Attribute selectors

As you know from your study of HTML, elements can have attributes that give further detail about the element being marked up. In CSS you can use attribute selectors to target elements with certain attributes. This lesson will show you how to use these very useful selectors.

Prerequisites:	Basic computer literacy, basic software installed, basic knowledge of working with files, HTML basics (study Introduction to HTML), and an idea of how CSS works (study CSS first steps.)
Objective:	To learn what attribute selectors are and how to use them.

Presence and value selectors

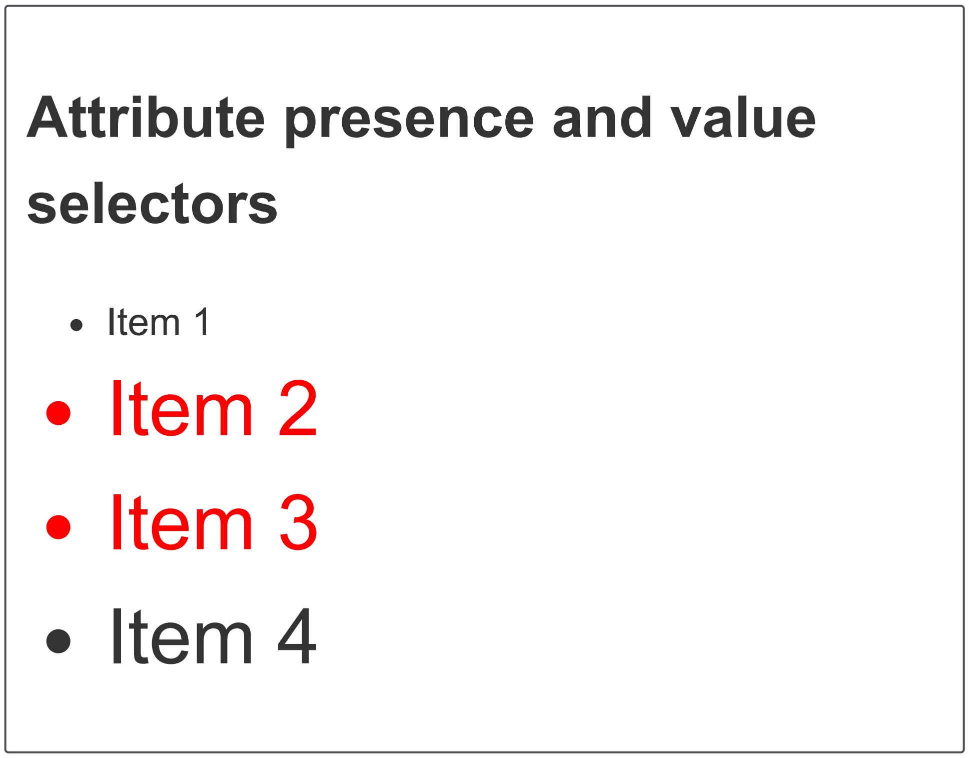
These selectors enable the selection of an element based on the presence of an attribute alone (for example `href`), or on various different matches against the value of the attribute.

Selector	Example	Description

Selector	Example	Description
`[attr]`	`a[title]`	Matches elements with an attribute name of *attr* — the value in square brackets.
`[attr=value]`	`a[href="https://example.com"]`	Matches elements with an attribute name of *attr* whose value is exactly *value* — the string inside the quotes.
`[attr~=value]`	`p[class~="special"]`	Matches elements with an attribute name of *attr* whose value is exactly *value*, or elements with an *attr* attribute containing one or more values, at least one of which matches *value*. Note that in a list of multiple values the separate values are whitespace-separated.
`[attr|=value]`	`div[lang|="zh"]`	Matches elements with an attribute name of *attr* whose value can be exactly *value* or can begin with *value* immediately followed by a hyphen.

In the example below you can see these selectors being used.

