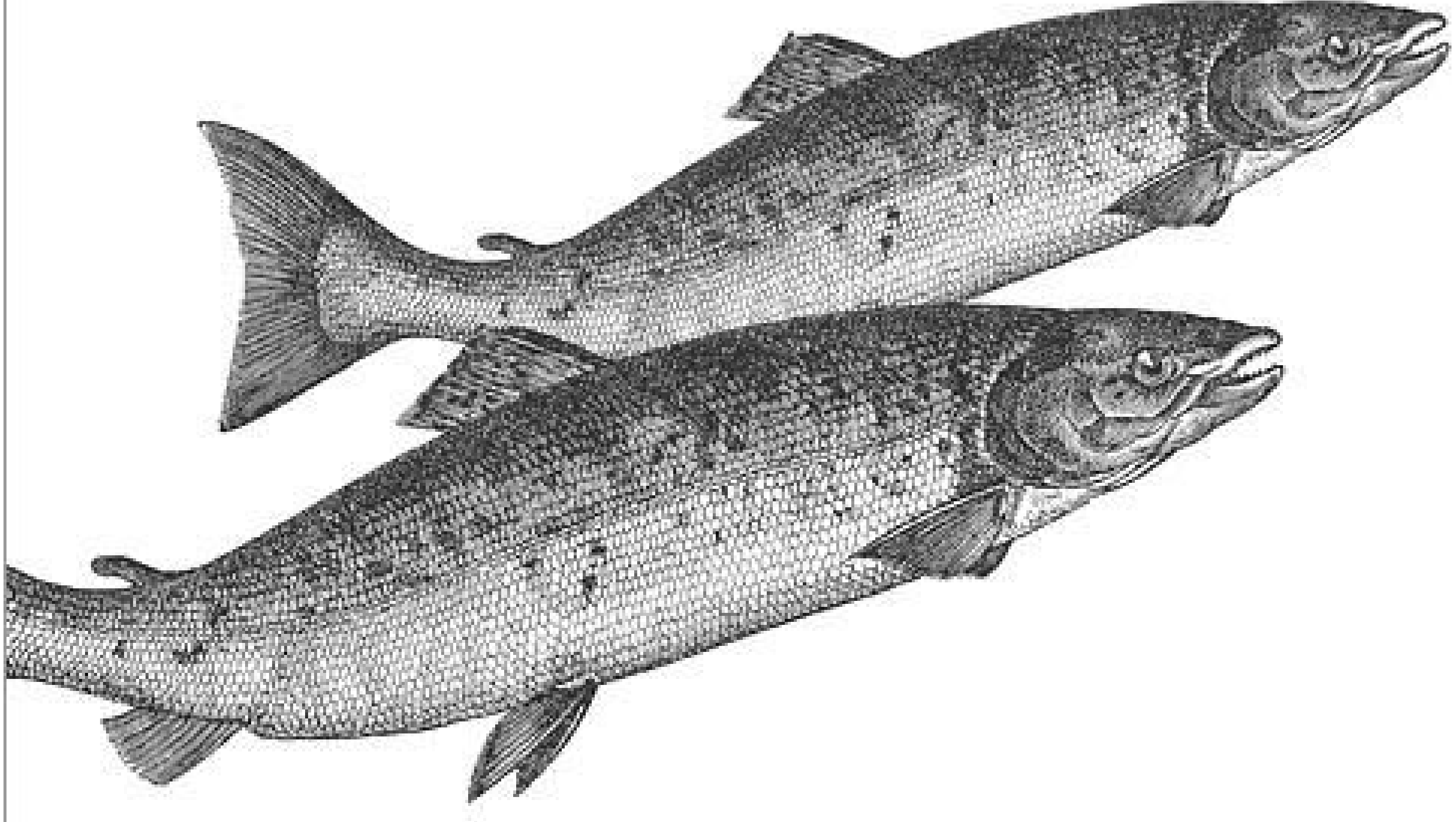
Visual Presentation for the Web

3rd Edition

CSS

The Definitive Guide



O'REILLY®

Eric Meyer

Table of Contents

APPENDIX B

Selector, Pseudo-Class, and Pseudo-Element Reference

Selectors

Universal Selector

This selector matches any element name in the document's language. If a rule does not have an explicit selector, then the universal selector is inferred.

