Pseudo-class selectors give developers

great control over what they select and style.

By knowing how to use these selectors,

you will not only improve

the interactivity of your web pages,

but you will also be able to add

advanced styling without too much effort.

These selectors are also often

referred to as just pseudo-classes.

Previously, you were briefly

introduced to the pseudo-class invalid.

Remember, pseudo-classes are state-based selectors,

which means that they allow you to select

elements based on their state.

For example, the hover state.

You use pseudo-class selectors

to improve the interactivity of

web pages by styling elements in response to user input.

In this video, you will explore

practical examples of how to do this.

There are many types of pseudo-classes.

Other than the hover state,

examples include selecting an element

when it is active or in-focus,

or when a link has already been visited.

Pseudo-classes are also very

effective to target specific elements such as,

let's say, the fifth item in a list,

bold items, empty elements, and so on.

Let's review the general syntax

that you use for various pseudo-classes.

You add the selector, a colon,

the pseudo-class, and then the properties.

Well, there isn't a broadly

accepted classification for pseudo-classes,

you can group them in terms of

general similarities and their purpose.

Let's explore a few of these general classifications.

The first group you need to know

about is user action states,

which include the hover pseudo-class,

which changes the style of

an element when a cursor hovers over it.

The active pseudo-class,

which styles an element only while

a user actively presses and holds the mouse button,

and the focus pseudo-class,

which focuses styling on the element that you use it for.

These pseudo-classes have an effect while a user

is actively engaging with a HTML element.

Let's explore an example of how the hover

and active pseudo-classes can be used.

Say you have a HTML file with an a tag

inside a paragraph and

a button element inside a div element.

The paragraph element has a class defined as

"mypage" and the button class is defined as "mybutton."

Now, let's explore the CSS.

First it contains basically styling for

the mypage and mybutton class elements.

Then to make the link change

its appearance when a user hovers over it,

you use the hover pseudo-class to target a tag.

To change the mybutton class,

you use the active pseudo-class.

The active pseudo-class changes the properties of

an element when the state of that element is active.

In this case, the styling properties will be

applied only when a user

presses and holds the mouse button.

In this example, the pseudo-class selectors

target HTML elements,

but you can also use them to

target HTML class attributes.

Next, let's focus on Form states.

Previously, when discussing form validation you were

introduced to the invalid pseudo-class selector.

But there are more pseudo-classes that are

specifically used for HTML forms.

They usually come in pairs and target

the different states that elements can have.

Form state pseudo-classes include disabled and enabled,

generally used for buttons,

checked and indeterminate, that are used for checkboxes,

and valid and invalid used in

case of fields like emails and phone numbers.

Another pseudo-class type that you should know about

is specific position-based states.

They allow you to target specific items,

for instance, a specific list

item among the list elements.

Some examples of this type include,

first-of-type, last-of-type,

nth-of-type, and nth-last-of-type.

Let's explore an example of first-of-type.

Say you have two list items in

an HTML page, Adrian and Mario.

With the first-of-type pseudo-class,

you can style the first item of

its type while the remaining items will not be affected.

In this video, you explored

a few examples of how you can use

pseudo-classes to improve

the interactivity of your web pages.

There are plenty of

other pseudo-classes and some

of them are more popular than others.

You're encouraged to follow your own style and explore

the creative possibilities that pseudo-classes offer.

**Pseudo-elements**

By now you know that pseudo-elements help you style a specific part of an element. For example, you can decide to apply styling to only the first word or line in a given element. First, let’s examine the syntax of a pseudo-element.

**Syntax**

*selector::pseudo-element {*

*property: value;*

*}*

It is important to note that pseudo-elements use two colon characters instead of one.

Now, let’s explore some examples of popular pseudo-elements.

**::first-letter**

You can use first-letter to change the color of just the first letter of each of the three points in the example text.

