Styling links

Previous
Overview: Styling text
Next

When styling links, it is important to understand how to make use of 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:	Basic computer literacy, HTML basics (study Introduction to HTML), CSS basics (study Introduction to CSS), CSS text and font fundamentals.
Objective:	To learn how to style link states, and how to use links effectively in common UI features like navigation menus.

Let's look at some links

We looked at how links are implemented in your HTML according to best practices in Creating hyperlinks. In this article we'll build on this knowledge, showing you best practices for styling links.

Link states

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

- **Link (unvisited)**: The default state that a link resides in, when it isn't in any other state. This can be specifically styled using the `:link` pseudo class.
- **Visited**: A link when it has already been visited (exists in the browser's history), styled using the `:visited` pseudo class.
- **Hover**: A link when it is being hovered over by a user's mouse pointer, styled using the `:hover` pseudo class.
- **Focus**: A link when it has been focused (for example moved to by a keyboard user using the `Tab` key or similar, or programmatically focused using `HTMLElement.focus()`) — this is styled using the `:focus` pseudo class.
- **Active**: A link when it is being activated (e.g. clicked on), styled using the `:active` pseudo class.

Default styles

The following example illustrates what a link will behave like by default (the CSS is simply enlarging and centering the text to make it stand out more.)

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

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

A simple link

Note: All the links in the examples in this page are fake links — a `#` (hash, or pound sign) is put in place of the real URL. This is because if the real links were included, clicking on them would break the examples (you'd end up with an error, or a page loaded in the embedded example that you couldn't get back from.) `#` just links to the current page.

You'll notice a few things as you explore the default styles:

- 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 (On Mac, you may need enable the *Full Keyboard Access: All controls* option by pressing `Ctrl` + `F7` before this will work.)
- Active links are red (Try holding down the mouse button on the link as you click it.)

Interestingly enough, these 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 behaviour — if links were styled differently, it would confuse a lot of people. This doesn't mean that you shouldn't style links at all, just that you should not stray too far from the expected behaviour. 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 being that border takes up space in the box and an outline doesn't; it just sits over the top of the background). The outline is a useful accessibility aid, so think carefully before turning it off; you should at least double up the styles given to the link hover state on the focus state too.

> **Note**: You are not just limited to the above properties to style your links — you are free to use any properties you like. Just try not to go too crazy!

Styling some links

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

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

```
a {

}

a:link {

}

a:visited {

}

a:focus {

}

a:hover {

}
```

```
21
22  a:active {
23
24  }
```

This order is important because the link styles build on one another, for example the styles in the first rule will apply to all the subsequent ones, and when a link is being activated, it is also being hovered over. If you put these in the wrong order, things won't work properly. 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:

```
1   body {
2     width: 300px;
3     margin: 0 auto;
4     font-size: 1.2rem;
5     font-family: sans-serif;
6   }
7
8   p {
9     line-height: 1.4;
10  }
11
12  a {
13    outline: none;
14    text-decoration: none;
15    padding: 2px 1px 0;
16  }
17
```

```css
a:link {
  color: #265301;
}

a:visited {
  color: #437A16;
}

a:focus {
  border-bottom: 1px solid;
  background: #BAE498;
}

a:hover {
  border-bottom: 1px solid;
  background: #CDFEAA;
}

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

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

```html
<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:

> There are several browsers available, such as Mozilla Firefox, Google Chrome, and Microsoft Edge.

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 default text underline and focus outline (which varies across browsers anyway), and adds a tiny amount of padding to each link — all of this will become clear later on.
- 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 different background colors, plus an underline to make the link stand out even more. Two points to note here are:
  - The underline has been created using `border-bottom`, not `text-decoration` — some people prefer this because the former has better styling options than the latter, and is drawn a bit lower, so doesn't cut across the descenders of the word being underlined (e.g. the tails on g and y).

    - The `border-bottom` value has been set as `1px solid`, with no color specified. Doing this makes the border adopt the same color as the element's text, which is useful in cases like this where the text is a different color in each case.
- 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!

Active learning: Style your own links

In this active learning session, 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.

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 insert the example we showed above.

HTML Input

```
<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>
```

CSS Input

```
a {

}
```

```
}

a:link {

}

a:visited {
```

Output

There are several browsers available, such as Mozilla Firefox, Google Chrome, and Mi

Reset Show solution

Including icons on links

A common practice is to include icons on links to provide more of an indicator as to what kind of content the link points to. Let's look at a really simple example that adds an icon to external links (links that lead to other sites.) Such an icon usually looks like a little arrow pointing out of a box — for this example, we'll use this great example from icons8.com.