- By using `li[class]` we can match any selector with a class attribute. This matches all but the first list item.
- `li[class="a"]` matches a selector with a class of `a`, but not a selector with a class of `a` with another space-separated class as part of the value. It selects the second list item.
- `li[class~="a"]` will match a class of `a` but also a value that contains the class of `a` as part of a whitespace-separated list. It selects the second and third list items.



Attribute presence and value selectors

- Item 1
- Item 2
- Item 3
- Item 4

```
li[class] {
    font-size: 200%;
}

li[class="a"] {
    background-color: yellow;
}

li[class~="a"] {
    color: red;
}
```

```
<h1>Attribute presence and value selectors</h1>
<ul>
    <li>Item 1</li>
    <li class="a">Item 2</li>
    <li class="a b">Item 3</li>
    <li class="ab">Item 4</li>
</ul>
```

Substring matching selectors

These selectors allow for more advanced matching of substrings inside the value of your attribute. For example, if you had classes of `box-warning` and `box-error` and wanted to match everything that started with the string "box-", you could use `[class^="box-"]` to select them both.

Selector	Example	Description
`[attr^=value]`	`li[class^="box-"]`	Matches elements with an attribute name of *attr* whose value has the substring *value* at the start of it.
`[attr$=value]`	`li[class$="-box"]`	Matches elements with an attribute name of *attr* whose value has the substring *value* at the end of it.
`[attr*= ]`	`li[class*="box"]`	Matches elements with an attribute name of *attr* whose value contains at least one occurrence of the substring *value* anywhere within the string.

The next example shows usage of these selectors:

- `li[class^="a"]` matches any attribute value which starts with `a`, so matches the first two list items.
- `li[class$="a"]` matches any attribute value that ends with `a`, so matches the first and third list item.
- `li[class*="a"]` matches any attribute value where `a` appears anywhere in the string, so it matches all of our list items.

Attribute substring matching selectors

- Item 1
- Item 2
- Item 3
- Item 4

```
li[class^="a"] {
    font-size: 200%;
}

li[class$="a"] {
    background-color: yellow;
}

li[class*="a"] {
    color: red;
}
```

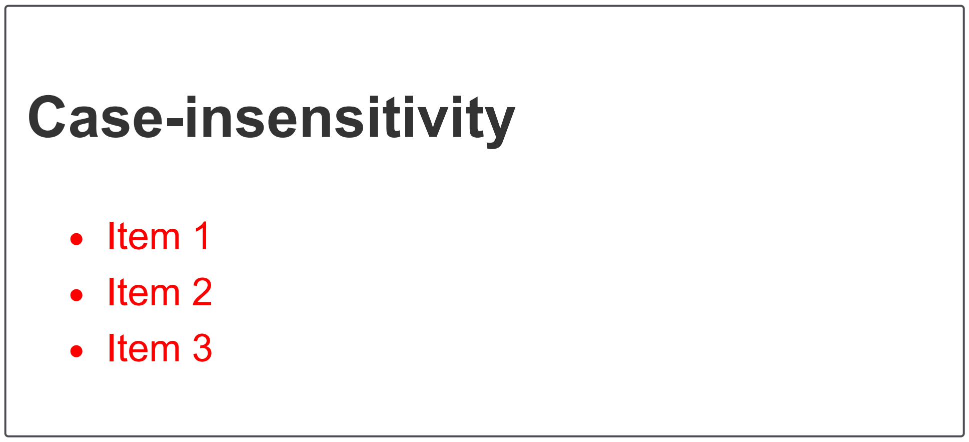
```
<h1>Attribute substring matching selectors</h1>
<ul>
    <li class="a">Item 1</li>
    <li class="ab">Item 2</li>
    <li class="bca">Item 3</li>
    <li class="bcabc">Item 4</li>
</ul>
```

Reset

Case-sensitivity

If you want to match attribute values case-insensitively you can use the value `i` before the closing bracket. This flag tells the browser to match ASCII characters case-insensitively. Without the flag the values will be matched according to the case-sensitivity of the document language — in HTML's case it will be case sensitive.

