Advanced Javascript

Webworks – A Workshop Series in Web Design (Session 5)

Table of Contents:

Browser and Object Detection

1. Introduction

There are many browsers out in the great unknown of the internet today. Internet Explorer, Netscape, Mozilla Firefox, and Opera are just a few of them, and even they have internal version numbers as well. As you can imagine, each has slightly different, or even vastly different, ways of interpreting your JavaScript or HTML code. Some HTML tags are supported in some but not in others. Not all browsers support the same version of JavaScript. Some functions may be support on one browser, but not on the next. Displayed text or boxes could even look misaligned. How can you limit the number of bugs on your page?

2. Browser Detection vs Object Detection

Browser detection sniffs out the browser of the entire page, while object detection is a more dynamic way of checking the validity of each function as it gets called. Browser detection leverages the navigator object that is available to all JavaScript code, and while it had its glory days, object detection is now the more popular way to sniff out incompatibility. First, we'll go over browser detection.

3. Browser Detection

Browser detection was widely popular in the past. JavaScript gets a "navigator" object, from which browser data can be collected.

Useful properties:

Property	Description
Navigator.appName	Name of the browser (for ex, Microsoft Internet Explorer
Navigator.appVersion	Version of the browser (for ex, 6.0)
Navigator.userAgent	User agent header (for ex, Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Hotbar 3.0))

For example,

```
<script language="JavaScript"
type="text/JavaScript">
if(navigator.appName == "Netscape"){
    window.location = "http://www.netscape.com"
}
if(navigator.appName == "Microsoft Internet Explorer"){
    window.location = "http://www.msn.com"
}
window.location == "meh.html"
</script>
```

So, we could use this to set up many variables that detect the browser type and version, and then use this to either detect which version of our code to use, or to phase out parts of our code that might not work under other browsers.

4. Object Detection

So what's the problem with using browser detection? Well, as you can see, it's pretty tedious. Many browsers hide their names or have different formats for displaying version numbers. You can to check for every browser, and since there are so many different kinds, it's almost impossible to preplan for each. Besides, who wants to do that? Here's where object detection comes in. It actually figures out whether the visiting browser supports the object that is going to be used before it uses it.

Let's look at the classic rollover example. Rollovers use the document.images object to get the images on a page. However, this object is only supported by Netscape 3+ and IE 4+.

Here's how to do it:

```
function imgOn(imgName) {
   if (document.images) {
         document[imgName].src = eval(imgName + "ON").src;
   }
}
```

Notice that I called the object, document.images. If it exists, it will return true and execute the following code. Notice that if it does not exist, the code will continue as if nothing had happened.

Cookies – Quick Overview

What are They?

A very small text file placed on your hard drive by a web page server when you access a site on that server. It serves as an identification the next time you access web pages on that server so as to extend the capabilities of web-based client/server applications. It is uniquely yours and can only be read by the server that gave it to you.

What are They For?

To let the server know that you have returned to a page/site you have previously visited. This can be useful in the following situations:

- Suppose you are filling a form and then accidentally close the window. A cookie set on your machine lets the server remember fields already filled-out.
- Allows you to customize what you want to see

when you visit a portal like myway.com
- Monitors which advertisements were shown to you and how often each was shown.

Types of Cookies
There are two types:
- Session cookies – expire after your session on that site
- Persistent cookies – remain until you clear your cache and delete cookies

Types of Values a Cookie can Pass
- The **`name`** of the cookie.
- The **`value`** of the cookie.
- The **`expiration date`** of the cookie - how long the cookie is active in your browser.
- The **`path`** of the cookie is valid for - the URL path the cookie us valid in. Web pages outside of that path cannot use the cookie.
- The **`domain`** the cookie is valid for - makes the cookie accessible to pages on any of the servers on a multi-server domain.
- The need for a **`secure`** connection – states if the cookie can only be used on a secure server connection, such as on a site using SSL.

Let's Bake some Cookies

Stage 1 – Cascading Style-Sheets

Task 1
- Visit CSS-Zen Garden <http://www.csszengarden.com/>
- Click on any of the links to the left and observe the content of the pages

As you might have observed, although the index pages all have the same text, their formatting and layouts differ radically. Each uses a Cascading Style-Sheet (CSS).

Questions:
- What is a style-sheet?
- Do you find all the layouts equally appealing?
- What is the chance that everyone finds your favorite one most appealing?

Stage 2 – Creating the Style-Sheets
Lets now apply the knowledge about cookies to allow each one of us to be able select how they want their site to be displayed. But first we need to have a style-sheet created for each new look we want to make available.

In real life you would do this when you start designing your site. If you want to learn more about style-sheets, use Google, visit WC3.org (http://www.w3.org/Style/CSS/) or come to our Spring CSS tutorial.

For now:

Task 2

- Download and unzip **`cookies.zip`** to the desktop
- Open the index file for editing in Macromedia DreamWeaver.

Stage 3 – Attaching the Style-Sheets to the Page

The **`link`** element is used to declare the style-sheet. There are three different relationships external style sheets can have with their parent document: **`persistent`**, **`preferred`**, and **`alternate`**.