Let's look at some HTML and CSS that will give us the effect we want. First, some simple HTML to style:

```
<p>For more information on the weather, visit our <a href="http://#">weather page</a>,
look at <a href="http://#">weather on Wikipedia</a>, or check
out <a href="http://#">weather on Extreme Science</a>.</p>
```

Next, the CSS:

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

p {
  line-height: 1.4;
}

a {
  outline: none;
  text-decoration: none;
  padding: 2px 1px 0;
}

a:link {
  color: blue;
}

```

```
a:visited {
  color: purple;
}

a:focus, a:hover {
  border-bottom: 1px solid;
}

a:active {
  color: red;
}

a[href*="http"] {
  background: url('https://mdn.mozillademos.org/files/12982/external-link-52.png') no-repeat 100% 0;
  background-size: 16px 16px;
  padding-right: 19px;
}
```

For more information on the weather, visit our weather page , look at weather on Wikipedia , or check out weather on Extreme Science .

So what's going on here? We'll skip over most of the CSS, as it's just the same information you've looked at before. The last rule however is interesting — here we are inserting a custom

background image on external links in a similar manner to how we handled custom bullets on list items in the last article — this time however we are using `background` shorthand instead of the individual properties. We set the path to the image we want to insert, specify `no-repeat` so we only get one copy inserted, and then specify the position as 100% of the way over to the right of the text content, and 0 pixels from the top.

We also use `background-size` to specify the size we want the background image to be shown at — it is useful to have a larger icon and then resize it like this as needed for responsive web design purposes. This does however only work with IE 9 and later, so if you need to support those older browsers, you'll just have to resize the image and insert it as is.

Finally, we set some `padding-right` on the links to make space for the background image to appear in, so we aren't overlapping it with the text.

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. The text "http" should therefore only appear in external links, 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 contains "http" somewhere inside it.

So that's it — try revisiting the active learning 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 quite commonly styled to look and behave like buttons in certain circumstances — a website navigation menu is usually marked up as a list containing links, and this can be easily 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:

```
<ul>
  <li><a href="#">Home</a></li><li><a href="#">Pizza</a></li><li><a href="#">Music</a></li><li><a href="#
</ul>
```

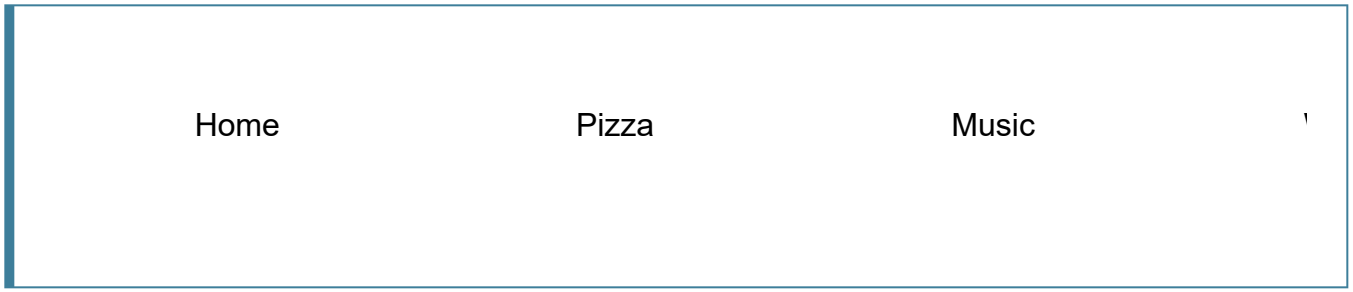
And now our CSS:

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

ul {
  padding: 0;
  width: 100%;
```

```
}

li {
  display: inline;
}

a {
  outline: none;
  text-decoration: none;
  display: inline-block;
  width: 19.5%;
  margin-right: 0.625%;
  text-align: center;
  line-height: 3;
  color: black;
}

li:last-child a {
  margin-right: 0;
}

a:link, a:visited, a:focus {
  background: yellow;
}

a:hover {
  background: orange;
}

a:active {
```

```
    background: red;
    color: white;
}
```

This gives us the following result:



Let's explain what's going on here, focusing on the most interesting parts:

- Our second rule removes the default `padding` from the `<ul>` element, and sets its width to span 100% of the outer container (the `<body>`, in this case).
- `<li>` elements are normally block by default (see types of CSS boxes for a refresher), meaning that they will sit on their own lines. In this case, we are creating a horizontal list of links, so in the third rule we set the `display` property to inline, which causes the list items to sit on the same line as one another — they now behave like inline elements.
- The fourth rule — which styles the `<a>` element — is the most complicated here; let's go through it step by step:
    - As in previous examples, we start by turning off the default `text-decoration` and `outline` — we don't want those spoiling our look.
    - Next, we set the `display` to `inline-block` — `<a>` elements are inline by default and, while we don't want them to spill onto their own lines like a value of `block`

would achieve, we do want to be able to size them. `inline-block` allows us to do this.

- Now onto the sizing! We want to fill up the whole width of the `<ul>`, leave a little margin between each button (but not a gap at the right hand edge), and we have 5 buttons to accommodate that should all be the same size. To do this, we set the `width` to 19.5%, and the `margin-right` to 0.625%. You'll notice that all this width adds up to 100.625%, which would make the last button overflow the `<ul>` and fall down to the next line. However, we take it back down to 100% using the next rule, which selects only the last `<a>` in the list, and removes the margin from it. Done!
- The last three declarations are pretty simple and are mainly just for cosmetic purposes. We 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.

> **Note**: You may have noticed that the list items in the HTML are all put on the same line as each other — this is done because spaces/line breaks in between inline block elements create spaces on the page, just like the spaces in between words, and such spaces would break our horizontal navigation menu layout. So we've removed the spaces. You can find more information about this problem (and solutions) at Fighting the space between inline block elements.

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 fonts on your websites, or web

fonts as they are better known.

Previous | Overview: Styling text | Next

In this module