LETTING THE WEB APP USER "THEME" YOUR UI

An Introduction to Dynamic Theming using jQuery UI

by Jeremy Fonte

INTRODUCTION

The techniques discussed in this article stem from my work on a "Web Audio Player", that is, a functional prototype for an HTML5 audio streaming site. In addition to making the prototype play songs, I wanted to give the end-user a fun, easy way to alter the UI to fit their personal sense of style and aesthetics.

Basically, the audio player site used jQuery UI's CDNs to load the selected CSS themes at run-time. It also altered the background image on the site, to match the colors of the selected theme. This article assumes you have some knowledge of jQuery and are familiar with general HTML/CSS/JavaScript development techniques and terminology.

If you'd like to check out a working demo of a user-themeable UI, go to http://fonte.me/jeremyfonte/labs/projects/webaudioplayer/default.html. By clicking on the theme button in the bottom right and selecting a theme, you can alter the look and feel of the site.

THE FIRST STEPS TOWARDS "THEMEABILITY"

Before you bother writing any code, get familiar with jQuery UI and the various themes they have available. Spending some time browsing the demos and themes at http://jqueryui.com will help with designing your user-themeable website; any non-jQuery UI components will have to be altered for each theme to maintain a consistent color scheme and look and feel. Also, background images and icons may need to be altered for each possible theme.

Once you're comfortable with the details of jQuery UI, go to their blog and find the post for the most recent final version of the library. There will be links to the CDN locations of the various themes; these links are what make the magic of user-themeable web apps so easy to attain.

When the user selects a new theme, you will swap the current theme's CSS file in the <head> section of the page for the new theme's CSS file, and also tweak any peripheral styles, such as background images, to match the new theme.

THE CODE!

What follows is a sequence of code excerpts and explanations from the JavaScript code I used to make my "Web Audio Player" have a user-themeable UI.

```
$(document).ready(function() {

        refreshDialogs('Default');

        function refreshDialogs(customClass) {
                $('#mainWindow').hide();
```

```
                $('#themeDialog').hide();

                $('#mainWindow').dialog('destroy');
                $('#themeDialog').dialog('destroy');

                $('link[data-windowTheme]').remove();

                var currentThemeHref = new String();
                if(customClass === 'Default') {
                        currentThemeHref =
'http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/themes/ui-lightness/jquery-ui.css';
                }
                if(customClass === 'Dark') {
                        currentThemeHref =
'http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/themes/ui-darkness/jquery-ui.css';
                }
                else if(customClass === 'Underwater') {
                        currentThemeHref =
'http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/themes/cupertino/jquery-ui.css';
                }
                else if(customClass === 'Cartoon') {
                        currentThemeHref =
'http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/themes/hot-sneaks/jquery-ui.css';
                }
                else if(customClass === 'Techno') {
                        currentThemeHref =
'http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/themes/trontastic/jquery-ui.css';
                }
                else if(customClass === 'Nature') {
                        currentThemeHref =
'http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/themes/le-frog/jquery-ui.css';
                }
                else if(customClass === '7') {
                        currentThemeHref =
'http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.18/themes/cupertino/jquery-ui.css';
                }
                else if(customClass === '8') {
                        currentThemeHref =
'http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.18/themes/cupertino/jquery-ui.css';
                }
                var currentTheme = document.createElement('link');
                currentTheme.setAttribute('type', 'text/css');
                currentTheme.setAttribute('rel', 'stylesheet');
                currentTheme.setAttribute('href', currentThemeHref);
                currentTheme.setAttribute('data-windowTheme', customClass);

                document.getElementsByTagName('head')[0].appendChild(currentTheme);

                initWindows(customClass);
                $('body').css('display', 'block');
                $('#mainWindow').dialog('open');
        }
```

The code sample above takes care of initializing the theme on the second line, by calling the dialog theming function and providing the text “Default” as the parameter (for which theme to apply). It is important to note that in the initial HTML code, there are no jQuery UI CSS themes (<link> elements); the JavaScript grabs the theme from jQuery’s CDN when the page is initially loading, as well as each time the user decides to change the theme.

The “refreshDialogs” function hides and then destroys the dialog boxes that make up the UI. It then removes all <link> elements with the data attribute “data-windowTheme”. This ensures that as the user changes the theme, old <link> elements are removed, rather than having them all pile up in the HTML.

