Roadmap Series

Issue 7

Diagnostic CSS Styling for Better Standards Support

Prepared by Danilo Celic

4-20-09

Recommended Experience Level

1 Beginner 2 Intermediate 3 Advanced

Diagnostic CSS Styling for Better Standards Support

Roadmap Series

Issue 7

Diagnostic CSS Styling for Better Standards Support

Prepared by Danilo Celic

What is Diagnostic CSS Styling?

You've probably been using diagnostic CSS styling for a while now, even if you didn't know it had a name. When a design isn't displaying the way you want, you add a thick red border to a to see if it is really where you think it should be. The idea is that you make something appear to look different and stand out, so that you can "diagnose" any issues with your page. Such disruptive styling will draw your eyes to the problem you're trying to solve and hopefully make it easier to determine what to do about the issue. This article discusses how to work with Dreamweaver Design Time Style Sheets using Diagnostic CSS to make it obvious what you should be paying attention to from a standards perspective. Diagnostic CSS will allow you to quickly focus on the tags and tag attributes on your page that are important to pay attention to.

By styling elements with specific attributes and attribute values that you do not want to have on your page, you can quickly identify those elements so that you can modify them to better accommodate accessibility and other web design standards. This can include flagging JavaScript that is attached directly to events on tags. For example, an image without an alt attribute is considered to be accessibility issue for visitors with screen readers; or you may be moving to external style sheets and the current version of your pages use a mix of inline styles as well as style related attributes such as align and border. These examples, and many others, can be identified by using specially crafted diagnostic styling.

Starting points

A set of Diagnostic CSS rules can be built from scratch using only the rules that you've identified that matter to your specific pages, but having a starting point is quite useful to see what elements and attributes you might be interested in taking a look at. Eric Meyer has a page that discusses Diagnostic CSS along with a CSS file that can be used as a starting point: http://meyerweb.com/eric/tools/css/diagnostics/.

Here are the basic diagnostic styles that Eric starts off with:

```
div:empty, span:empty,
li:empty, p:empty,
td:empty, th:empty {padding: 0.5em; background: yellow;}

*[style], font, center {outline: 5px solid red;}
*[class=""], *[id=""] {outline: 5px dotted red;}

img[alt=""] {border: 3px dotted red;}
img:not([alt]) {border: 5px solid red;}
img[title=""] {outline: 3px dotted fuchsia;}
img:not([title]) {outline: 5px solid fuchsia;}

table:not([summary]) {outline: 5px solid red;}
table[summary=""] {outline: 3px dotted red;}
```

```
th {border: 2px solid red;}
th[scope="col"], th[scope="row"] {border: none;}

a[href]:not([title]) {border: 5px solid red;}
a[title=""] {outline: 3px dotted red;}
a[href="#"] {background: lime;}
a[href=""] {background: fuchsia;}
```

The styles are broken down into 5 groups:

1. tags with no contents (empty) selectors.
2. tags with a style attribute, font and center tags, and tags with empty class or id attributes.
3. image related selectors.
4. table related selectors.
5. link related selectors.

The intention of each group of selectors is to highlight, in a disruptive manner, elements that match the diagnostic selectors in Eric's diagnostic CSS. For more information about the specific selectors used in this starting point, as well as the reasoning behind the styles applied, make sure to check out Eric's 24Ways.org article: http://24ways.org/2007/diagnostic-styling.

Using Design Time Style Sheets

Dreamweaver's CSS support in its Design View has steadily gotten better with each new release. To accommodate dynamically generated pages, a few versions back, the Dreamweaver team added in Design time Style sheets (DTSS) to allow users to set up one or more styles sheets that are to be considered part of the page, without needing those style sheets being directly added to the page. This allows a developer to add a style sheet via a server side include, or dynamically via JavaScript, but still be able to work within Dreamweaver's Design View and get a pretty good representation of what a visitor might see.

