

Oomph: A Microformat Toolkit

Contents

Introduction: What Is This Thing Anyway?	1
Oomph2	2
Consuming Microformats	3
Introduction	3
User Features	3
Aggregation of Microformats	3
Exporting Data	4
Mapping Locations with Virtual Earth	4
Adding Oomph To Your Website	4
Miscellaneous Notes	5
Modifying Oomph	5
Extending Oomph	5
Notes on the Oomph Microformat Parser	6
Notes on the Oomph Javascript	6
Notes on the Oomph Internet Explorer Add In	7
Adding Microformatted Content To Your Website	7
Styling Microformats	8
Acknowledgements	9
Appendix	9
Extending Oomph: A Detailed Example	9

Introduction: What Is This Thing Anyway?

You are browsing the web. You see a contact you'd like to add to your address book (maybe Outlook, maybe your Yahoo address book). Wouldn't it be nice if adding that contact were as simple as the click of button? Or, imagine a concert that you discover on a webpage. Wouldn't it be nice if adding that event to your calendar (again, maybe Outlook, maybe your Windows Live Calendar) were easy?

This is the promise of Microformats. As defined on the Microformats.org website, "Designed for humans first and machines second, Microformats are a set of simple, open data formats built upon existing and widely adopted standards." Microformats straddle the unstructured chaos that is HTML and the vision of the "semantic web". They are simple, practical, and perhaps most interestingly, they are gaining traction among the top website properties.

Designed to help promote the creation and consumption of Microformats, *Oomph: A Microformat Toolkit* (referred to as *Oomph* moving forward) is composed of three facets. First is a tool that aggregates the various contacts and events on the page and provides an overlay panel from which visitors to your site can easily add events and contacts to calendars and contact-stores. The second facet is a library of Microformats styles designed using CSS and ready to be used. Lastly, *Oomph* provides a Windows Live Writer plug-in, designed to make producing your own Microformats easier. In short, *Oomph* makes using and producing content for the web just a little easier, more efficient, more fun.

Oomph2

Oomph2 was released in August 2009 and provides several updates to Oomph:

- **Complete Implementation of the Value Class Pattern.** The Value Class Pattern is an important update to the microformats specifications, which addresses significant issues around accessibility and internationalization. Oomph now successfully parses and renders hCalendars and hCards marked up according to this specification.
- **Support for hMedia.** Oomph will now parse and render audio, video and image content marked up with the hMedia microformat. In the case of audio and video, Oomph2 provides an inline media player for experiencing that content
- **ASP.NET Microformats Control.** With the ASP.NET Microformats control, you can easily add properly formatted microformats to your ASP.NET pages. This control is particularly handy to use when programmatically creating microformatted content, aka populating pages with contacts from a database
- **User Interface Enhancements.** Oomph2 has been redesigned to use its screen real estate more effectively as well as surfacing map information more directly.
- **Bug Fixes and Code Refactoring.** Various issues reported by users around the first version of Oomph have been addressed. Some code was refactored. In addition, Oomph2 uses jQuery 1.3.2.

Consuming Microformats

Introduction

The most ambitious aspect of Oomph is the Internet Explorer Add On and accompanying Javascript/CSS which is injected into pages that contain Microformats. There are two intentions for this. The first intention is for users who install the Internet Explorer Add-on. When installed, pages with Microformats will light up, displaying a Microformat gleam with options for the user.

For those that don't use IE, a web author can easily add the javascript/CSS to their own website, providing the same experience to all browsers on all platforms. Thus web authors can mark up content (hCards and hCalendars) according to the Microformat specifications, add one line of Javascript and web pages will light up with the Microformat gleam no matter what browser is used. You can see an example of this on the page <http://visitmix.com/labs/oomph/2.0/client/style/firstrun.html>.

