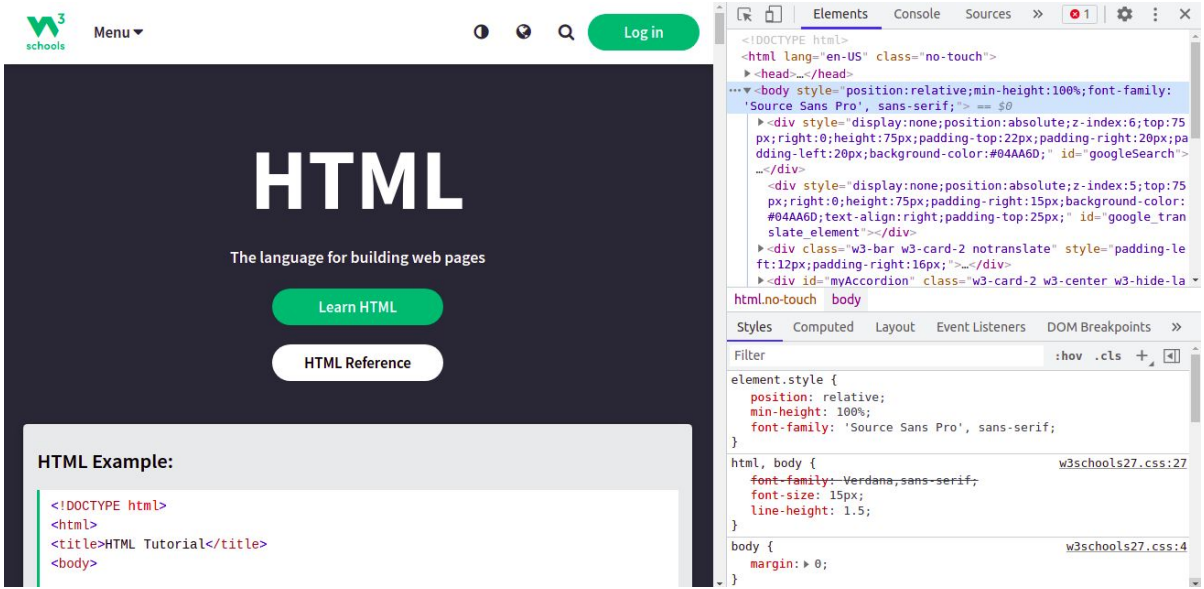
Hello!

How to

Browser inspector

Every modern web browser includes a powerful suite of developer tools. These tools do a range of things, from inspecting currently-loaded HTML, CSS and JavaScript to showing which assets the page has requested and how long they took to load. This article explains how to use the basic functions of your browser's devtools.





CSS properties

CSS properties

background-color

The background-color property specifies the background color of an element.

Example:

```
<div style="background-color: red">
   <h1>A simple heading</h1>
   <p>A simple Paragraph</p>
</div>
```

CSS properties

text-align

The text-align property specifies the horizontal alignment of text in an element.

Syntax:

```
text-align: left|right|center|justify|initial|inherit;
```

Example:

```
<div style="text-align: center">
   <h1>A simple heading</h1>
   <p>A simple Paragraph</p>
</div>
```

color

Text Color

The color property is used to set the color of the text. The color is specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Example:

```
<div>
   <h1 style="color: blue">A simple heading</h1>
   <p style="color: red">A simple Paragraph</p>
</div>
```

CSS properties

font-size

The font-size property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.

Example:

```
<div style="font-size: 40px">
   <h1>A simple heading</h1>
   <p>A simple Paragraph</p>
</div>
```

CSS properties

border

The **border** properties allow you to specify the style, width, and color of an element's border.

Example:

```
<div style="border: 4px solid red">
   <h1>A simple heading</h1>
   <p>A simple Paragraph</p>
</div>
```

Format:

```
border: {WIDTH} {STYLE} {COLOR};
```

CSS properties

margin

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Individual properties:

- **margin-top**: *{value}*
- **margin-right**: *{value}*
- **margin-bottom**: *{value}*
- **margin-left**: *{value}*

```
<div style="background-color: red;margin: 50px">
   Test!
</div>
```

Shorthand:

- margin: *{value}*
- margin: *{value-vertical}* *{value-horizontal}*
- margin: *{value-top}* *{value-right}* *{value-bottom}* *{value-left}*

CSS properties

padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Individual properties:

- **padding-top**: *{value}*
- **padding-right**: *{value}*
- **padding-bottom**: *{value}*
- **padding-left**: *{value}*

```
<div style="background-color: red;padding: 50px">
   Test!
</div>
```

Shorthand:

- padding: *{value}*
- padding: *{value-vertical}* *{value-horizontal}*
- padding: *{value-top}* *{value-right}* *{value-bottom}* *{value-left}*

CSS properties

float

The float property specifies how an element should float.

