

NOVEMBER 02, 2001

Alternative Style: Working With Alternate Style Sheets

So you've got a web page. You've marked it up with structural XHTML. You've also been a good little web developer and used style sheets to control what your document looks like. You've even gone the extra mile and created several alternative style sheets to show how hardcore you are.

Great. But now you need a cross-browser way to dynamically switch between the style sheets.

Styling your site

Style sheets can be associated with documents using a list of link elements in the head. There are three different relationships external style sheets can have with the document: persistent, preferred, and alternate.

PERSISTENT

These style sheets are always enabled (they are always "on") and are combined with the active style sheet. They can be used for shared rules common to every style sheet. To make a style sheet persistent, the `rel` attribute is set to "stylesheet" and no `title` attribute is set.

To make the style sheet `paul.css` persistent, the following link element would be included in the head:

```
<link rel="stylesheet"
type="text/css" href="paul.css" />
```

PREFERRED

These style sheets are enabled by default (they are "on" when the page is loaded). They can then be disabled if the user selects an alternate style sheet.

To make a style sheet preferred, the `rel` attribute is set to "stylesheet" and the style sheet is named with the `title` attribute.

Several preferred style sheets can be grouped together by giving them identical title attributes. These grouped style sheets are then all enabled and disabled together. If more than one group of preferred style sheets are declared, the first group takes precedence.

To make `paul.css` preferred, a title attribute is added, giving the default style a name.

```
<link rel="stylesheet"
type="text/css" href="paul.css"
title="bog standard" />
```

ALTERNATE

These style sheets can be selected by the visitor as alternatives to the preferred style sheet. This allows the visitor to personalize a site and choose his or her favorite scheme. They can also be used for accessibility.

To specify an alternate style sheet, the `rel` attribute is set to "alternate stylesheet" and the style sheet is named with a title attribute. As with preferred sheets, these style sheets can also be grouped together by giving them identical title attributes.

Using the previous example again; to make `paul.css` into an alternate style sheet, the keyword "alternate" is added to the `rel` attribute.

```
<link rel="alternate stylesheet"
type="text/css" href="paul.css"
title="wacky" />
```

Note that these relationships only apply to external style sheets which are included using the `link` element.

Swappin' Styles

When a document is initially loaded, the persistent and preferred style sheets are applied to the document. The alternate style sheets can then be selected by the user. The W3C tells us that the

browser should give us a choice of the style sheet we want to use, and suggests that perhaps a drop-down menu or tool bar will be provided.

So far, so good. We have several style sheets and the visitor can choose their favorite from a menu. But then we encounter a problem. A major one. Mozilla provides a menu to select the style sheet we want to use under the view menu item. But Microsoft Internet Explorer (MSIE) provides no such menu. So we have several style sheets, and no way to access them in MSIE.

Here's where a little bit of JavaScript can be used along with the DOM to provide a way for MSIE and Mozilla users to select the style sheet they want to use. Their preference can also be stored in a cookie. And because we are using the link tags as the W3C tells us to, the JavaScript doesn't interfere with the menu in Mozilla, and it degrades very gracefully.

The Script

First we need the script to be able to differentiate between the three different types of style sheet. This is relatively easy to do, as we only need to check two of the attributes of each link element.

Is it a link to a style sheet?

```
HTMLLinkElement.getAttribute("rel").indexOf("style") != -1
```

Is there a title attribute?

```
HTMLListElement.getAttribute("title")
```

Does the rel attribute contain the keyword "alternate"?

```
HTMLLinkElement.getAttribute("rel").indexOf("alt") != -1
```

Note that we check for the string "alt" because some browsers accept the keyword "alternative" in place of "alternate."

Using these three checks we can write a function to switch style sheets. This involves looping through every link element in the document, disabling all preferred and alternate style sheets that we *don't* want active, and enabling all preferred and alternate style sheets that we *do* want active.

Note that only preferred and alternate style sheet link elements will have a title attribute.

The change function looks like this:

```
function setActiveStyleSheet(title) {
    var i, a, main;
    for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
        if(a.getAttribute("rel").indexOf("style") != -1
            && a.getAttribute("title")) {
            a.disabled = true;
            if(a.getAttribute("title") == title) a.disabled = false;
        }
    }
}
```

COOKIES

Now we can change the style sheet. Cool. We have a more personalized page. Excellent. But we don't have a personalized site. The preference is only applied to the current page; when we leave the current page the preference leaves with us. This situation, however, can be rectified with a cookie.

To store a cookie we need another function to return the current style sheet. We also need two functions to store and read the cookie.

To return the current style sheet we look for an active preferred or alternate style sheet and check its title.