In the example below, the first selector will match a value that begins with `a` — it only matches the first list item because the other two list items start with an uppercase A. The second selector uses the case-insensitive flag and so matches all of the list items.



Case-insensitivity

- Item 1
- Item 2
- Item 3

```
li[class^="a"] {
    background-color: yellow;
}

li[class^="a" i] {
    color: red;
}
```

```
<h1>Case-insensitivity</h1>
<ul>
    <li class="a">Item 1</li>
    <li class="A">Item 2</li>
    <li class="Ab">Item 3</li>
</ul>
```

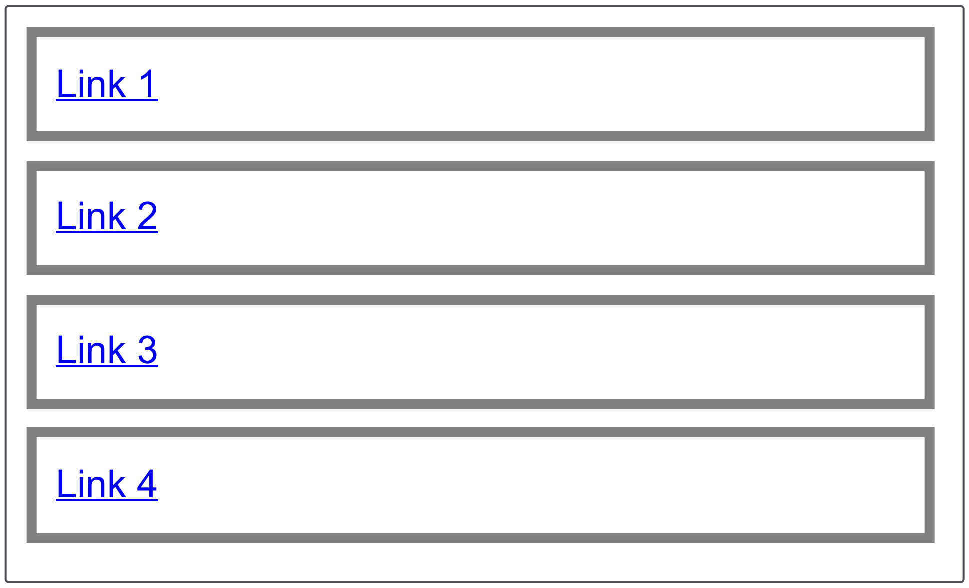
Reset

Note: There is also a newer value `s`, which will force case-sensitive matching in contexts where matching is normally case-insensitive, however this is less well supported in browsers and isn't very useful in an HTML context.

Try it out

In the live example below, add CSS using attribute selectors to do the following:

- Target the `<a>` element with a `title` attribute and make the border pink (`border-color: pink`).
- Target the `<a>` element with an `href` attribute that contains the word `contact` somewhere in its value and make the border orange (`border-color: orange`).
- Target the `<a>` element with an `href` value starting with `https` and give it a green border (`border-color: green`).



```
a {
    border: 5px solid grey;
}
```

```
<ul>
    <li><a href="https://example.com">Link 1</a></li>
    <li><a href="http://example.com" title="Visit
example.com">Link 2</a></li>
    <li><a href="/contact">Link 3</a></li>
    <li><a href="../contact/index.html">Link 4</a></li>
</ul>
```

Reset

Note: You can take a look at the solution here — but try to figure it out yourself first!

Next steps

Now we are done with attribute selectors, you can continue on to the next article and read about pseudo-class and pseudo-element selectors.

← Previous ↑ Overview: Building blocks Next →

In this module

Last modified: Dec 2, 2019, by MDN contributors

Related Topics

Further resources

▸ Common questions

How to contribute