Persistent Sheets

Always on. To make a style sheet **`persistent`**, the **`rel`** attribute is set to "**`stylesheet`**" and NO **`title`** attribute is set.

To make the style sheet "style1.css" persistent, add this code in the head:

```
<link rel="stylesheet" type="text/css" href="style1.css" />
```

Preferred Sheets

These style sheets are enabled by default. They can, however, be disabled them if an alternate style sheet is selected.

To make a style sheet preferred, the **`rel`** attribute is set to "**`stylesheet`**" and the style sheet is given a **title** attribute. For instance, this code makes sheet "style2.css" preferred.

```
<link rel="stylesheet" type="text/css" href="style2.css"
title="preferred"/>
```

Note: It is **`preferred`** not because its title is "**`preferred`**" but because it has a **`title`**.

Style sheets having identical **`title`** attributes are form a group. Grouped sheets can be enabled and disabled in one go. If more than one group of preferred style sheets are declared, the first group takes precedence.

Alternate Sheets

These style sheets can be selected by the visitor as an alternatives to the author's 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. To make style-sheet "style3.css" an alternate sheet, we place this code in the head of the document.

```
<link rel="alternate stylesheet" type="text/css"
href="style2.css" title="alternate" />
```

Task 3

- There are three style-sheets in the cookie folder you created from the zip-file you downloaded.
- Add sheets "**`style1.css`**", "**`style2.css`**" and "**`style3.css`**" to the head of your "**`index.html`**" file
- Make sheet "**`style1.css`**" the **`persistent`** sheet.
- Make sheet "**`style2.css`**" the **`preferred`** sheet entitled "**`sassy`**"
- Make sheet "**`style3.css`**" the **`alternate`** sheet entitled "**`boring`**"

Swapping Styles

Stage 4 – Programmatically Differentiating Sheets

Ideally, users should be able to select the style-sheet of their choice from the available list. Sadly, this is not the case so we have to use JavaScript to allow users to make that choice and remember it for them using a cookie.

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
```

Using these three checks we can check 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: Only preferred and alternate style sheet link elements have a title attribute.

The function to do this looks like this:

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

Stage 5 - Baking Cookies

Now we can change the style sheet. However, the preference only persists on the current page. To remember the preference, we use cookies.

Cookie processing involves storing the preference to the hard-drive in the form of a cookie and reading it another time.

The following snippet creates a cookie.

```
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=/";
}
```

To read the cookie we use this snippet

```
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;
}
```

Stage 6 - Getting the Current Active Style-sheet

To return the current style sheet we locate an active preferred or alternate style sheet and check its title. To do this we look at all link elements on the page to decide if each is a style sheets. If it is, we check if has a title. This tells us that the style sheet is either preferred or alternative.

Finally, we check if the style sheet is active. If all three checks return true, we have the current style sheet and we can return the title. If not, then we return null.

The function looks like this:

```
function getActiveStyleSheet() {
   var i, a, tagNames;
   tagNames = document.getElementsByTagName("link");
   for(i=0; i < tagNames.length; i++) {
      a = tagNames[i];

       if(a.getAttribute("rel").indexOf("style") != -1
            && a.getAttribute("title")
            && !a.disabled) return a.getAttribute("title");
      }
      return null;
}
```

Stage 5 - Using the Cookies

To use these cookie functions, we need to add onload() and onUnload() event listener handles to the window.

`OnLoad()`

When document is loaded, we need to tell the browser which style-sheet (our preferred style-sheet) we want to use as our active style-sheet. The function **`setActiveStyleSheet()`** (see page4) does this.

To find out which style sheet is the preferred style sheet, we need another function - **`getActiveStyleSheet()`** which may look like this:

```
function getPreferredStyleSheet() {
   var i, a, tagNames;
   tagNames = document.getElementsByTagName("link");
   for(i=0; i < tagNames.length; i++) {
      a = tagNames[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 was retrieved from the cookie, or if there isn't one, the title of the author's 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);
}
```

onUnload()

When we leave the page, we want to remember the style-sheet for another day, so we save it. We use the **onUnload** listener message handle to do this. This function gets the current active style-sheet, and saves it in a *cookie*:

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

Stage 6 - Integrating it all

We store all these functions in a JavaScript file named **styleswitcher.js**, and load the file inside the head of the document, making sure the script file is loaded after all the style-sheet link elements like this:

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

To allow visitors to change the active style sheet, we use hyperlinks with a JavaScript **onClick()** events. For example, to have the option to switch between two themes with titles “boring” and “sassy,” we would use the following HTML:

```
<a href="#"
     onclick="setActiveStyleSheet('boring');
     return false;">
     Boring Theme
</a>

<a href="#"
     onclick="setActiveStyleSheet('sassy');
     return false;">
     Sassy Theme
</a>
```

Task 4

- Attach the “**style-switcher.js**” file to the “**index.html**” file
- Add the hyperlinks that allow the user to switch style sheets

Resources Used

Cookie-Central < *http://www.cookiecentral.com/faq/>*
Webopedia.Com < *http://www.webopedia.com/DidYouKnow/Internet/2002/Cookies.asp>*
Adapted from Sowden, Paul. “Alternative Style: Working With Alternate Style Sheets.”
< *http://www.alistapart.com/articles/alternate/>*