User Features

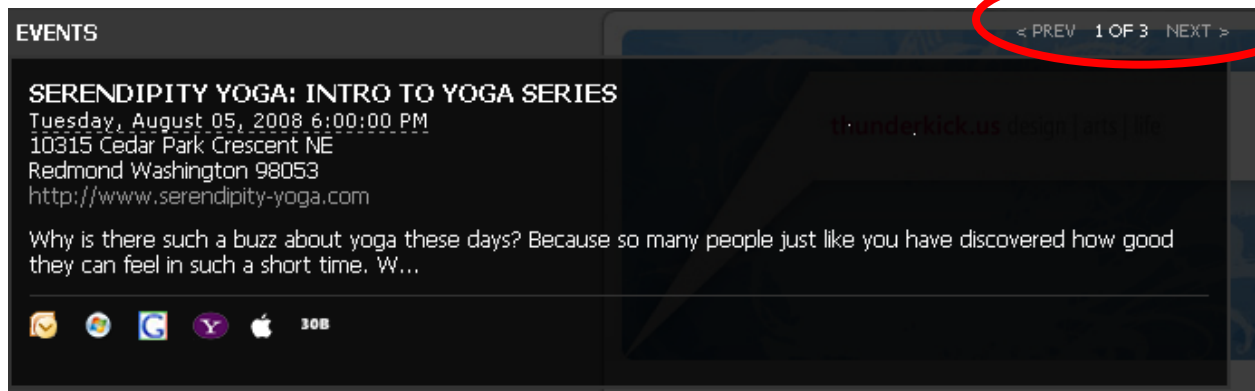
There are three main features that are part of the Oomph gleam and panel: aggregation of Microformats, exporting of data and mapping of data using Microsoft Virtual Earth. The Microformat gleam indicates to the user that there are Microformats on the page:



When the gleam is hovered over, a bar extends out. If the user clicks the first icon, all the Microformats will be aggregated with options export their data.

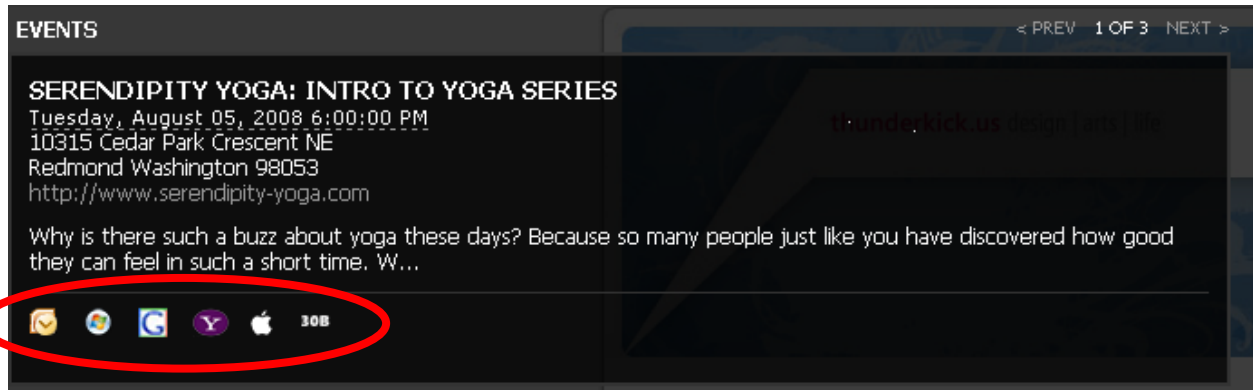
Aggregation of Microformats

When a page has Microformats of the hCard and hCalendar variety, the panel will aggregate all of the Microformats, so that the user can quickly page through the different Microformats and act on them. In the screenshot below, you can see how there are 3 hCalendar events on the page and the tool has provided paging to toggle through each Microformat:



