Combinators and selector lists

Previous Overview: Introduction to CSS Next

In our final article on selectors, we'll explore combinators and groups of selectors — two ways of combining multiple selectors together for further useful selection capabilities.

Combinators and selector lists

Using one selector at a time is useful, but can be inefficient in some situations. CSS selectors become even more useful when you start combining them to perform fine-grained selections.

CSS has several ways to select elements based on how they are related to one another. Those relationships are expressed with *combinators* as follows (A and B represent any selector seen above):

Name	Syntax	Selects
Selector list	`A, B`	Any element matching A and/or B (see Groups of selectors on one rule, below - **Group of Selectors** is not considered to be a combinator).
Descendant combinator	`A B`	Any element matching B that is a *descendant* of an element matching A (that is, a child, or a child of a child, etc.). the combinator is one or more spaces or dual greater than signs.
Child combinator	`A > B`	Any element matching B that is a *direct child* of an element matching A.
Adjacent sibling combinator	`A + B`	Any element matching B that is the next *sibling* of an element matching A (that is, the next child of the same parent).
General sibling combinator	`A ~ B`	Any element matching B that is one of the next *siblings* of an element matching A (that is, one of the next children of the same parent).

Combinators example

Let's look at an example with all of this working together:

```
<table lang="en-US" class="with-currency">
  <thead>
    <tr>
      <th scope="col">Product</th>
      <th scope="col">Qty</th>
      <th scope="col">Price</th>
```

```html
    </tr>
  </thead>
  <tfoot>
    <tr>
      <th colspan="2" scope="row">Total:</th>
      <td>148.55</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>Lawnchair</td>
      <td>1</td>
      <td>137.00</td>
    </tr>
    <tr>
      <td>Marshmallow rice bar</td>
      <td>2</td>
      <td>1.10</td>
    </tr>
    <tr>
      <td>Book</td>
      <td>1</td>
      <td>10.45</td>
    </tr>
  </tbody>
</table>

```

Then, let's use the following style sheet:

```css
/* Basic table setup */
```

```css
table {
  font: 1em sans-serif;
  border-collapse: collapse;
  border-spacing: 0;
}

/* All <td>s within a <table> and all <th>s within a <table>
   Comma is not a combinator, it just allows you to target
   several selectors with the same CSS ruleset */
table td, table th {
  border: 1px solid black;
  padding: 0.5em 0.5em 0.4em;
}

/* All <th>s within <thead>s that are within <table>s */
table thead th {
  color: white;
  background: black;
}

/* All <td>s preceded by another <td>,
   within a <tbody>, within a <table> */
table tbody td + td {
  text-align: center;
}

/* All <td>s that are a last child,
   within a <tbody>, within a <table> */
table tbody td:last-child {
  text-align: right;
```

```css
}

/* All <th>s, within a <tfoot>s, within a <table> */
table tfoot th {
  text-align: right;
  border-top-width: 5px;
  border-left: none;
  border-bottom: none;
}

/* All <td>s preceded by a <th>, within a <table> */
table th + td {
  text-align: right;
  border-top-width: 5px;
  color: white;
  background: black;
}


/* All pseudo-elements "before" <td>s that are a last child,
   appearing within elements with a class of "with-currency" that
   also have an attribute "lang" with the value "en-US" */
.with-currency[lang="en-US"] td:last-child::before {
  content: '$';
}

/* All pseudo-elements "after" <td>s that are a last child,
   appearing within elements with the class "with-currency" that
   also have an attribute "lang" with the value "fr" */
.with-currency[lang="fr"] td:last-child::after {
  content: ' €';
```

```
}
```

This has given us the following rather nice table styles:

Product	Qty	Price
Lawnchair	1	$137.00
Marshmallow rice bar	2	$1.10
Book	1	$10.45
	Total:	$148.55

Active learning: Writing your own combinators

The above example was designed to show the sort of complexity you can start to achieve with combinators. In this active learning, we will get you to write some of your own, more simple selectors that include combinators. In this exercise, you need to add a selector to rules 2–4, to:

1. Style links, but only links that are inside the unordered list.
2. Style links inside the unordered list, but only when they are being hovered over.
3. Style only the paragraph that comes directly after the top-level heading.

If you make a mistake, you can always reset it using the *Reset* button. If you get really stuck, press the *Show solution* button to see a potential answer.

HTML Input

HTML Input

```
<ul>
  <li><a href="#">Home</a></li>
  <li><a href="#">Portfolio</a></li>
  <li><a href="#">About</a></li>
</ul>

<h1>Welcome to my website</h1>

<p>Hello, and welcome! I hope you enjoy your time here.</p>
```

CSS Input

```
ul {
  padding: 0;
  list-style-type: none;
}

 {
  text-decoration: none;
  display: block;
  color: black;
  background-color: red;
```

Output

Home
Portfolio
About

Welcome to my website

Hello, and welcome! I hope you enjoy your time here.

Reset   Show solution

Groups of selectors on one rule

You have seen multiple examples of this in action already, but let's spell it out for clarification. You can write groups of selectors separated by commas to apply the same rule to multiple sets of selected elements at once. For example:

```css
p, li {
  font-size: 1.6em;
}
```

Or this:

```css
h1, h2, h3, h4, h5, h6 {
  font-family: helvetica, 'sans serif';
}
```

Important: If a single selector is not supported by the browser, then the whole selector list gets ignored.

The `:is()` pseudo-class is not subject to this limitation, but isn't yet widely supported at the time of writing.

What's next

Congratulations — you've come to the end of our rather long journey into learning about CSS selectors. Even the most skilled web developers are still amazed by what's possible using selectors. However, don't feel bad if you can't remember all the options — bookmark the main selectors page and refer back to it when you need to.

In our next article, we'll turn to another really important fundamental CSS topic — the kind of values properties can have, and what units are involved in expressing the length, color, or any other values you want. Let's explore CSS values and units.

Previous   Overview: Introduction to CSS   Next

In this module

- How CSS works
- CSS syntax
- Selectors
- Simple selectors
- Attribute selectors

- Pseudo-classes and pseudo-elements
- Combinators and multiple selectors
- CSS values and units
- Cascade and inheritance
- The box model
- Debugging CSS
- Fundamental CSS comprehension