

One Final Look at CSS Selectors

Yeah, I know we have spent an ample lot of time on selectors, but CSS is nothing without selectors. Every declaration begins with a selector. It is worth the deep study. In this lesson, we will take a look at Pseudo-element Selectors.

Pseudo-element Selectors

Pseudo-element selectors act as if you added new HTML markup into the HTML document, and then styled that markup.

Did you get that?

Let's see some examples.

The Pseudo-element selectors are very much like the Pseudo-class selectors in terms of how they are written. Except that the CSS3 specification states that they should be written with double colons.

Remember that Pseudo-class selectors are written with a single colon like this:

```
1 li:nth-child(2n) {  
2   color: red;  
3 }
```



Officially, pseudo-elements are denoted with a double-colon e.g. `p::first-line`

For backward compatibility, and better browser support due to browser manufacturers, the use of a single colon for these pseudo-elements is encouraged.

First Letter

The `first-letter` pseudo-element selector targets the first letter of the content inside the parent element.

Let's see an example

Let's see an example.

Consider the basic markup below:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>A Simple Page</title>
<link rel="stylesheet" href="styles.css" />
</head>
<body>
  <div class="content">
    <p><span class="firstletter">H</span>ello World.</p>
  </div>
</body>
</html>
```

A ‘smart’ fellow could target the first letter **H** with a class and then style it, like so:

```
.firstletter {
  color: red;
  font-weight: bold;
  font-size: 200%;
}
```

This will yield this:

Output
HTML
CSS (SCSS)

Hello World.



On the contrary the same could have been achieved by using the `:first-letter` pseudo-element selector.

```
.content:first-letter {  
  color: red;  
  font-weight: bold;  
  font-size: 200%;  
}
```



Same results!

Note that even though the content is wrapped in a `p` tag, the `:first-letter` selector still works. This is because it targets the first letter of the **content** i.e texts.

First Line

Just like `:first-letter` but with a twist. The `:first-line` pseudo-element selector targets the first line of the content.

For example:

```
content:first-line {  
  color: red;  
  font-weight: bold;  
  font-size: 200%;  
}
```



The beauty of the `:first-line` selector is that it dynamically selects the first line even if the length of the first line changes e.g due to resizing the browser.

Before and After Pseudo-element Selectors

I may be spending a lot more time explaining these.

The `before` and `:after` pseudo-element selectors are very useful for many css tricks you'll find out there.

They are used to generate content from css, and style them too.

What is a generated content?

Generated content is content (e.g. text) that is added to your html page from CSS. Really? Yes!

How does this work?

Consider the basic markup below:

```
<div class="article">  
  <p>I love CSS. Do you?</p>  
</div>
```



With CSS we will add the text, “Yes, I do” before the paragraph with the text, *I love CSS. Do you?* above.

Here's the CSS to do so

```
article:before {
```

```
.article:before {  
  content: "Yes, I do"  
}
```

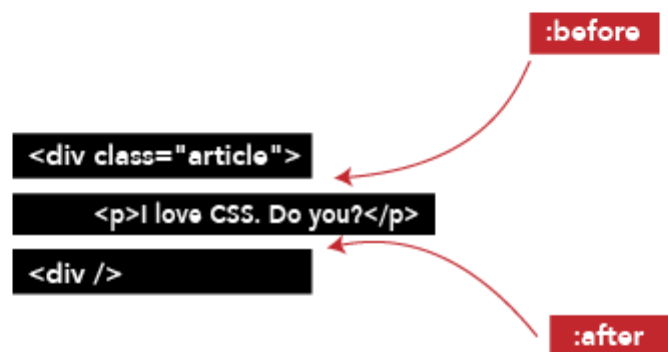
and add “No, I dont” after the paragraph, *I love CSS. Do you?*

```
.article:after {  
  content: "No, I dont"  
}
```

Note that the value of the content property must be specified, even if it is empty. i.e `content: ''`

Where are the Generated Contents Placed?

`:before` places the content **before** any child elements within the parent element, and `:after` places the content **after** every child element within the parent.



A few things to note are:

1. Content added using pseudo-elements is only visually displayed. It is not

inserted into the DOM.

2. Don't rely on `:before` and `:after` to insert content that is relevant to the meaning and completeness of the content on the page.

Don't worry. As we build stuff in the practical sections, we will see some very useful use cases for `:before` and `:after`

Attribute Selectors

Attribute selectors target an element based on an attribute. Here's a bit of a refresher on what attributes are:

```
<a href="educative.io" title="Visit Google">Google</a>
```



In the markup above, `href` and `title` are both attributes. So, if there were 2 `input` elements on a page. Like this:

```
<input type="text" name="name" id="username" />
<input type="pin" name="password" id="userpassword" />
```

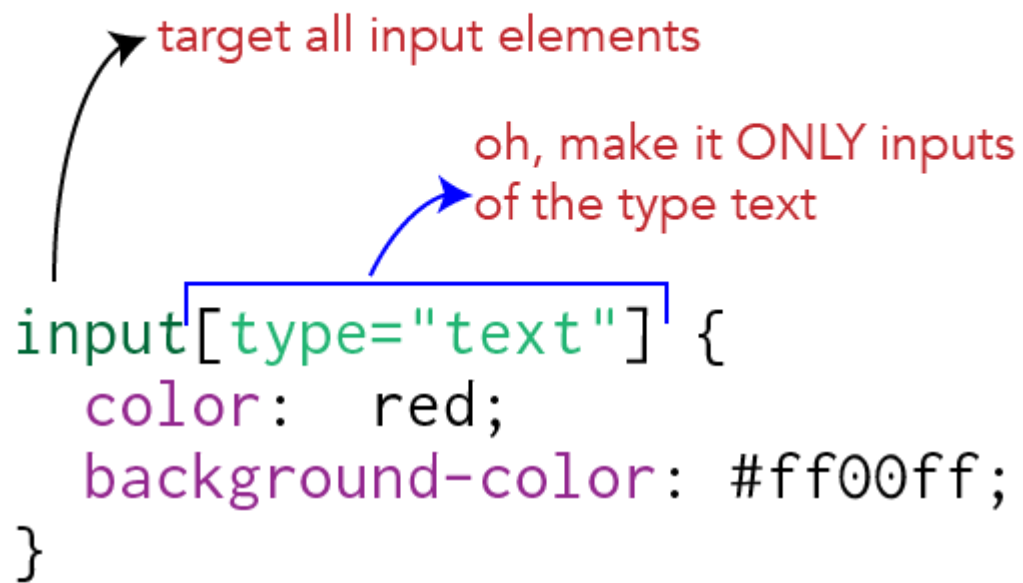


You can style the `input` with the **type attribute** differently, like so:

```
input[type="text"] {
  color: red;
  background-color: #ff00ff;
}
```



You see how that works?



target all input elements

oh, make it ONLY inputs
of the type text

```
input[type="text"] {  
  color: red;  
  background-color: #ff00ff;  
}
```

The diagram illustrates the refinement of a CSS selector. A black arrow points from the text 'target all input elements' to the 'input' part of the selector 'input[type="text"]'. A blue arrow points from the text 'oh, make it ONLY inputs of the type text' to the '[type="text"]' part of the same selector. The CSS code is displayed below, with 'input' in green, '[type="text"]' in green, 'color: red;' in purple, and 'background-color: #ff00ff;' in purple.

We will be taking a closer look at attribute selectors in the practical sections coming.

Wait for it!