First we loop through all the link elements in the document again. We then check whether the link is a style sheet. If it is, we check whether the style sheet has a title. This tells us that the style sheet is either preferred or alternative.

The last check is to see whether or not the style sheet is active. If all three checks return true, we have the current style sheet and we can return the title.

The function ends up looking like this:

```
function getActiveStyleSheet() {
var i, a;
for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
    if(a.getAttribute("rel").indexOf("style") != -1
    && a.getAttribute("title")
    && !a.disabled) return a.getAttribute("title");
}
return null;
}
```

As this is an article on style, and cookies are a completely different topic, I won't explain the cookie functions here, but I will include them for your convenience (these functions are written by ALA author Peter-Paul Koch).

```
function createCookie(name,value,days) {
if (days) {
    var date = new Date();
    date.setTime(date.getTime()+ (days*24*60*60*1000));
    var expires = "; expires="+date.toGMTString();
}
else expires = "";
document.cookie = name+"="+value+expires+"; path=/";
}
function readCookie(name) {
```

```

var nameEQ = name + "=";
var ca = document.cookie.split(';');
for(var i=0;i < ca.length;i++) {
    var c = ca[i];
    while (c.charAt(0)==' ') c = c.substring(1,c.length);
    if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length,c.length);
}
return null;
}

```

To use these cookie functions, we need to add onload and onunload event listeners to the window.

ONLOAD

There is a w3c specified DOM Level 2 attribute, "disabled," that is set to false when a style sheet is applied to the document. This attribute is correctly implemented in Mozilla, but unfortunately not in MSIE.

MSIE *does* have a proprietary HTML attribute, also called "disabled," that applies to link elements.

This attribute is initially set to false for all link elements.

To set the MSIE disabled attribute to match the DOM Level 2 disabled attribute, we can call the `setActiveStyleSheet()` function with the name of the preferred style sheet.

To find out which style sheet is the preferred style sheet, we need another function. Because this function is so similar to the `getActiveStyleSheet()` function I'm not going to explain how it works, but here is what it may look like:

```

function getPreferredStyleSheet() {
var i, a;
for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
    if(a.getAttribute("rel").indexOf("style") != -1
        && a.getAttribute("rel").indexOf("alt") == -1
        && a.getAttribute("title")

```

```
        ) return a.getAttribute("title");
    }
    return null;
}
```

In the onload function, we first set a title variable. This either holds the value of the previous style sheet that is stored in the cookie, or if there isn't one, the title of our preferred style sheet. To keep things logical, let's call the cookie "style."

Next we call up the `setActiveStyleSheet()` function passing the title variable as the title. Our onload function looks something like this:

```
window.onload = function(e) {
    var cookie = readCookie("style");
    var title = cookie ? cookie : getPreferredStyleSheet();
    setActiveStyleSheet(title);
}
```

Note that it may be desirable to call this function before the onload event as well, causing the document to "paint" with our style sheet preference.

If you choose to do this, make sure the function is called after the functions and the link elements have been defined.

ONUNLOAD

To save the cookie in the onunload event is simpler. All we have to do is use the `getActiveStyleSheet()` function to return the active style sheet, and save this in a cookie. Using the function to store a cookie we will end up with something like this:

```
window.onunload = function(e) {
    var title = getActiveStyleSheet();
    createCookie("style", title, 365);
}
```

Puttin' it all together

To use these functions to make your website more sexy, you need to include them in your document. To make it easy, I have put them all together in a javascript file, ready for you to download and add to your site.

Download styleswitcher.js

To include the javascript file, you add a script element to the head of your document, making sure that it is put below all the style sheet link elements you have. The HTML would look like this:

```
<script type="text/javascript"
src="/scripts/styleswitcher.js"></script>
```

To allow the visitor to change the active style sheet, you could use javascript onClick events. For example, to have the option to switch between two themes with titles "default" and "paul," you could use the following HTML:

```
<a href="#"
onclick="setActiveStyleSheet('default');
return false;">change style to default</a>
```

```
<a href="#"
onclick="setActiveStyleSheet('paul');
return false;">change style to paul</a>
```

Once the visitor has selected a theme, it will be stored in a cookie. To use the same theme throughout your website, the same style sheet and javascript link elements should be included in the head of every page of the site.

THAT'S ALL, FOLKS!

There you have it, a customizable website that uses link elements to link to style sheets as the W3C has told us we should. Enjoy!

Learn More

Related Topics: CSS, HTML and XHTML, Scripting

About the Author



Paul Sowden is a teenager from London, England. He believes the web would be a better place if everyone used standards-compliant design techniques.