CS7026 – CSS3

CSS3 – Attribute Selectors
What You’ll Learn

- We’ll be adding icons to links based on their type, as well as styling photos differently than other types of images, using these CSS3 selectors:
  - The “end of the value” ($) attribute selector
  - The “somewhere in the value” (*) attribute selector
The Base Page

- The page that we’ll be using as our starting point is the finished page from the previous lecture.

- It contains a lot of links to different types of files, but all of these links are styled the same right now.

- It would be nice if links to certain file types were styled differently, to give the user a visual cue to the type of document they’re about to open.

- The page also contains several images, most of which are photos, but one of which is a thumbnail of a calendar. Again, it would be nice to style photos differently than other images, but right now all these images are styled the same way.
What are Attribute Selectors?

- We can add type-based styling to the links and images using attribute selectors.

- Attribute selectors are powerful and useful because they allow you to target specific elements without needing IDs or classes in the HTML.

- Instead, attribute selectors target an element based on the existence or value of a specific attribute on that element.
What are Attribute Selectors?

- For instance, the selector `img[alt]` is made up of the type selector `img` followed by the attribute selector `[alt]`.

- All attribute selectors are designated by square brackets, but what goes in the brackets depends on what you’re trying to target.

- The `img[alt]` selector targets all `img` elements that have an `alt` attribute present.

- Using this selector while testing your pages, you could give all images that have `alt` attributes a bright green outline, so you could see at a glance which images don’t have the outline and need `alt` attributes added.
What are Attribute Selectors?

```css
img[alt] {
    border: 3px solid #0C0;
}
```

```html
<img src="images/cat.jpg" width="320" height="241" alt="My cat Poe">
<img src="photos/dog.jpg" width="320" height="240">
```
What are Attribute Selectors?

- The `img[alt]` selector is an example of the simplest type of attribute selector - one that checks only for the presence of an attribute, regardless of its value.

- It’s one of the four types of attribute selectors that are in the CSS 2.1 spec.
### CSS 2.1 Attribute Selectors

<table>
<thead>
<tr>
<th>ATTRIBUTE SELECTOR</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[attr]</code></td>
<td>Matches an element with an <code>attr</code> attribute present, regardless of its value.</td>
</tr>
<tr>
<td><code>[attr=val]</code></td>
<td>Matches an element with an <code>attr</code> attribute whose value is exactly <code>val</code>.</td>
</tr>
<tr>
<td><code>[attr~=val]</code></td>
<td>Matches an element with an <code>attr</code> attribute whose value is a space-separated list of words, one of which is exactly <code>val</code>.</td>
</tr>
<tr>
<td><code>[attr!=val]</code></td>
<td>Matches an element with an <code>attr</code> attribute whose value is either exactly <code>val</code> or begins with <code>val</code> immediately followed by a hyphen.</td>
</tr>
</tbody>
</table>

- The W3C calls these CSS 2.1 attribute selectors *attribute presence and value selectors.*
CSS3 Attribute Selectors

- CSS3 introduces three new attribute selectors that offer even more fine-grained control over what you’re trying to target.

<table>
<thead>
<tr>
<th>ATTRIBUTE SELECTION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>[attr^=val]</td>
<td>Matches an element with an attr attribute whose value starts with val.</td>
</tr>
<tr>
<td>[attr$=val]</td>
<td>Matches an element with an attr attribute whose value ends with val.</td>
</tr>
<tr>
<td>[attr*=val]</td>
<td>Matches an element with an attr attribute whose value contains val somewhere within it.</td>
</tr>
</tbody>
</table>

- The W3C calls these CSS3 attribute selectors *substring matching attribute selectors* because they match a part of a value instead of the whole thing.
Indicating File Types with Dynamically Added Icons

- Open the base page. Throughout the page, there are links to documents to download, in these file types:
  - PDF
  - MOV
  - DOC
  - JPG

- We are going to add some sort of file-type indicator by using attribute selectors.

- Every link ends with a file-type extension, so we can use the “end of the value” attribute selector to examine the extension and add the appropriate icon as a background image on the `a` element.
Indicating File Types with Dynamically Added Icons

- First, prepare the `a` elements inside the file-download lists to have background images added to them:

```css
ul a {
  display: block;
  min-height: 15px;
  padding-left: 20px;
  background-repeat: no-repeat;
  background-position: 0 3px;
}
```
Indicating File Types with Dynamically Added Icons

- This makes the links block elements with a minimum height matching the height of the icon images, so the icons won’t ever get cut off.

- It also adds left padding to create empty space for each icon to sit in.

- Each icon background image will display only once (no-repeat) and be positioned three pixels down from the top of the link (0 3px) to align it with the top of the text.
Indicating File Types with Dynamically Added Icons

- Now we can add the attribute selectors to target each file type extension:

```html
a[href$=".pdf"] { 
  background-image: url(images/icon_pdf.png);
}
a[href$=".doc"] { 
  background-image: url(images/icon_doc.png);
}
a[href$=".mov"] { 
  background-image: url(images/icon_film.png);
}
a[href$=".jpg"] { 
  background-image: url(images/icon_photo.png);
}
```
Indicating File Types with Dynamically Added Icons

- The `href$=` part of each attribute selector tells the browser “find every `href` attribute that ends with,” and then the values in quotation marks, such as `.pdf`, give the ending attribute value to match against.

- When the browser finds a match, it applies the background image indicated, adding appropriate icons to all the links.
Alternative Icon Ideas

- The icons are a nice little hint to help your users, but if you wanted to be even more obvious and explicit, you could use generated content to write out the file-type extension at the end of each link instead of or in addition to the icons.

