Styling links

- Link states
- Default styles
- Styling links
- Style your own links
- Including icons on links
- Styling links as buttons
- Summary

When styling links, it is important to understand why default link styles are important, how to use pseudo-classes to style link states effectively, and how to style links for use in common, varied interface features such as navigation menus and tabs. We'll look at all these topics in this article.

Prerequisites:	Structuring content with HTML and CSS Styling basics.
Learning outcomes:	• Understand why default link styles are important for usability on the web — they are familiar and help users recognize links. • Styling link states: `:hover`, `:focus`, `:visited`, and `:active`. • Understanding why link states are necessary for accessibility and usability. • Including icons on links. • Creating a navigation menu with lists and links.

Link states

The first thing to understand is the concept of link states — different states that links can exist in. These can be styled using different pseudo-classes:

- **Link**: A link that has a destination (i.e., not just a named anchor), styled using the :link pseudo class.

- **Visited**: A link that has already been visited (exists in the browser's history), styled using the :visited pseudo class.
- **Hover**: A link that is hovered over by a user's mouse pointer, styled using the :hover pseudo class.
- **Focus**: A link that is focused (e.g., moved to by a keyboard user using the Tab key or something similar, or programmatically focused using HTMLElement.focus()) — this is styled using the :focus pseudo class.
- **Active**: A link that is activated (for example, clicked on), styled using the :active pseudo class.

Default styles

The example below illustrates what a link will look and behave like by default, though the CSS is enlarging and centering the text to make it stand out more. You can compare the look and behavior of the default styling in the example with that of other links on this page, which have more CSS styles applied. Default links have the following properties:

- Links are underlined.
- Unvisited links are blue.
- Visited links are purple.
- Hovering a link makes the mouse pointer change to a little hand icon.
- Focused links have an outline around them — you should be able to focus on the links on this page with the keyboard by pressing the tab key.
- Active links are red. Try holding down the mouse button as you click the link.

```
<p><a href="#">A simple link</a></p>
```

```
p {
  font-size: 2rem;
  text-align: center;
}
```

> **NOTE:** All the link examples on this page link to the top of their embedded window. The empty fragment (`href="#"`) is used to create simple examples and ensure the live examples, which are each contained in an iframe, don't break.

Interestingly, the default styles are nearly the same as they were back in the early days of browsers in the mid-1990s. This is because users know and have come to expect this behavior — if

links were styled differently, it would confuse people. This doesn't mean that you shouldn't style links at all. It just means that you shouldn't stray too far from the expected behavior. You should at least:

- Use underlining for links, but not for other things. If you don't want to underline links, at least highlight them in some other way.
- Make them react in some way when hovered/focused, and in a slightly different way when activated.

The default styles can be turned off/changed using the following CSS properties:

- color for the text color.
- cursor for the mouse pointer style — you shouldn't turn this off unless you've got a very good reason.
- outline for the text outline. An outline is similar to a border. The only difference is that a border takes up space in the box, and an outline doesn't: outlines sit on top of the background. The outline is a useful accessibility aid, so it should not be removed without adding another method of indicating the focused link.

> **NOTE:** You are not just limited to the above properties to style your links — you are free to use any properties you like.

Styling links

Now that we've explored the default states in some detail, let's look at a typical set of link styles.

To start with, we'll write out our empty rulesets:

```
a {
}

a:link {
}

a:visited {
}

a:focus {
}
```

```
a:hover {
}

a:active {
}
```

This order is important because link styles build on one another. For example, the styles in the first rule will apply to all the subsequent ones. When a link is activated, it's usually also hovered over. If you put these in the wrong order, and you're changing the same properties in each ruleset, things won't work as you expect. To remember the order, you could try using a mnemonic like **L**o**V**e **F**ears **HA**te.

Now let's add some more information to get this styled properly:

```
body {
  width: 300px;
  margin: 0 auto;
  font-size: 1.2rem;
  font-family: sans-serif;
}

p {
  line-height: 1.4;
}

a {
  outline-color: transparent;
}

a:link {
  color: #6900ff;
}

a:visited {
  color: #a5c300;
}

a:focus {
  text-decoration: none;
  background: #bae498;
```

```
}

a:hover {
  text-decoration: none;
  background: #cdfeaa;
}

a:active {
  background: #6900ff;
  color: #cdfeaa;
}
```

We'll also provide some sample HTML to apply the CSS to:

```
<p>
  There are several browsers available, such as <a href="#">Mozilla Firefox</a>,
  <a href="#">Google Chrome</a>, and <a href="#">Microsoft Edge</a>.
</p>
```

Putting the two together gives us this result:

So what did we do here? This certainly looks different to the default styling, but it still provides a familiar enough experience for users to know what's going on:

- The first two rules are not that interesting to this discussion.
- The third rule uses the `a` selector to get rid of the focus outline (which varies across browsers anyway).
- Next, we use the `a:link` and `a:visited` selectors to set a couple of color variations on unvisited and visited links, so they are distinct.
- The next two rules use `a:focus` and `a:hover` to set focused and hovered links to have no underline and different background colors.
- Finally, `a:active` is used to give the links an inverted color scheme while they are being activated, to make it clear something important is happening!

Style your own links

For this task, we'd like you to take our empty set of rules and add your own declarations to make the links look really cool. Use your imagination, go wild. We are sure you can come up with something cooler and just as functional as our example above.