HTML code:

<!DOCTYPE html>

<html>

    <head>

        <link rel="stylesheet" href="pseudo4.css">

    </head>

<body>

    <ul>

        <li>Based in Chicago, Illinois, Little Lemon is a family-owned Mediterranean restaurant, focused on traditional recipes served with a modern twist. </li>

    <li>The chefs draw inspiration from Italian, Greek, and Turkish culture and have a menu of 12–15 items that they rotate seasonally. The restaurant has a rustic and relaxed atmosphere with moderate prices, making it a popular place for a meal any time of the day.</li>

    <li>Little Lemon is owned by two Italian brothers, Mario and Adrian, who moved to the United States to pursue their shared dream of owning a restaurant. To craft the menu, Mario relies on family recipes and his experience as a chef in Italy.</li>

  </ul>

</body>

</html>

li::first-letter {

color:coral;

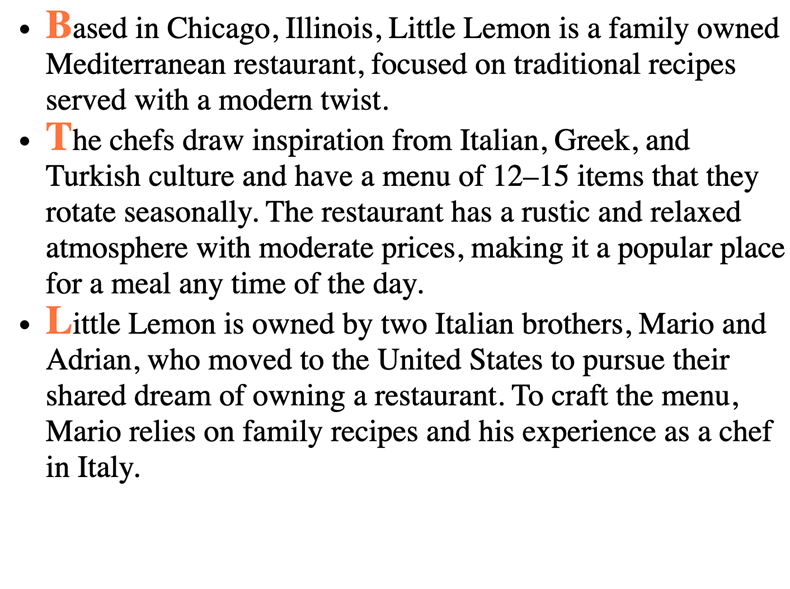
font-size: 1.3em;

font-weight: bold;

line-height: 1;

}

Output



Although the code only changed the first letter of each bullet point, it makes a big difference in terms of presentation. Now let’s change the font in a different way.

**::first-line**

First-line will change the complete first line of each of the bullet points to light sea green.

CSS code:

ul{

    list-style-type: none;

}

li::first-line {

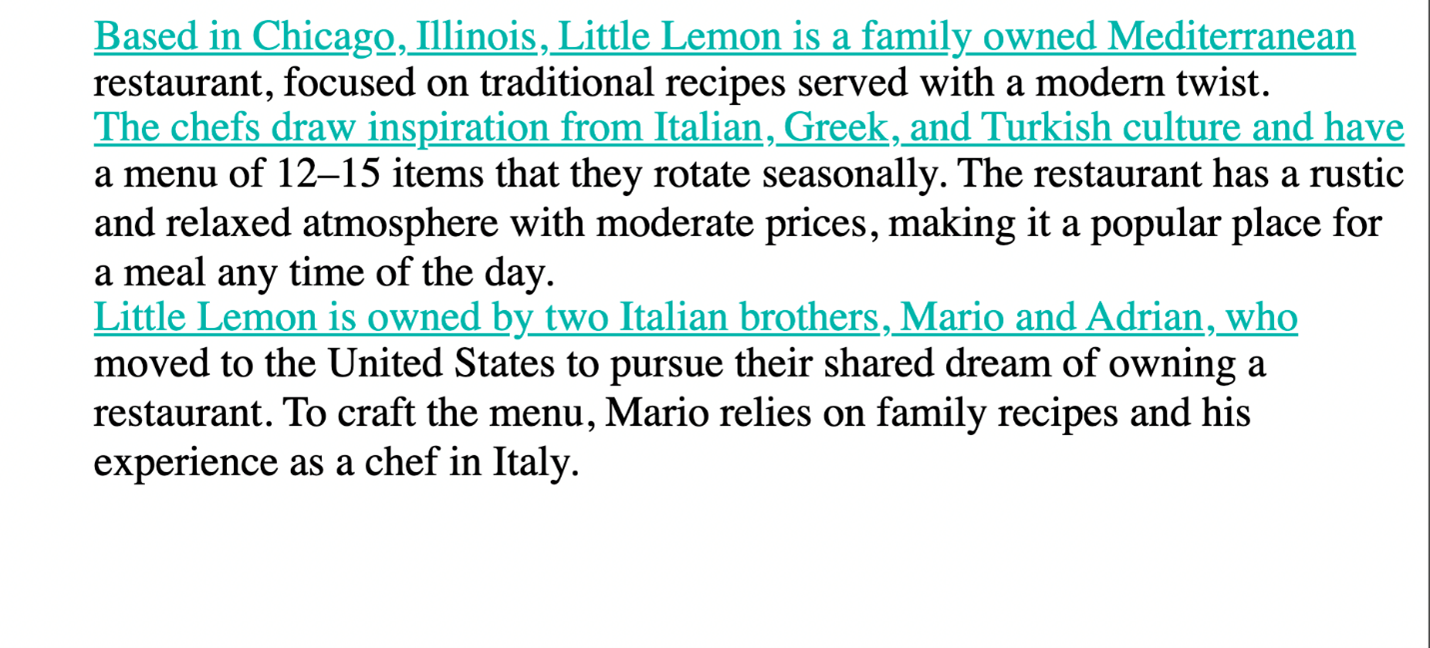
    color: lightseagreen;