Pattern: *

Examples:

```
* {color: red;}
div * p {color: blue;}
```

Type Selector

This selector matches the name of an element in the document's language. Every instance of the element name is matched. (CSS1 referred to these as *element selectors*.)

Pattern: `element1`

Examples:

```
body {background: #FFF;}
p {font-size: 1em;}
```

Descendant Selector

This allows the author to select an element based on its status as a descendant of another element. The matched element can be a child, grandchild, great-grandchild, etc., of the ancestor element. (CSS1 referred to these as contextual selectors.)

Pattern: `element1 element2`

Examples:

```
body h1 {font-size: 200%;}
table tr td div ul li {color: purple;}
```

Child Selector

This type of selector is used to match an element based on its status as a child of another element. This is more restrictive than a descendant element, since only a child will be matched.

Pattern: `element1 > element2`

Examples:

```
div > p {color: cyan;}
ul > li {font-weight: bold;}
```

Adjacent Sibling Selector

This allows the author to select an element that is the following adjacent sibling of another element. Any text between the two elements is ignored; only elements and their positions in the document tree are considered.

Pattern: `element1 + element2`

Examples:

```
table + p {margin-top: 2.5em;}
h1 + * {margin-top: 0;}
```

Class Selector

In languages that permit it, such as HTML, SVG, and MathML, a class selector using "dot notation" can be used to select elements that have a class containing a specific value or values. The name of the class value must immediately follow the dot. Multiple class values can be "chained" together. If no element name precedes the dot, the selector matches all elements containing that class value.

Patterns: `element1.classname element1.classname1.classname2`

Examples:

```
p.urgent {color: red;}
a.external {font-style: italic;}
.example {background: olive;}
```

ID Selector

In languages that permit it, such as HTML, an ID selector using “hash notation” can be used to select elements that have an ID containing a specific value or values. The name of the ID value must immediately follow the octothorpe (#). If no element name precedes the octothorpe, the selector matches all elements containing that ID value.

Pattern: `element1#idname`

Examples:

```
h1#page-title {font-size: 250%;}
body#home {background: silver;}
#example {background: lime;}
```

Simple Attribute Selector

This allows authors to select any element based on the presence of an attribute, regardless of the attribute’s value.

Pattern: `element1[attr]`

Examples:

```
a[rel] {border-bottom: 3px double gray;}
p[class] {border: 1px dotted silver;}
```

Exact Attribute Value Selector

This allows authors to select any element based on the precise complete value of an attribute.

Pattern: `element1[attr="value"]`

Examples:

```
a[rel="Home"] {font-weight: bold;}
p[class="urgent"] {color: red;;}
```

Partial Attribute Value Selector

This allows authors to select any element based on a portion of the space-separated value of an attribute. Note that `[class~="value"]` is equivalent to `.value` (see above).

Pattern: `element1[attr~="value"]`

Examples:

```
a[rel~="friend"] {text-transform: uppercase;}
p[class~="warning"] {background: yellow;}
```

Beginning Substring Attribute Value Selector

This allows authors to select any element based on a substring at the very beginning of an attribute's value.

Pattern: `element1[attr^="substring"]`

Examples:

```
a[href^="/blog"] {text-transform: uppercase;}
p[class^="test-"] {background: yellow;}
```

Ending Substring Attribute Value Selector

This allows authors to select any element based on a substring at the very end of an attribute's value.

Pattern: `element1[attr$="substring"]`

Example:

```
a[href$=".pdf"] {font-style: italic;}
```

Arbitrary Substring Attribute Value Selector

This allows authors to select any element based on a substring found anywhere within an attribute's value.

Pattern: `element1[attr*="substring"]`

Examples:

```
a[href*="oreilly.com"] {font-weight: bold;}
div [class*="port"] {border: 1px solid red;}
```

Language Attribute Selector

This allows authors to select any element with a `lang` attribute whose value is a hyphen-separated list of values, starting with the value provided in the selector.