Note: Absolutely positioned elements ignore the float property!

```
<img src="session26/flower.jpg" width="200" style="float: right;">
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Doloribus, nisi nulla! A aliquam architecto consequuntur, eaque et nam nostrum officia, perspiciatis qui quisquam reprehenderit rerum sapiente. Maxime minus optio tempora.</p>
```

- **left** - floats to the left of its container
- **right** - floats to the right of its container
- **none** - does not float (will be displayed just where it occurs in the text). This is default
- **inherit** - inherits the float value of its parent

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Doloribus, nisi nulla! A aliquam architecto consequuntur, eaque et nam nostrum officia, perspiciatis qui quisquam reprehenderit rerum sapiente. Maxime minus optio tempora.

CSS properties

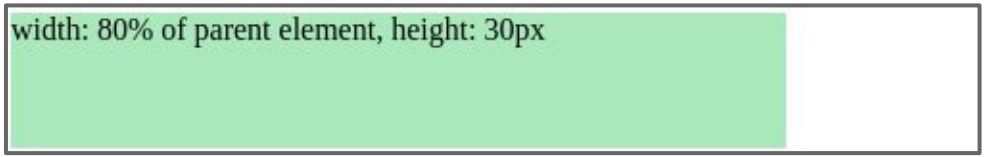
height/width

The CSS height and width properties are used to set the height and width of an element.

The height and width properties may have the following values:

- **auto** - This is default. The browser calculates the height and width
- **length** - Defines the height/width in px, cm etc.
- **%** - Defines the height/width in percent of the containing block
- **initial** - Sets the height/width to its default value
- **inherit** - The height/width will be inherited from its parent value

```
<p style="width: 40%; height: 70px; background-color : #b1dfbb">
width: 40% of parent element, height: 30px
</p>
```



Some properties

Some properties

- **display**
- min-height
- min-width
- max-height
- max-width
- clear
- float
- background-color
- background-image
- background-repeat
- background-size
- background
- overflow
- border-color
- border-left
- border-radius
- border-left-width
- box-shadow
- align-items
- align-content
- bottom
- **position**
- transform
- transition
- background: linear-gradient()
- direction
- cursor
- filter
- font-family
- object-fit
- @font-face
- clip
- opacity
- text-align

Advanced selectors

Syntax

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- **Combinator selectors (select elements based on a specific relationship between them)**
- **Pseudo-class selectors (select elements based on a certain state)**
- **Pseudo-elements selectors (select and style a part of an element)**
- **Attribute selectors (select elements based on an attribute or attribute value)**

Combinator selectors

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

There are four different combinators in CSS

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

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

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

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

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

Descendant selector

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

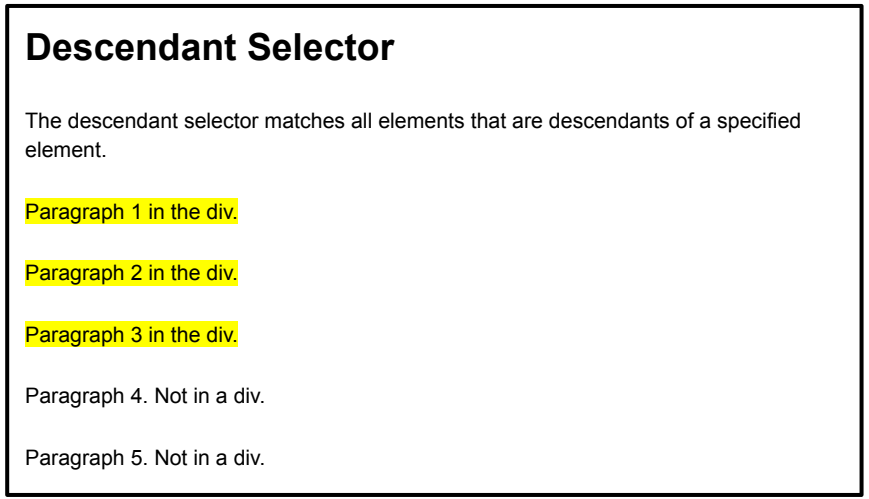
The following example selects all <p> elements inside elements:

```
<!DOCTYPE html>
<html>
<head>
   <style>
       div p {
           background-color: yellow;
       }
   </style>
</head>
<body>
<h2>Descendant Selector</h2>
<p>The descendant selector matches all elements that are descendants of a specified element.</p>
<div>
   <p>Paragraph 1 in the div.</p>
   <p>Paragraph 2 in the div.</p>
   <section><p>Paragraph 3 in the div.</p></section>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

Descendant selector

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

The following example selects all <p> elements inside elements:




```
<!DOCTYPE html>
<html>
<head>
   <style>
       div p {
           background-color: yellow;
       }
   </style>
</head>
<body>
<h2>Descendant Selector</h2>
<p>The descendant selector matches all elements that are descendants of a specified element.</p>
<div>
   <p>Paragraph 1 in the div.</p>
   <p>Paragraph 2 in the div.</p>
   <section><p>Paragraph 3 in the div.</p></section>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

Child selector

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

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

Child Selector

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

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4 in the div.