    text-decoration: underline;

    line-height: 1;

}

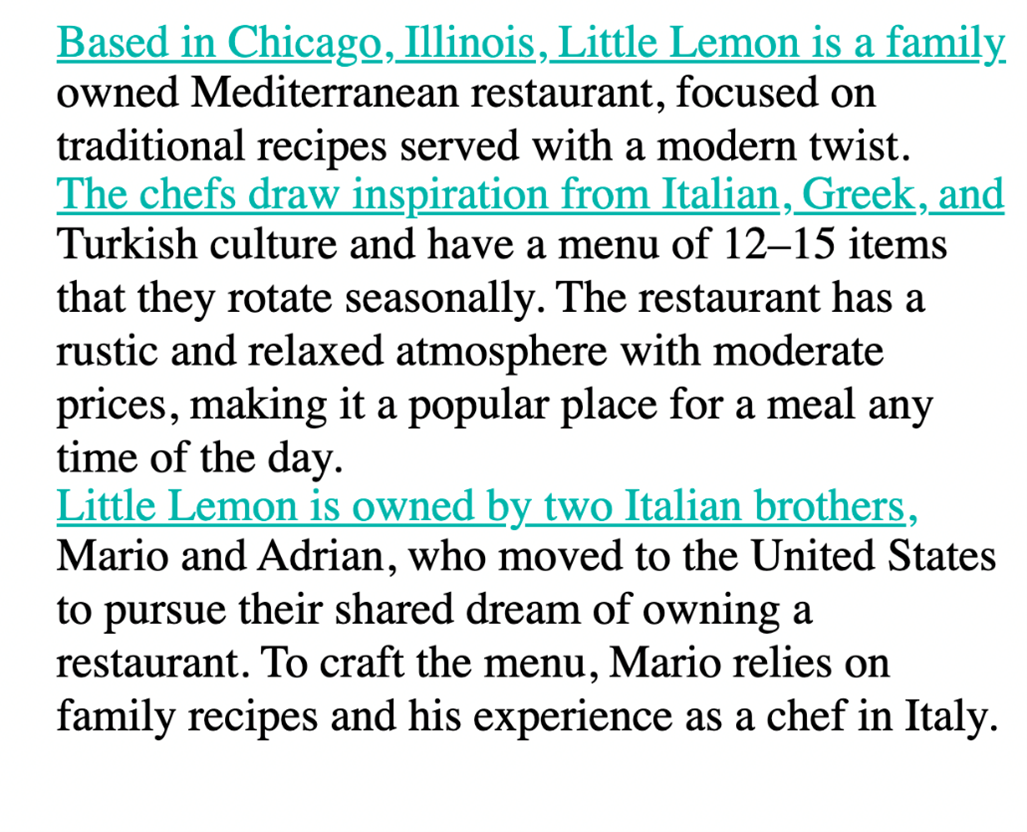
Output:



Because it’s only the first line of each bullet point, it almost functions like dividers between the three different points instead of having to rely on bullets.

