Combinators

- Descendant combinator
- Child combinator
- Next-sibling combinator
- Subsequent-sibling combinator
- Combining combinators with selectors
- Summary

The final selectors we will look at are called combinators. Combinators are used to combine other selectors in a way that allows us to select elements based on their location in the DOM relative to other elements (for example, child or sibling).

Prerequisites:	HTML basics (study Basic HTML syntax), Basic CSS selectors.
Learning outcomes:	• The basic concept of combinators. • Descendant and child combinators. • Next- and subsequent-sibling combinators. • Nesting. • Combining combinators with selectors.

Descendant combinator

The **descendant combinator** — represented by a single space ( ) character — combines two selectors such that elements matched by the second selector are selected if they have an ancestor (a parent, a parent's parent, or a parent's parent's parent, etc.) element matching the first selector. Selectors that utilize a descendant combinator are called *descendant selectors*.

```
body article p {
}
```

In the example below, we are matching only the `<p>` element which is inside an element with a class of `.box`.

```
<div class="box"><p>Text in .box</p></div>
<p>Text not in .box</p>
```

```
.box p {
  color: red;
}
```

> **NOTE:** Aside: Compound selectors *MDN learning partner* from Scrimba is an interactive lesson providing a practical treatment of descendant combinators.

Child combinator

The **child combinator** (`>`) is placed between two CSS selectors. It matches only those elements matched by the second selector that are the direct children of elements matched by the first. Descendant elements further down the hierarchy don't match. For example, to select only `<p>` elements that are direct children of `<article>` elements:

```
article > p {
  /* … */
}
```

In this next example, we have an ordered list (ol) nested inside an unordered list (ul). The child combinator selects only those `<li>` elements which are direct children of a `<ul>`, and styles them with a top border.

```
<ul>
  <li>Unordered item</li>
  <li>
    Unordered item
    <ol>
      <li>Item 1</li>
      <li>Item 2</li>
    </ol>
  </li>
</ul>
```

```
ul > li {
  border-top: 5px solid red;
}
```

In the previous example, try removing the `>` that designates the selector as a child selector. You will end up with a descendant selector, and all `<li>` elements will get a red border.

Next-sibling combinator

The **next-sibling combinator** (`+`) is placed between two CSS selectors. It matches only those elements matched by the second selector that come right after the element matched by the first selector. For example, to select all `<img>` elements that are immediately preceded by a `<p>` element:

```
p + img {
  /* … */
}
```

A common use case is to do something with a paragraph that follows a heading, as in the example below. Here, we select any paragraph that shares a parent element with an `<h1>`, and immediately follows that `<h1>`.

```
<article>
  <h1>A heading</h1>
  <p>
    Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion
    daikon amaranth tatsoi tomatillo melon azuki bean garlic.
  </p>

  <p>
    Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette
    tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato.
    Dandelion cucumber earthnut pea peanut soko zucchini.
  </p>
</article>
```

```
body {
  font-family: sans-serif;
}
```

```
h1 + p {
  font-weight: bold;
  background-color: #333333;
  color: white;
  padding: 0.5em;
}
```

In the previous example:

1. Try inserting another element such as an `<h2>` in-between the `<h1>` and the `<p>`. You will find that the paragraph is no longer matched by the selector and so does not get the background and foreground color applied when the element is adjacent.
2. Now modify the `h1 + p` selector so that the special style is applied to the first paragraph once more.

Subsequent-sibling combinator

If you want to select siblings of an element even if they are not directly adjacent, then you can use the **subsequent-sibling combinator** (`~`). To select all `<img>` elements that come *anywhere* after `<p>` elements, we'd do this:

```
p ~ img {
  /* … */
}
```

In the example below, we are selecting all `<p>` elements that come after the `<h1>`, and even though there is a `<div>` in the document as well, the `<p>` that comes after it is selected.

```
<article>
  <h1>A heading</h1>
  <p>I am a paragraph.</p>
  <div>I am a div</div>
  <p>I am another paragraph.</p>
</article>
```

```
body {
  font-family: sans-serif;
}
```

```
h1 ~ p {
  font-weight: bold;
  background-color: #333333;
  color: white;
  padding: 0.5em;
}
```

Combining combinators with selectors

You can combine any of the selectors that we discovered in previous lessons with combinators in order to select part of your document. For example, to select list items with a `class` of `a` which are direct children of a `<ul>`, try the following:

```
ul > li[class="a"] {
}
```

Take care, however, when creating big lists of selectors that select very specific parts of your document. It will be hard to reuse the CSS rules since you have made the selector very specific to the location of that element in the markup.

It is often better to create a simple class and apply that to the element in question. That said, your knowledge of combinators will be very useful if you need to style something in your document and are unable to access the HTML, perhaps due to it being generated by a CMS.

Summary

That's it for selectors, for now. Next, we'll give you some tests that you can use to check how well you've understood and retained the information we've provided on CSS selectors.

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:32:54 UTC