It would be fairly easy to download Eric Meyer's starting point and link that CSS file into any file that you want to work with, and then preview the diagnostic styles in a browser; so why use a DTSS instead of directly linking in a diagnostic CSS file? The most important reason, and why I recommend using DTSS, is that the diagnostic CSS file isn't linked into your page, and therefore when you publish your modified page, you don't have to remember to remove the diagnostic CSS file. Remember, diagnostic styling is purely meant to be a design helper, and not something that your visitors see, so linking the file directly is opening yourself up to a potential workflow bug.

Here's how you can add a DTSS to your page:

1. Create a CSS file that contains diagnostic styles.
2. With that page open, go to **Format > CSS Styles > Design-time**.
3. Click the Show only at design time **Add Item (+)** button.
4. Locate and select the style sheet that contains your diagnostic CSS selectors.
5. Click **OK** twice to close both dialogs.

Now that your DTSS is attached, you will begin to see what issues your page has according to the selectors in your diagnostic CSS.

Modifying styles for Dreamweaver's Design View

Now, I mentioned that Dreamweaver's Design View has been improving with each version; however, it's not perfect. In particular, the :empty and :not pseudoclasses aren't supported, and the outline property is not supported. I'm sure that there are other limitations, but these are the ones that are specific to Eric's starting point and Dreamweaver's Design View. While I've not found a workaround to the lack of :empty support; the :not psuedoclass can be worked around through judicious use of the cascade by styling an element with a border and then removing that border when the element has the appropriate attributes present. The border attribute can used to obtain a similar look to what would be provided within the browser. As inline elements will not display a border (except for images), any matched elements that need to display a border will also need to have display:block applied to the element. Below is a modified version of Eric's starting point that seems to work well enough.

> **Note:** I've also added !important to each style to ensure that the styles will be applied and override other styles that may be more specific.

```
*[style], font, center {display: block !important; border: 5px solid red !important;  }
*[class=""], *[id=""] {display: block !important; border: 5px dotted red !important; }

img {border: 5px solid fuchsia !important; }
img[alt][title] {display: block !important; border-width: 0 !important; }
img[alt=""] {display: block !important; border: 3px dotted red !important; }
img[title=""] {display: block !important; border: 3px dotted fuchsia !important; }

table{border: 5px solid red !important; }
table[summary] {border-width: 0 !important; }
table[summary=""] {border: 3px dotted red !important; }
th {border: 2px solid red !important; }
th[scope="col"], th[scope="row"] {border: none !important;}

a[href] {display: block !important; border: 5px solid red !important; }
a[href][title] { border-width:0 !important;}
a[title=""] {display: block !important; border: 3px dotted red !important; }
a[href="#"] {background: lime !important; }
a[href=""] {background: fuchsia !important; }
```

Check out Eric's demo page at http://meyerweb.com/eric/tools/css/diagnostics/demo-not.html with the above styles applied as a DTSS rather than the in page styles. As you can see, it's pretty easy to see what you may want to take a closer look at.

You're now disruptively identifying elements that do not meet the coding standards that are flagged by your Diagnostic CSS.

Styling to Identify Inline JavaScript

JavaScript that has been applied to elements via the event attributes, such as `onclick`, `onmouseover`, etc. can also be identified through diagnostic CSS. By adding the following styles to your DTSS, you can highlight inline JavaScript for a few common events with a background color that will stand out using code similar to the following:

