4.10 Child Selectors

There are still a few Mangie's List CSS details that need to be covered. In this section, we tackle child selectors. In introducing child selectors, it's helpful to compare them to descendant selectors. Remember how descendant selectors work? That's when you have two selectors separated by a space, and the browser searches for a pair of elements that match the selectors such that the second element is a descendant of the first element (i.e., the second element is contained anywhere within the first element). A *child selector* is a more refined version of a descendant selector. Instead of allowing the second element to be any descendant of the first element, the second element must be a child of the first matched element (i.e., the second element must be within the first element, and there are no other container elements inside the first element that surround the second element).

The syntax for a child selector is the same as the syntax for a descendant selector, except that > symbols are used instead of spaces. Here's the syntax:

list-of-elements-separated-with->'s {property1 : value ; property2 : value ; }

For an example, look at this CSS rule from the Mangie's List web page:

```
section > h2 {background-color: palegreen;}
```

That rule matches each `h2` element that is a child element of a `section` element. Go back to Figure 4.11B and look for the `h2` elements inside the `section` containers. Those `h2` elements say "Dining" and "Clothing." Thus, the preceding CSS rule causes browsers to display those two words with pale green background colors.

The > symbol is called a *combinator*. We won't use that term all that often, but if you're perusing the W3C literature (a good bedtime read, by the way), you'll run into it every now and then. The > symbol is called a "combinator" because it's used to combine the element at its left with the element at its right. There are other combinator symbols, and if you're interested, you can learn about them by searching for them on the W3C website.

What Happens When There Are Conflicting CSS Rules

For the `section > h2` CSS rule shown in the previous section, why was the combinator necessary? Why not use the following simpler CSS rule?

```
h2 {background-color: palegreen;}
```

If you look back at the Mangie's List web page source code, you can see that there are three `h2` elements. Without "`section >`" at the left of `h2`, all three `h2` elements would be matched. That means the `h2` element at the top (inside the `header` container) would be pale green, and we don't want that. We want the `h2` element at the top to use one of the other CSS rules in the Mangie's List web page so it gets a powder blue background. Here's that rule:

```
header {background-color: powderblue;}
```

Both CSS rules (the `h2` selector rule and the `header` selector rule) attempt to assign the background color for the `h2` element at the top of the Mangie's List web page. The first rule clearly impacts the `h2` element because the selector is `h2`. The second rule also impacts the `h2` element, but the connection is less straightforward. The second rule's selector is `header`, so it sets the background color for the `header` container. By default, that background color applies to all of the elements inside the header container, including the `h2` element.

So with both rules attempting to assign the background color of the `h2` element at the top, which rule takes precedence? The pale green `h2` rule has higher precedence because it's more specific—its `h2` selector precisely matches the `h2` element that is being targeted. But as stated earlier, we don't want the pale green rule to win for the `h2` element at the top. The solution is to tighten up the pale green rule's selector. Instead of matching all `h2` elements, we match just those elements inside of a `section` container by prefacing the selector with "section >". Here's the result, copied from the Mangie's List web page source code:

```
section > h2 {background-color: palegreen;}
```

Tweaking the Aside Box

Here's another child selector CSS rule from the Mangie's List web page:

```
aside > h3 {margin: 0;}
```

By default, browsers display heading elements, including `h3`, with rather large margins above and below the heading. We already added padding inside the `aside` element's border, so the additional default spacing from the `h3` element creates gaps that are too big. Thus, we specify a margin width of 0 for the `h3` element. This is another example where tweaking was necessary. It would be ideal if you could always write code that displays perfectly. But in the real world, most of the time your first-cut code won't be perfect, and you're going to have to go back and improve it incrementally. That's what tweaking is all about.

Since the Mangie's List web page has only one `h3` element, we could have used a simple `h3` type selector rule (without `aside >`), but we're planning for the future. If someone adds an `h3` element later on, we want it to have a 0 margin only if the `h3` element is inside an `aside` element. That type of thinking and coding makes your web pages more maintainable, which is very important in the real world. Making something easy to maintain means cost savings down the road.

4.11 CSS Inheritance

Now for one last CSS concept in this chapter—inheritance. CSS *inheritance* is when a CSS property value flows down from a parent element to one or more of its child elements. That should sound familiar. It parallels the inheritance of genetic characteristics (e.g., height and eye color) from a biological parent to a child.

Some CSS properties are inheritable and some are not. To determine whether a particular CSS property is inheritable, go to Mozilla's list of CSS keywords at https://developer.mozilla.org/en-US/docs/Web/CSS/Reference and click on the property you're interested in. That should take you to a description of that property, including its inheritability. Of the CSS properties covered so far in this book, here are the ones that are inheritable:

- `color`
- `font` (and all of its more granular properties, like `font-size`)
- `line-height`
- `list-style` (and all of its more granular properties, like `list-style-type`)
- `text-align`
- `text-transform`

To explain CSS inheritance, we'll refer once again to the Mangie's List web page. Specifically, we'll refer to this `aside` element:

```
<aside>
  <h3>Casey's Special</h3>
  Half-price hot dogs when rotisseried more than 24 hours!
</aside>
```

Also, we'll refer to this associated CSS rule:

```
aside {
  border: thin dashed red;
  color: red;
  background-color: white;
  float: right;
  width: 200px;
  padding: 10px;
  margin: 5px;
}
```

In this rule, the only inheritable CSS property is `color`. If you specify `papayawhip` (look it up; it's real) for a `body` element's color, all the elements inside the `body` container would inherit that color. That would cause the browser to use that color when displaying all the text within the web page. So for the Mangie's List web page, the `red` color gets inherited by the `h3` element that is a child of the `aside` element.

Inheritance is blocked for an inheritable property when an element explicitly specifies a new value for that property. In other words, if a parent element and its child element have two different CSS rules with the same property specified, then the child element's property-value pair (and not the parent element's property-value pair) gets applied to the child element. Formally, we say that the child element's property-value pair *overrides* the inherited property-value pair. So, for the Mangie's List web page, if `color: blue;` was specified for the `h3` element inside the `aside` element, then the browser would display blue text for the `h3` element.

For the other properties shown in the preceding `aside` type selector rule, their values do not flow down via inheritance. Thus, inside the `aside` element, the `h3` element does not get its own border. But it does get a background color of white, and it gets floated to the right. Why? Those properties are not inherited, but by applying those properties to the `aside` element, the `h3` child element is naturally affected.