Exporting Data

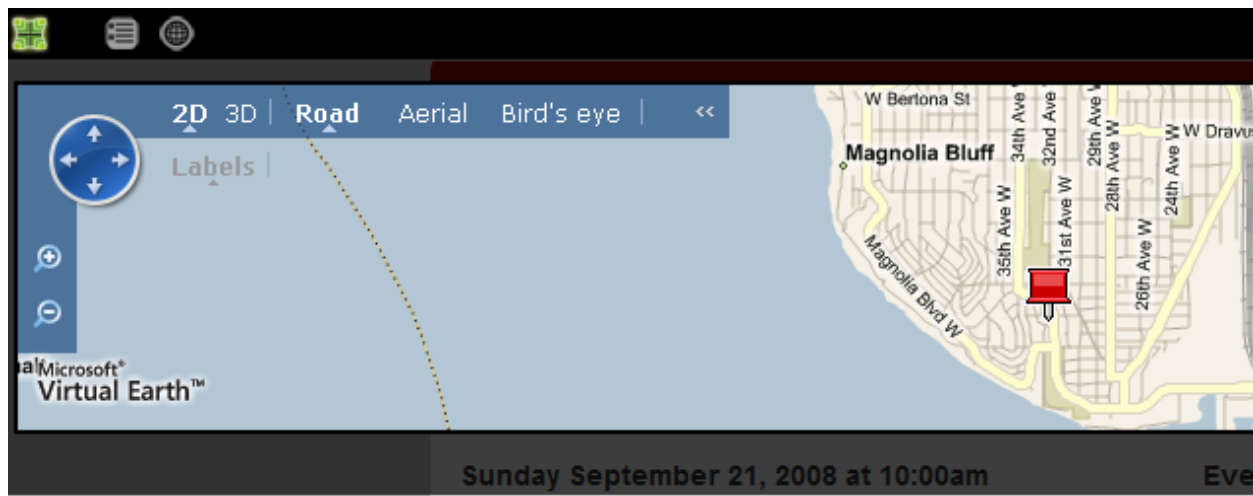
One of the promises of Microformats is the ability to make data available to other websites like Windows Live or Yahoo and/or the local machine. *Oomph* supports this for both hCards and hCalendars through the indication of icons below each format. The Outlook and Apple icons will allow a user to save the hCard or hCalendar to their local machine. The other icons will export the Microformat to a cloud service.



Mapping Locations with Virtual Earth

Many times, a Microformat will contain address information. If this information is available, Oomph will map it using Virtual Earth.

Then, the user will see a map with the various hCards that provided location information:



Adding Oomph To Your Website

As a web author, you are encouraged to add *Oomph* to your website pages that use Microformats. Adding *Oomph* is simple and requires the addition of two javascript files to your webpage:

```
<script type="text/javascript"
src="https://visitmix.com/labs/oomph/2.0/Client/jquery-
1.3.2.min.js"></script>
```

```
<script type="text/javascript"
src="https://visitmix.com/labs/oomph/2.0/Client/oomph.min.js"></script
>
```

Of course, there is no point in adding Oomph unless you have content on your website that is marked up with Microformat syntax of the hCard, hCalendar or hMedia variety. See <http://microformats.org> on how to create content for your website that is Microformat compliant. Another way to easily add Microformat content to your website is to use the Windows Live Writer plug-ins discussed below.

Miscellaneous Notes

- If you are already using jQuery on your website, there is no need to add the jQuery library.
- If you are already using Virtual Earth on your website, you should modify oomph.js to not pull in the Virtual Earth library.
- In the event that a user navigates to your page and has the *Oomph Add In* installed already, the Add In will discover that you have added Oomph and will not inject the script.
- If you like the functionality that Oomph provides but don't like the User Interface, you are more than welcome to modify the javascript and/or CSS. Going about doing this is explained in the section "Modifying Oomph."
- The code has no server dependencies and is completely client-side. The code does call out to a service hosted at VisitMix.com that converts hCards and hCalendars into vCards and iCals, simply passing values in a querystring and returning the vCard or iCal.

Modifying Oomph

For web authors who add *Oomph* to their website, there is an opportunity to customize the look and feel of *Oomph*.

All of the CSS for *Oomph* is loaded within the oomph.js file in the method called insertCSS(). The easiest way to override that CSS is to modify the CSS, upload it to your own server and then change the link in the insertCSS() method to point to your own CSS.

Much of *Oomph* is HTML that is generated dynamically. As such, if you want to do more dramatic modifications beyond just CSS changes, you likely will need to dig into oomph.js itself in order to change the actual HTML that gets generated. All of the HTML is generated using jQuery syntax.

Of course, if you modify Oomph, you will no longer load Oomph from the visitmix server but rather host your own copy of the javascript and css on your server.

Extending Oomph

The three Microformats that *Oomph* has tackled are hCard, hCalendar and hMedia. There are many other Microformats with new ones emerging all the time. Oomph has been designed so that it can be extended to incorporate other Microformats.

Extending Oomph does mean writing Javascript that is in tune with the *Oomph* Javascript. At this time, *Oomph* does not support a plug-in model, but rather requires direct modification of oomph.js as well as direct modification of some CSS. However, the modifications aren't too invasive. See the appendix for a complete example of how to extend Oomph.