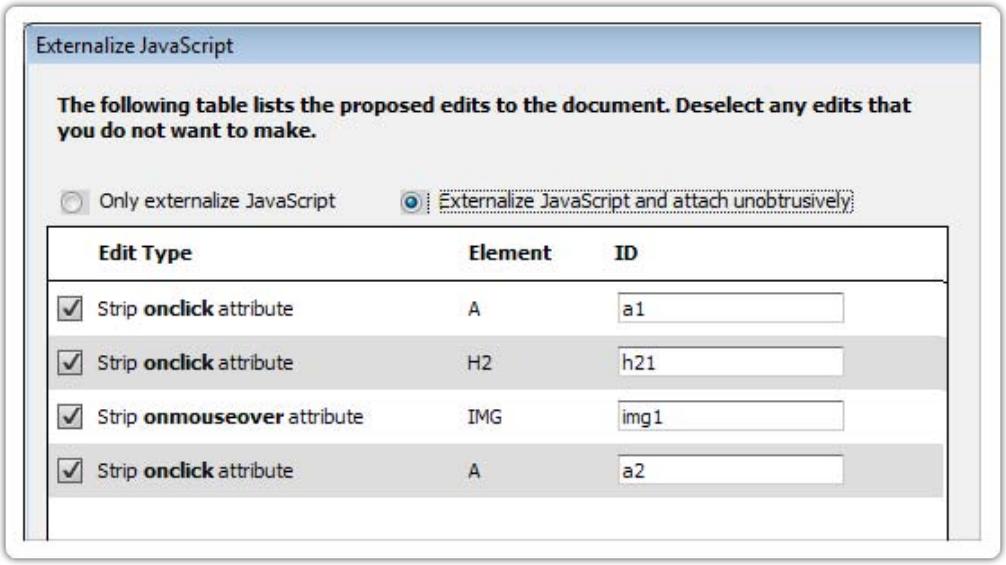
```
*[onclick], *[onmouseover], *[onmouseout] { background:lime; }
```

If you need to track down other events, simply add `*[eventname]` to the list of selectors, and away you go. Please be aware of the cascade and specificity, for example, `*[onclick]` will not override a more specific selector such as `a[title=""]`. Once you have fixed the empty title attribute on the link, then assuming that no other style is more specific, you'll see the styling that the `*[onclick]` selector adds to the link.

When you've identified elements that have inline JavaScript applied to an event attribute of a tag, then you can address them one at a time, or you can move all inline JavaScript out to an external file by using the Externalize JavaScript command. To do this:

1. Go to **Commands > Externalize JavaScript**.
2. In the Externalize JavaScript dialog, select **Externalize JavaScript and attach unobtrusively**.

> **Note:** You'll be warned that any events that are applied via Dreamweaver behaviors will no longer be editable



3. Click **OK**.
4. You'll be presented with a summary of the actions that were taken as well as a list of files that you'll need to upload that contain the JavaScript necessary to replicate the JavaScript that was moved out of your document.

Your page should no longer display any of the styles that are focused on the event attributes.

Diagnostic CSS Styling for Better Standards Support

Roadmap Series

Issue 7

Moving on

I hope that your eyes have been opened to what CSS can do for you beyond making pages look the way that you want them do. When you can identify elements that do not meet the coding standards that you're trying to support, you're making the pages you work on better as well as a better experience for your visitors.

Resources

Eric Meyer: CSS Tools: Diagnostic CSS
http://meyerweb.com/eric/tools/css/diagnostics/

Eric Meyer: Diagnostic Styling 24Ways.org article:
http://24ways.org/2007/diagnostic-styling

Neal Grosskopf: CSS Diagnostics - Find Deprecated Elements Using CSS
http://www.nealgrosskopf.com/tech/thread.asp?pid=3

Dreamweaver CS4 Design time style sheets documentation
http://help.adobe.com/en_US/Dreamweaver/10.0_Using/WScbb6b82af5544594822510a94ae8d65-7e17a.html

Dreamweaver CS4 Externalize JavaScript command documentation
http://help.adobe.com/en_US/Dreamweaver/10.0_Using/WSC9E873BB-A381-437b-8CCF-A7872B575723.html

This free training piece is part of the Roadmap Series. Click here to view all other Roadmaps.

About WebAssist

WebAssist helps you build better websites faster by offering software, solutions, and training needed to succeed on the web. WebAssist.com hosts a thriving community of over 300,000 designers, developers, and business owners. WebAssist's partners include Adobe, Microsoft, and PayPal.

Join our Community

From free software and training, to special offers, there is something for everyone. Come take a look at what is waiting for you and sign up today to gain access to all of the valuable benefits awarded to the members of our community. Click here to join our community.

WEBASSIST®
Build better websites, faster.

WebAssist.com Corporation
8899 University Center Lane
San Diego, CA 92122
USA
www.webassist.com