A series of if statements (which should be a switch statement) then test the theme name to determine what the “href” attribute of the new theme’s <link> element should point to; the jQuery UI

CDN is used for this, rather than local copies of the various themes' CSS files and images. A new <link> element is then created with the proper attributes, including the "data-windowTheme" attribute.

The <head> element is then grabbed using DOM functions, and the new <link> element is appended to it. Eventually, the dialog (which holds the user interface, in this case) will be opened – but first, the following function is called:

```
function initWindows(customClass) {
	var bgImage = new String();
	var bgColor = new String();

	switch(customClass) {
		case 'Underwater':
			bgImage = 'url("images/web_audio_underwater_bg.jpg")';
			bgColor = 'transparent';
			break;
		case 'Cartoon':
			bgImage = 'url("images/web_audio_cartoon_bg.jpg")';
			bgColor = 'transparent';
			break;
		case 'Dark':
			bgImage = 'url("images/web_audio_dark_bg.jpg")';
			bgColor = 'transparent';
			break;
		case 'Techno':
			bgImage = 'url("images/web_audio_techno_bg.jpg")';
			bgColor = 'transparent';
			break;
		case 'Nature':
			bgImage = 'url("images/web_audio_nature_bg.jpg")';
			bgColor = 'transparent';
			break;
		case 'Default':
			bgImage = 'none';
			bgColor = 'black';
			break;
	}

	$(document.body).css({
			'background-image': bgImage,
			'background-repeat': 'repeat',
			'background-color': bgColor
	});
```

Let's discuss this part of the function before looking at the rest of it. It is fairly straightforward – the name of the theme is passed in to the function as the variable "customClass". A switch statement then determines which theme it is, and sets the values of the background image and the background color (only one of them is used at a time – having both a background image and a background color is unnecessary). It is important to note that the unused background attribute, either image or color, is "turned off" – that is, setting the image to "none" or the color to "transparent". This cleans up after the previous theme, to make sure there isn't a conflict between a prior theme's settings and the new theme's settings.

At the bottom of this segment, the background image, repeat, and color are all set, based on the current theme. Let's examine the rest of this function now:

```
	$('#themeDialog').dialog({
		autoOpen: false,
		title: 'Choose a Color Theme',
		show: 'fade',
```

```
                hide: 'fade',
                width: 300,
                height: 200,
                minWidth: 300,
                minHeight: 200,
                maxWidth: 300,
                maxHeight: 200,
                resizable: false,
                draggable: true,
                zIndex: 10000,
                buttons: {
                        "Change Theme": function() {
                                var userThemeChoice = $('#themeChoice').val();
                                refreshDialogs(userThemeChoice);
                        }
                }
        });

        var thisWindowDim = getWindowSize();

        function getWindowSize() {
                var windowWidth = $(window).width();
                var windowHeight = $(window).height();
                var windowDim = new Object();

                windowDim.width = windowWidth;
                windowDim.height = windowHeight;

                return windowDim;
        }

        $('#mainWindow').dialog({
                autoOpen: false,
                title: '<span class="FonteLabs">Fonte Labs</span> - Web Audio API
Demo. All music &copy; 2012 Jeremy Fonte.  All rights reserved.',
                show: 'fade',
                hide: 'fade',
                position: ['center','top'],
                width: 960,
                height: 600,
                minWidth: 960,
                minHeight: 600,
                maxWidth: thisWindowDim.width,
                maxHeight: thisWindowDim.height,
                close: function() {
                        returnToLabs();
                },
                buttons: {
                        "Theme": function() {
                                $('#themeDialog').dialog('open');
                        }
                }
        });

        addAudioButtons();
}
```

First, the theme selection dialog is initialized. It is not set to open automatically on page load; the user has to click on the "Theme" button on the main dialog if they want to change the look and feel of the page.

A small nested function returns the width and height of the browser window. This info is then passed on to the function that creates and opens the main dialog in the web app. The main dialog box is then opened and the user is ready to enjoy the web app!

APPLYING THIS IDEA TO YOUR OWN APP

I hope you've found this tutorial useful. I'm using jQuery UI in my own web app, but the concepts presented here would work equally well with Dojo's Dijit UI library, or even your own components and styles. It all revolves around the idea of swapping one CSS file in the <head> of the page for a new one. You could also locally host your CSS files, rather than using a CDN to load the styles.

Furthermore, if you're using your own CSS styles, you wouldn't need to programmatically set the background images and background colors on the page; you could simply attach a CSS class to the <body> element, much like Dojo does when you're using their Dijit library's themes.