Combining selectors

Combining selectors

Being able to define a CSS selector in terms of a tag, class or id is very powerful. But it's not practical to place classes on every tag in your document, much less to put unique ids throughout. It's also inconvenient to constantly repeat CSS rules. But by combining composing selectors, all that can be avoided.

Comma separated selectors

Let's say we want to make all our <blockquote> tags, <q> tags, and anything with "speech" in it's class string, to be red italic text. How might we do that? We could make three separate rule sets. Or, better, we can separate our selectors with commas (,) before one rule set. Like so:

separate	joined

`blockquote {`
`  color: red;`
`  font-style: italic;`
`}`
`q  {`
`  color: red;`
`  font-style: italic;`
`}`
`.speech  {`
`  color: red;`
`  font-style: italic;`
`}`
	`blockquote,`
	`q,`
	`.speech {`
	`  color: red;`
	`  font-style: italic;`
	`}`

The joined version on the right is much easier to read and maintain.

If the "speech" items need to also be bold, that can simply be added by an additional rule:

```
blockquote,
q,
.speech {
   color: red;
   font-style: italic;
}
.speech { font-weight: bold; }
```

Specialized selectors

If two selectors of different types (like tag and class) appear next to each other with no spacing separating them, then they form a specialized selector. To match, a candidate must match **both** rules. If a tag selector is used, it must appear first.

This is most useful with class and tag selectors, like so:

```
blockquote.speech { font-color: green; }
```

In the example above, the `blockquote.speech` selector is a `blockquote` tag selector combined with a `.speech` class selector. So this rule will not necessarily apply to every `blockquote`, nor every element with the speech class. Instead, it will only apply to those blockquotes that also have the speech class.

It isn't unusual to see multiple classes joined this way as well:

```
.insect.flying { text-decoration: underline; font-weight:bold; }
```

html	**css**	**result**

```
   <ul>
    <li class="bird
flying">parrot</li>
    <li class="bird">ostrich</li>
    <li class="insect">ant</li>
    <li class="insect
flying">wasp</li>
    <li class="insect
flying">moth</li>
    <li class="flying">airplane</li>
   </ul>
```

```
 .insect.flying {
  text-decoration:
underline;
  font-weight: bold;
 }
```

- parrot
- ostrich
- ant
- **wasp**
- **moth**
- airplane

Descendant selectors

In the following HTML, we see some paragraphs that have some links (<a>) inside. The link tags are inside the paragraphs, but not necessarily direct children.

```
1. <section id="intro">Welcome to <a
   href="#palaceland">PalaceLand</a>, world renown <q>Land of
   endless palaces and <a href="#delight">delights</a></q>. As
   you make your way about, remember the words of our founder
   <blockquote>Shouldn't we have <a href="#chairs">chairs</a>?
   Never made much sense wandering room a room looking for a
   place to sit a spell. Folk that don't sit are not likely
   all right in the <a href="#head">head</a></blockquote>
   </section>
2. <section id="guideline">There are guidelines to follow
   while in <a href="#palaceland">PalaceLand</a>. They are
   outlined on the back of your <q>Daring Footman <a
   href="#trademark">(tm)</a></q> card. But the spirit of the
   guidelines are best summed up by our founder
   <blockquote>Don't just <a href="#standthere">stand
   there</a> with your mouth hanging open waiting for a pair
   of nesting birds.</blockquote> (and no <a
   href="#camera_policy">flash photography</a> please.)
   </section>
```

What if we wanted all the links in the introductory section to be red, but all the link in the guideline section to be green? That is what descendant selectors are for. Here is an example for the problem we are facing:

```
#intro a { color: red; }
#guideline a { color: #00FF00; }
```

We merely separate the tag, identifier, or class selectors by a space.

So, in the first rule, we see that the selector will match to any `<a>` tag that is a descendant of `#intro`. The `<a>` tag can appear directly within `#intro`, or be buried within its children. Here is the result:

Welcome to PalaceLand, world renown Land of endless palaces and delights. As you make your way about, remember the words of our founder

> Shouldn't we have chairs? Never made much sense wandering room a room looking for a place to sit a spell. Folk that don't sit are not likely all right in the head

There are guidelines to follow while in PalaceLand. They are outlined on the back of your Daring Footman (tm) card. But the spirit of the guidelines are best summed up by our founder

> Don't just stand there with your mouth hanging open waiting for a pair of nesting birds.

(and no flash photography please.)

But what if we want the links in the founder blockquote in the intro section to be bold? Again, a descendant selector will work. We add this:

```
#intro blockquote a { font-weight: bold; }
```

Any `<a>` tags anywhere inside a `<blockquote>` anywhere inside the `#intro` section will now be bold.

Direct descendant selectors (>)

Sometimes you don't want to apply a style to any _possible_ child, but to only to the direct children. This can be done with the **>** symbol. Use it between selectors to limit the application to the direct children of the parent. For example, this rule, if applied to the HTML of the previous selector, would cause the links in the intro section to be larger, but not the links in any nested quotes or blockquotes. :

```
#intro > a { font-size: large; }
```

Everything selector (*)

The asterisk (*) can be used to match **any** tag. By itself, this is only marginally useful. But combined with other selectors into a descendant selector, it can be pretty useful.

```
body > * { margin-left: 10px; } /* all the _direct_ children of the body receive the margin */
p * { text-decoration: underline; } /* the text of the paragraph will be normal, but any children anywhere inside it will be underlined */
```