Note that the contents of the line to which this pseudo-element is applied will change as you increase or decrease the size of your viewport.

Output:



**::selection**

Selection is another useful pseudo-element. For example, you may use it when you are taking notes on your device because it allows you to highlight specific text. The effect of it becomes obvious only after the user selects content. On web pages today, you will typically see inverted colors from white-black to black-white when selecting a portion of text.

CSS code:

1

2

3

4

5

6

7

8

9

10

ul{

    list-style-type: none;

}

li::selection {

    color:brown;

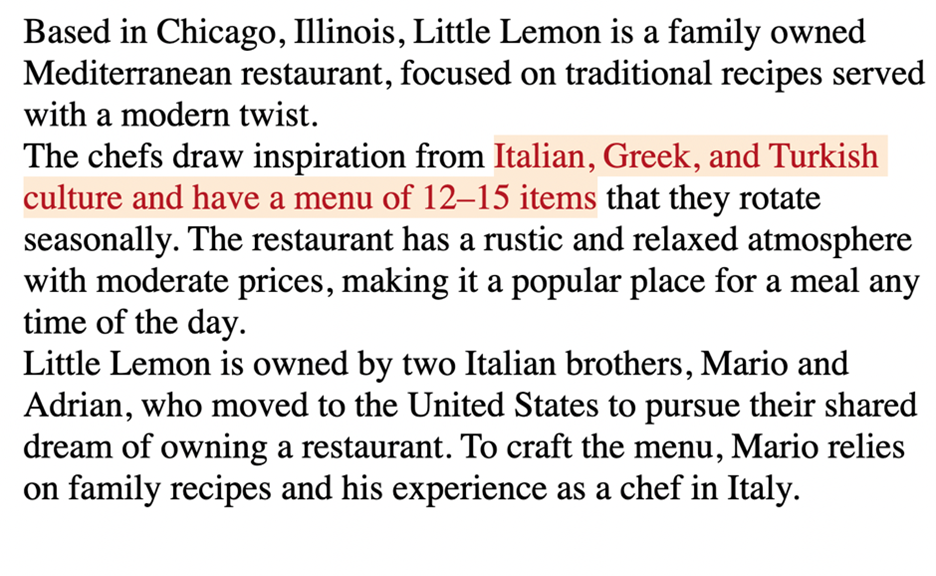
    background-color: antiquewhite;

    line-height: 1;

}

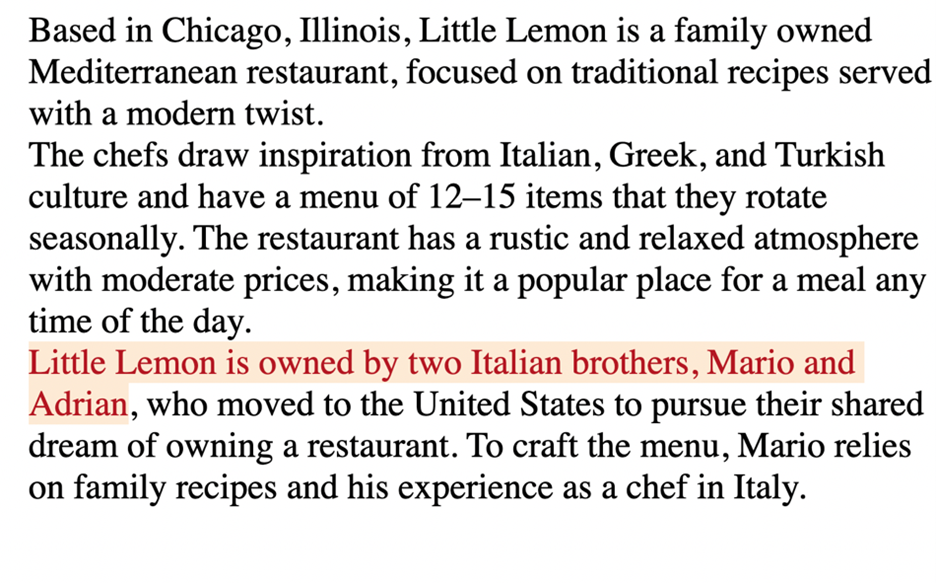
Here is an example of a selection of text.