Paragraph 5. Not in a div.

Paragraph 6. Not in a div.

```
<!DOCTYPE html>
<html>
<head>
   <style>
       div > p {
           background-color: yellow;
       }
   </style>
</head>
<body>
<h2>Child Selector</h2>
<p>The child selector (>) selects all elements that are the children of a specified element</p>
<div>
   <p>Paragraph 1 in the div.</p>
   <p>Paragraph 2 in the div.</p>
   <section><p>Paragraph 3 in the div.</p></section> <!-- not Child but Descendant -->
   <p>Paragraph 4 in the div.</p>
</div>
<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>
</body>
</html>
```

Pseudo-classes selectors

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

For example, it can be used to:

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

```
/* unvisited link */
a:link {
    color: #FF0000;
}

/* visited link */
a:visited {
    color: #00FF00;
}

/* mouse over link */
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

Hover

The :hover selector is used to select elements when you mouse over them.

Tip: The :hover selector can be used on all elements, not only on links.

Tip: Use the :link selector to style links to unvisited pages, the :visited selector to style links to visited pages, and the :active selector to style the active link.

Note: :hover MUST come after :link and :visited (if they are present) in the CSS definition, in order to be effective!

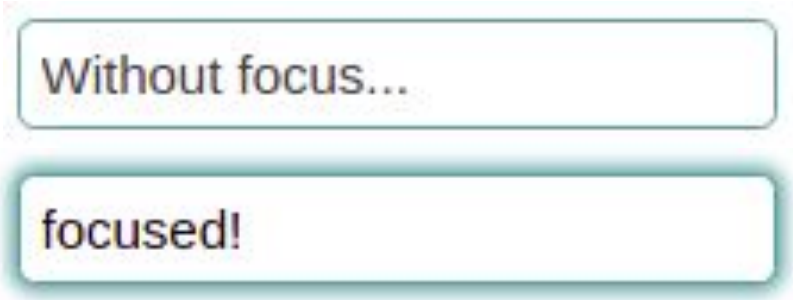

```css
button.my_btn {
    box-shadow: 0 3px 5px 0 gray;
    color: white;
    background: #3434d4;
    border: 1px solid gray;
    font-size: 1rem;
    padding: .5rem 1.5rem;
    border-radius: 20px;
     outline: none;
}

button.my_btn:hover {
    box-shadow: none;
    background: #0d0d5c;
    color: rgba(255, 255, 255, 0.66);
}
```

```html
<button class="my_btn">Test</button>
```

Focus

The :focus selector is used to select the element that has focus.

Tip: The :focus selector is allowed on elements that accept keyboard events or other user inputs.




```css
input.my_input {
   border: 1px solid cadetblue;
   padding: .4rem .4rem;
   border-radius: 5px;
   font-size: 1rem;
   outline: none;
   color: #3e3e3e;
}

input.my input:focus {
   box-shadow: 0 0 6px 2px cadetblue;
   color: #0e0e0e;
}
```

```html
<input class="my_input">
```

More pseudo-class Selectors

W3Schools: CSS 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
...		

Pseudo-Elements selectors

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

For example, it can be used to:

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

Syntax:

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

Examples:

1. **::first-letter :** Try it!
2. **::first-line :** Try it!
3. **::before :** Try it!
4. **::after :** Try it!
5. ...

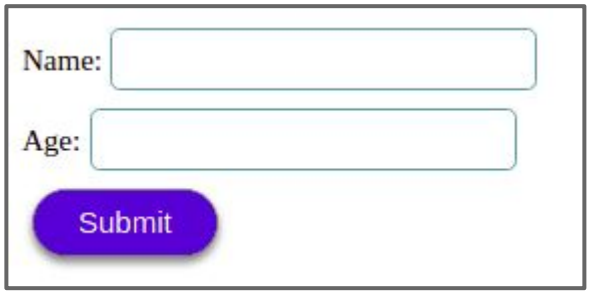
Attribute selectors

It is possible to style HTML elements that have specific attributes or attribute values.

The [attribute] selector is used to select elements with a specified attribute.

The following example selects all <a> elements with a target attribute:

W3Schools: Attribute Selectors




```css
input[type='submit'] {
    box-shadow: 0 3px 5px 0 gray;
    color: white;
    background: #3434d4;
    border: 1px solid gray;
    font-size: 1rem;
    padding: .5rem 1.5rem;
    border-radius: 20px;
    outline: none;
}

input:not([type='submit']) {
    border: 1px solid cadetblue;
    padding: .4rem .4rem;
    border-radius: 5px;
    font-size: 1rem;
    outline: none;
    color: #3e3e3e;
}
```