Notes on the Oomph Microformat Parser

The *Oomph* Microformat parser is best described as a “pragmatic” parser, in that the parser is tightly coupled to the application that consumes the parsed Microformat. Oomph is not intended to be a generic parser available for reuse in other projects that consume Microformats. Additionally, the parser makes some assumptions and decisions, thus there are cases where a Microformat will not be entirely parsed. Here are the known cases which the parser does not handle:

- Multiple Addresses in an hCard. The parser will only parse the first address.
- Multiple Categories in an hCard. The parser will only parse the first category.
- Agent, Key, Logo, Sort-String, Sound. The parser does not parse Agent, Key, Logo, Sort-String, Sound.

A future version of *Oomph* may support these other features. Because *Oomph* is available for the community to work on, it is hoped that perhaps the community will take on completing the parser.

Note that as of Oomph2, the Oomph parser is compliant with the Microformats Value Class Pattern.

Notes on the Oomph Javascript

The *Oomph* javascript relies on the jQuery library. In order to avoid namespace collisions, the very first thing that the script does is to establish a unique namespace for the oomph code, so as to prevent collisions with other libraries, both jQuery and others:

```
var oomph = {};  
oomph.query = jQuery.noConflict(false);
```

Next, the script searches the page for Microformats, aborting if no formats are found:

```
var hCardCollection = oomph.query('.vcard');  
var hCalendarCollection = oomph.query('.vevent');  
var microformatTotal = hCardCollection.length +  
hCalendarCollection.length;  
//no point in continuing if there are no Microformats  
if (microformatTotal == 0)  
return;
```

Then, a series of global variables are established as well as the HTML generated for the user interface. Lastly, there are a series of functions, which are documented in the code. It is worth calling out that the code was written as script, not as object oriented Javascript code. This was intentional on the part of the author, so as to make the code as readable and simple as possible.

Thanks to Michael Kaply for allowing the use of his ISO8601 date parsing code. Also, thanks to James Edwards for his function to deal with unqualified hrefs in javascript.

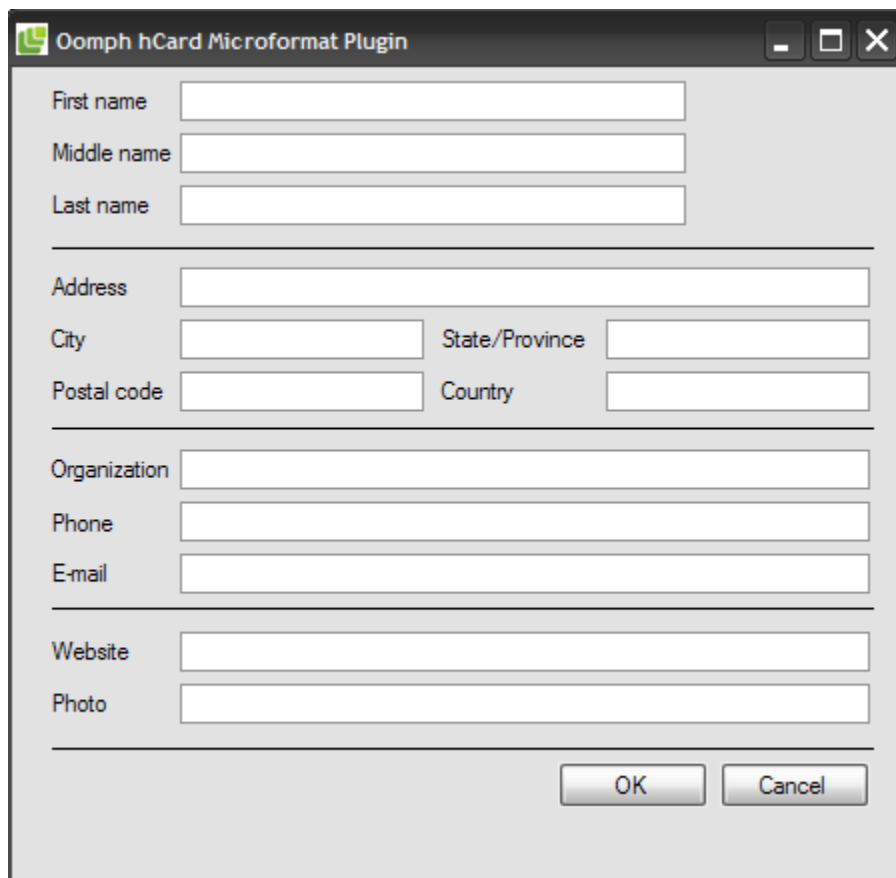
Notes on the Oomph Internet Explorer Add In