1. Click **"Play"** in the code block below to edit the example in the MDN Playground.

2. Give the links some default styling that will be applied to them all the time. You don't just have to stick to text color, but make sure the links are still recognizable as links.
3. Give *visited* links a slightly different color from the default link styles you set.
4. Give the link *focus* and *hover* states a distinct style that highlights them over other links. Also, remove the default underline when links are being focused/hovered.
5. Give the *active* state a different style again.

If you make a mistake, you can clear your work using the *Reset* button in the MDN Playground. If you get really stuck, you can view an example solution below the example output.

```
<p>
  There are several browsers available, such as
  <a href="https://www.mozilla.org/firefox/new/" target="_blank">Mozilla Firefox</a>,
  <a href="https://www.google.co.uk/chrome/" target="_blank">Google Chrome</a>, and
  <a href="https://www.microsoft.com/edge" target="_blank">Microsoft Edge</a>.
</p>
```

```
p {
  font-size: 1.2rem;
  font-family: sans-serif;
  line-height: 1.4;
}
```

```
a {
}

a:visited {
}

a:focus,
a:hover {
}

a:active {
}
```

Click here to show the solution

Your finished CSS could look like this:

```
a {
  outline-color: transparent;
  padding: 2px 1px 0;
  color: #265301;
}

a:visited {
  color: #437a16;
}

a:hover,
a:focus {
  background: #bae498;
  text-decoration: none;
}

a:active {
  background: #265301;
  color: #cdfeaa;
}
```

Including icons on links

A common practice is to include icons on links to indicate what kind of content the link points to. Let's look at a basic example that adds an icon to external links (links that lead to other sites). External link icons are usually arrows pointing out of boxes. For this example, we'll use external link icon from icons8.com.

First, some simple HTML to style:

```
<p>
  For more information on the weather, visit our <a href="#">weather page</a>,
  look at <a href="https://en.wikipedia.org/">weather on Wikipedia</a>, or check
  out
  <a href="https://www.nationalgeographic.org/topics/resource-library-weather/">
```

```
    weather on National Geographic</a>.
</p>
```

Next, here's some CSS that will give us the effect we want:

```
body {
  width: 300px;
  margin: 0 auto;
  font-family: sans-serif;
}

a[href^="http"]::after {
  content: "";
  display: inline-block;
  width: 0.8em;
  height: 0.8em;
  margin-left: 0.25em;

  background-size: 100%;
  background-image: url("external-link-52.png");
}
```

So what's going on here? We'll skip over most of the CSS, as it is the same as in previous examples. The last rule, however, is interesting: we're using ::after pseudo-element selector. The `0.8em x 0.8em` pseudo-element is rendered after the anchor text as an inline block, and the icon referenced in its background-image property is placed in the pseudo-element's background. We've also included some margin-left to create some space between the icon and the word that precedes it.

We've used a relative unit: `em`. It sizes the icon in proportion to the anchor's text size. If the text size of the anchor changes, the icon size also adjusts accordingly.

A final word: how did we select just external links? Well, if you are writing your HTML links properly, you should only be using absolute URLs for external links — it is more efficient to use relative links to link to other parts of your own site (as with the first link). The text "http" should therefore only appear in external links (like the second and third ones), and we can select this with an attribute selector: `a[href^="http"]` selects a elements, but only if they have an `href` attribute with a value that begins with "http".

So that's it. Try revisiting the task section above and trying this new technique out!

> **NOTE:** Don't worry if you are not familiar with backgrounds and responsive web design yet; these are explained in other places.

Styling links as buttons

The tools you've explored so far in this article can also be used in other ways. For example, states like hover can be used to style many different elements, not just links — you might want to style the hover state of paragraphs, list items, or other things.

In addition, links are commonly styled to behave like buttons. A website navigation menu can be marked up as a set of links, and this can be styled to look like a set of control buttons or tabs that provide the user with access to other parts of the site. Let's explore how.

First, some HTML:

```
<nav class="container">
  <a href="#">Home</a>
  <a href="#">Pizza</a>
  <a href="#">Music</a>
  <a href="#">Wombats</a>
  <a href="#">Finland</a>
</nav>
```

And now our CSS:

```
body,
html {
  margin: 0;
  font-family: sans-serif;
}

.container {
  display: flex;
  gap: 0.625%;
}

a {
  flex: 1;
```

```
  text-decoration: none;
  outline-color: transparent;
  text-align: center;
  line-height: 3;
  color: black;
}

a:link,
a:visited,
a:focus {
  background: palegoldenrod;
  color: black;
}

a:hover {
  background: orange;
}

a:active {
  background: darkred;
  color: white;
}
```

This gives us the following result:

The HTML defines a nav element with a `"container"` class. The `<nav>` contains our links.

The CSS includes the styling for the container and the links it contains.

- The second rule says:
  - The container is a flexbox. The items it contains — the links, in this case — will be *flex items*.
  - The gap between the flex items will be `0.625%` of the container's width.
- The third rule styles the links:
  - The first declaration, `flex: 1`, means that the widths of the items will be adjusted so they use all the available space in the container.
  - Next, we turn off the default text-decoration and outline — we don't want those spoiling our look.
  - The last three declarations are to center the text inside each link, set the line-height to 3 to give the buttons some height (which also has the advantage of centering the text vertically), and set the text color to black.

Summary

We hope this article has provided you with all you'll need to know about links — for now! The final article in our Styling text module details how to use custom web fonts on your websites.

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:33:01 UTC