Pattern: `element1[lang|="lc"]`

Examples:

```
html[lang|="en"] {color: gray;}
```

Pseudo-Classes and Pseudo-Elements

:active

This applies to an element during the period in which it is activated. The most common example of this is clicking on a hyperlink in an HTML document: during the time that the mouse button is held down, the link is active. There are other ways to activate elements, and other elements can in theory be activated, although CSS doesn't define this.

Type: Pseudo-class

Applies to: An element that is being activated

Examples:

```
a:active {color: red;}
*:active {background: blue;}
```

:after

This allows the author to insert generated content at the end of an element's content. By default, the pseudo-element is inline, but this can be changed using the property `display`.

Type: Pseudo-element

Generates: A pseudo-element containing generated content placed after the content in the element

Examples:

```
a.external:after {content: " " url(/icons/globe.gif);}
p:after {content: " | ";}
```

:before

This allows the author to insert generated content at the beginning of an element's content. By default, the pseudo-element is inline, but this can be changed using the property `display`.

Type: Pseudo-element

Generates: A pseudo-element containing generated content placed before the content in the element

Examples:

```
a[href]:before {content: "[LINK] ";}
p:before {content: attr(class);}
```

:first-child

With this pseudo-class, an element is matched only when it is the first child of another element. For example, `p:first-child` will select any `p` element that is the first child of some other element. It does *not*, as is commonly assumed, select whatever element is the first child of a paragraph; for that, an author would write `p > *:first-child`.

Type: Pseudo-class

Applies to: Any element that is the first child of another element

Examples:

```
body *:first-child {font-weight: bold;}
p:first-child {font-size: 125%;}
```

:first-letter

This is used to style the first letter of an element. Any leading punctuation should be styled along with the first letter. Some languages have letter combinations that should be treated as a single character, and a user agent may apply the first-letter style to both. Prior to CSS2.1, `:first-letter` could be attached only to block-level elements. CSS2.1 expands its scope to include all elements. There is a limited set of properties that can apply to a first letter.

Type: Pseudo-element

Generates: A pseudo-element that contains the first letter of an element

Examples:

```
h1:first-letter {font-size: 166%;}
a:first-letter {text-decoration: underline;}
```

:first-line

This is used to style the first line of text in an element, no matter how many or how few words may appear in that line. `:first-line` can be attached only to block-level elements. There is a limited set of properties that can apply to a first line.

Type: Pseudo-element

Generates: A pseudo-element that contains the first formatted line of an element

Examples:

```
p.lead:first-line {font-weight: bold;}
```

:focus

This applies to an element during the period in which it has focus. One example from HTML is an input box that has the text-input cursor within it; that is, when the user starts typing, text will be entered into that box. Other elements, such as hyperlinks, can also have focus, although CSS does not define which ones.

Type: Pseudo-class

Applies to: An element that has focus

Examples:

```
a:focus {outline: 1px dotted red;}
input:focus {background: yellow;}
```

:hover

This applies to an element during the period in which it is "hovered." Hovering is defined as the user designating an element without activating it. The most common example of this is moving the mouse pointer inside the boundaries of a hyperlink in an HTML document. Other elements can in theory be hovered, although CSS doesn't define which ones.

Type: Pseudo-class

Applies to: An element in a hovered state

Examples:

```
a[href]:hover {text-decoration: underline;}
p:hover {background: yellow;}
```

:lang

This matches elements based on their human language encoding. Such language information must be contained within or otherwise associated with the document; it cannot be assigned from CSS. The handling of `:lang` is the same as for `|=` attribute selectors.

Type: Pseudo-class

Applies to: Any element with associated language-encoding information

Examples:

```
html:lang(en) {background: silver;}
*:lang(fr) {quotes: '« ' ' »';}
```

:link

This applies to a link to a URI that has not been visited; that is, the URI to which the link points does not appear in the user agent's history. This state is mutually exclusive with the `:visited` state.

Type: Pseudo-class

Applies to: A link to another resource that has not been visited

Examples:

```
a:link {color: blue;}
*:link {text-decoration: underline;}
```

:visited

This applies to a link to a URI that has been visited; that is, the URI to which the link points appears in the user agent's history. This state is mutually exclusive with the `:link` state.

Type: Pseudo-class

Applies to: A link to another resource that has already been visited

Examples:

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