- You’d first need to make sure that this information wasn’t already manually written in each link.
Then, you could add the following rule, for example, to write out “(PDF)” after each link to a PDF file:

```css
a[href$=".pdf"]:after {
  content: " (PDF)";
}
```
Combining Multiple Attribute Selectors

- As with almost any other type of selector, you can combine multiple attribute selectors into one to give you even more fine-grained control over what you want to target.

- For instance, what if you wanted to show the photo icon for links to PNG images, but a chart icon for links to PNG images that also happened to be charts?

- Depending on how your images are named, this selector would work:

  ```css
  a[href$=".png"][href*="chart"] {
    background-image: url(images/icon_chart.png);
  }
  ```
This selector tells the browser “find all links that have ‘.png’ at the end of their href attributes and have ‘chart’ somewhere in the href attribute.”

So all of the following links would get matched:

- `<a href="images/chart_locations.png">`
- `<a href="images/piechart.png">`
- `<a href="charts/travel.png">`
Styling Full-size Photos and Thumbnails Differently

- Another great use of attribute selectors in our page is to give the photos a different style than the calendar thumbnail.

- To do this without CSS3, we could simply give the thumbnail a class and apply unique styles to this class.

- This would be quite easy in this particular page. But using classes is not always so simple in the real world.
The Trouble with Classes

- While classes have many legitimate uses, they do have some problems that make them difficult to use in some situations.

- **Classes add bulk to your HTML.**
  - In our example, adding one class isn’t going to hurt anyone, but in much larger pages and sites with more complex styles, a lot of extra classes could be necessary, adding a good chunk to the file size.
  - Any time you can avoid adding classes and IDs to the HTML and use another way to reliably target elements instead, you should do so.
The Trouble with Classes

- **Markup may be controlled by a CMS or plugin**, making it impossible for you to add classes to the HTML.

- **Your client may be the one adding content**, and you can’t count on them to remember to assign the proper classes.

- **You may not be allowed to touch the HTML** if you’re just the CSS developer on a project, or if you’ve been brought into an existing project just to make a few style updates.
The Trouble with Classes

- **Classes can be time-consuming** to add to an existing site with tons of pages, if you’re trying to go back and add new styles. It’s much easier to write CSS that takes advantage of whatever HTML is already there, without your having to go back and add extra style hooks into the HTML.
Using Attribute Selectors to Target by Type

- As long as there is some reliable difference between the HTML used for the thumbnails and the photos, we can tap into that difference with attribute selectors.

- In this case, the distinction is that the calendar thumbnail is saved in the folder named “thumbnails” and the photos are saved in the folder named “photos.”

- The folder name is part of the path in the `src` attribute, so we can use attribute selectors to target each image type independently via particular `src` attribute values.
Let’s start by floating the calendar thumbnail left instead of right:

```css
img[src*="thumbnails"] {
float: left;
margin: 0 20px 10px 0;
}
```

The * attribute selector tells the browser “find every src attribute that has ‘thumbnails’ somewhere within it.” This matches the calendar image:

```html
<img src="images/thumbnails/calendar.jpg" width="90" height="90" alt=""/>
```
Now let’s add some styling to the photos to make them look like Polaroid pictures. Add the following new rule:

```css
img[src*=photos] {
  padding: 5px 5px 30px 5px;
  background: #fff;
  -moz-box-shadow: 3px 6px 8px -4px #999;
  -webkit-box-shadow: 3px 6px 8px -4px #999;
  box-shadow: 3px 6px 8px -4px #999;
  -moz-transform: rotate(2deg);
  -o-transform: rotate(2deg);
  -webkit-transform: rotate(2deg);
  transform: rotate(2deg);
}
```
Using Attribute Selectors to Target by Type

- Now all the photos have a white border around them, a drop shadow behind them, and a slight angle.
More about Attribute Selectors

- Other than link icons and type-based image styling, you might want to use attribute selectors for:
  - Varying the styling of phrases in different languages (using `[lang|=en]`, for instance).
  - Adding a visual indication to elements that have title attributes set (using `[title]`).
  - Removing bullets from lists within navigation `div`s (using `div[id^=nav]` to match `<div id="nav-primary">` and `<div id="nav-secondary">`, for instance).
  - Styling links that go to external sites (using `a[href^=http]` or `a[rel=external]`), that are secure (using `a[href^=https]`), that go to a specific URL (such as `a[href^="paypal.com"]`), that open in new window (using `a[target="_blank"]`), or that go to your own home page (using `a[href="http://myurl.com"]` or `a[href="/index.html"]`).
More about Attribute Selectors

- Checking for empty links before launching a site; see http://fuelyourcoding.com/unconventional-css3-link-checking.
- Displaying the access key of a link (using a:after { content: ‘[’ attr(accesskey) ‘]’ }).
- Displaying the citation source of a blockquote (using blockquote[cite]:after { content: ‘ - ‘ attr(cite) }).
- Styling blockquotes differently based on the value of their cite attributes.
- Displaying an image’s alternative text as its caption (using img[alt]:after { content: attr(alt) }).
- Creating a user style sheet to hide ads on web pages; see http://24ways.org/2005/the-attribute-selector-for-fun-and-no-ad-profit.
- Hiding rules from IE 6.
## Attribute Selectors Browser Support

<table>
<thead>
<tr>
<th></th>
<th>IE</th>
<th>FIREFOX</th>
<th>OPERA</th>
<th>SAFARI</th>
<th>CHROME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes, 7+*</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*IE 7 and later support all the attribute selectors, but are sometimes buggy. Test well.*