Output:



And another example of the same text but with a different section selected and highlighted.

Output:



Different segments of the text are highlighted depending on the text that is selected at any given point.

**::marker**

Markers are typically used to add style elements to a list, for instance, to color bullet points. For example, you can enhance the user experience when you use a marker in the following way.

CSS code:

li::marker {

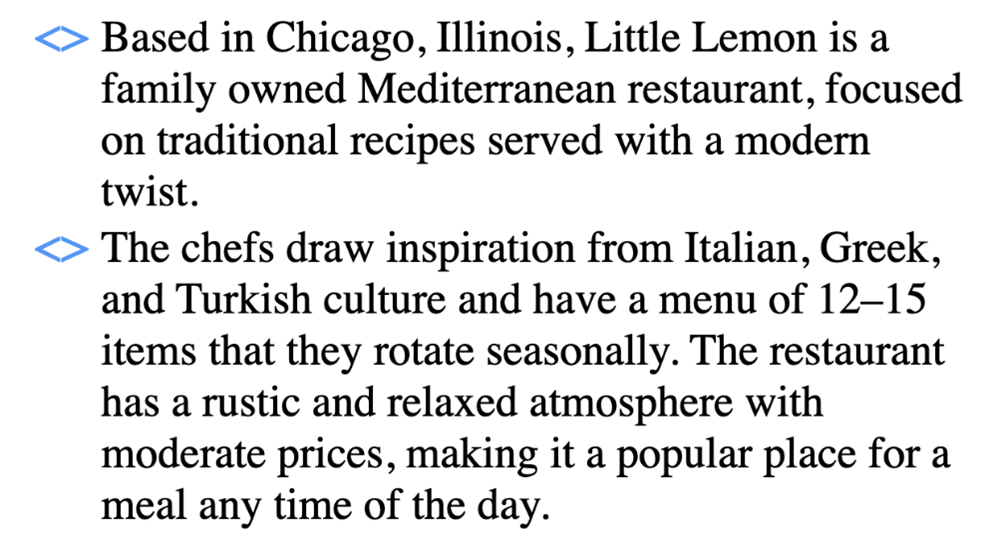
    color: cornflowerblue;

    content: '<> ';

    font-size: 1.1em;

}

Output



Now the bullet points are cornflower blue and they have the shape specified in the code.

**::before and ::after**

One more pair of pseudo-elements are the **::before** and **::after** pseudo-elements. They allow you to add content before and after an element on which they are allowed. In other words, new content can be added to a page without adding HTML code for it. You can also add styling options for this content. Let’s do an example where text is added both before and after some cooking guidelines to identify them as important tips.

HTML code:

<body>

    <p id="tips"> Don't rinse your pasta after it is drained. </p>

    <p> Slice the tomatoes. Take the extra efforts to seed them. </p>

    <p id="tips"> Peel and seed large tomatoes. </p>

</body>

#tips::before{

    background: darkkhaki;

    color:darkslategray;

    content: "Tip:";

    padding-left: 3px;

    padding-right: 5px;

    border-radius: 10%;

}

#tips::after{

    background:darkkhaki;

    color:darkslategray;

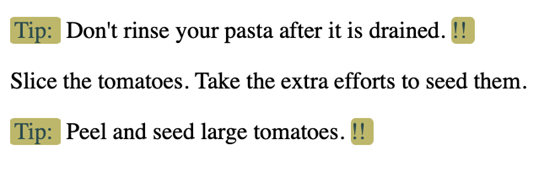
    content: "!!";

    padding-right: 5px;

    border-radius: 20%;

}

Output:



The “content” property is where the text for the guidelines goes. The word “tip” has been added before each guideline thanks to the rules added for **tips::before**. And, each of the three guidelines now has two exclamation marks after them thanks to the rules added for **tips::after**. Note how the second <p> element inside the HTML code remains unaffected. You don’t have to use after and before together like this, but sometimes it is useful to combine them.

The examples covered here illustrate that adding simple code for pseudo-elements can greatly enhance the appearance of websites. There are plenty of other pseudo-elements and some of them are more popular than others. You can follow your own style and explore the creative possibilities that pseudo-classes and pseudo-elements offer.