One known issue with the Add In is that, because it injects code which displays UI, it has the potential to conflict with the existing user interface. Oomph was designed to be as unobtrusive as possible, but there are cases where it may be obtrusive. When a web author has explicitly added Oomph to their website, the best remedy for this situation is to modify where the gleam is placed. This can be done by modifying the CSS for the root tag of ID #iwmf. Currently, its position is fixed at 0,0. This of course can be changed.

Adding Microformatted Content To Your Website

For Microformats to gain adoption, they need to be easy to add to a website. This is the purpose of the Windows Live Writer Plug In for inserting hCards into blog posts.

Here's a screenshot of what it looks like:



The screenshot shows a dialog box titled "Oomph hCard Microformat Plugin". It contains several text input fields for user information, organized into sections separated by horizontal lines. The fields are: First name, Middle name, Last name, Address, City, State/Province, Postal code, Country, Organization, Phone, E-mail, Website, and Photo. At the bottom right, there are "OK" and "Cancel" buttons.

Once you insert the content into your blog post, all of the tags will be properly marked according to the Microformat specification for usage.

There are several other Windows Live Writer Plug Ins for Microformats that are worth noting:

- Event Plug-in: <http://gallery.live.com/livitemDetail.aspx?li=9751e563-1408-4fc3-8028-bd4351edb1fb&bt=9&pl=8> This plug in supports adding hCalendars to your site.
- Insert Geo Microformat: <http://gallery.live.com/livitemDetail.aspx?li=6e57ab6f-aab9-439b-b578-0ea3b1d95d36&bt=9>
- XFN: This Codeplex project contains plug in for doing XFN.
<http://www.codeplex.com/WLWPlugins>

Another way Oomph helps you get your content formatted as Microformats is through the ASP.NET control provided by Oomph. That project has a sample project attached with two code samples, one showing how to use the control in ASP.NET markup and other showing how to programmatically use the control.

Styling Microformats

Part of the *Oomph* project is to provide a series of CSS styles for hCard and hCalendar Microformatted content. These styles can be used by web designers to easily enhance the look of Microformatted content on their site. In the Style directory, you will find an .HTML page called switcher.html. This page allows you to quickly toggle between the different CSS styles.



There are also two other HTML pages in the Styles directory, hCard.html and hCalendar.html. These pages provide commented HTML that demonstrate how the CSS styles were implemented.

The easiest way to implement one of the styles on your website is to simply copy the oomph.css directory up to your website and reference the css. That way, the paths to the images will all work.

Acknowledgements

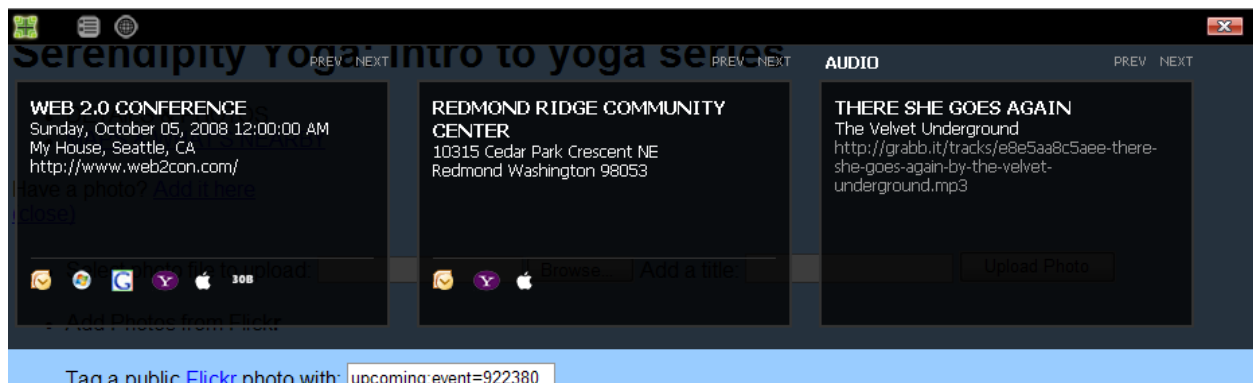
We would like to thank a whole range of people who have inspired and helped the project get off the ground:

- Michael Kaply and his Operator Add-in for Firefox, which was inspirational
- John Allsopp, whose book on Microformats was essential reading for the whole team
- All of the folks part of the Microformats community – especially those on the discuss alias -- whose passion for Microformats kept us invigorated
- The folks who wrote jQuery

Appendix

Extending Oomph: A Detailed Example

In the following example (which will be used for the duration of this documentation), the hAudio format (<http://microformats.org/wiki/haudio>) is being added. The end result looks like this:



1. Create a variable to hold the collection of Microformats. Then, create a search for all the root class names represented by that Microformat.

```
var hAudioCollection = $('.haudio');
```

2. Add this collection to the count. The code aborts if there are no Microformats to display.

```
var microformatTotal = hCardCollection.length +  
hCalendarCollection.length + hAudioCollection.length;
```

3. Add a counter variable and three variables for the UI containers. The convention used is to preface all CSS classes with iwmf.

```

var currentHAudio = 1;
var mainhAudioContainer = $('<div id="iwmf_mainhAudioContainer"
></div>');
var hAudioHead = $('<div id="iwmf_hAudioHead" ></div>');
var hAudios = $('<ul id="iwmf_hAudios" ></ul>');

```

Note you'll need to add these to the CSS; we'll get to that in a moment.

4. Wire paging. You'll need to add some code to the wirePaging() function. You'll need to create a variable for the next and previous divs and then just replace the counter with your Microformat counter:

```

        if (hAudioCollection.length > 1){

            var audioNext = $('<div
class="iwmf_Next"><a href="#"
>Next</a></div>').bind('click',function(event)

                {

                    currentHAudio = currentHAudio +

1;

                    if (currentHAudio <=

$('<div id="iwmf_hAudioContainer">').size()){

                        $('<div id="iwmf_hAudioContainer">').hide(animationSpeed);

                        var current =

$('<div id="iwmf_hAudioContainer">')[currentHAudio - 1];

                        $(current).show(animationSpeed);

                    }

                    else

                        currentHAudio =

currentHAudio - 1;

                });

            var audioPrev = $('<div
class="iwmf_Prev"><a
href="#">Prev</a></div>').bind('click',function(event)

```

```

        {
            currentHAudio = currentHAudio -
1;

            if (currentHAudio >= 1){

                $('iwmf_hAudioContainer').hide(animationSpeed);

                var current =
$('iwmf_hAudioContainer')[currentHAudio - 1];

                $(current).show(animationSpeed);

            }

            else

                currentHAudio = 1;

        });

        $(mainhAudioContainer).append(audioNext);
        oomph.query(mainhCardContainer).append('<div
id="iwmf_hCardNum" class="iwmf_Prev">' + currentHAudio + ' of ' +
hCardCollection.length + '</div>');

        $(mainhAudioContainer).append(audioPrev);

    }

```

5. Modify displayUI() function, adding the three UI containers that you created earlier.

```

$(mainContainer).append(mainhAudioContainer);
$(mainhAudioContainer).append(hAudioHead);
$(mainhAudioContainer).append(hAudios);

```

6. Add a call to displayUI() to a new method that finds and parses your new format:

```

findAndParseHAudio();

```

7. Write the findAndParse method for your new Microformat.

First, add a count variable to keep track of paging.

```

var hAudioCount = 0;

```

Then, walk the Microformats, as follows:

```
$(hAudioCollection).each(function() {  
    var hAudio = $(this);  
});
```

Within that code, parse the Microformat, extracting the values. JQuery has handy syntax for doing extraction. For example, here's code that extracts the title. You may have to do some other syntax for getting some values out.

```
var title = hAudio.find('.title').text();
```

Create a container for your Microformat and handle the logic for hiding all but the first one:

```
if (hAudioCount == 0)  
    var hAudioContainer = $('<li  
class="iwmf_hAudioContainer"></li>').show();  
else  
    var hAudioContainer = $('<li  
class="iwmf_hAudioContainer"></li>').hide();
```

Add the parsed values to your UI. Reuse .CSS classes as appropriate. You'll need to add new .CSS classes if you can't reuse other ones.

```
$(hAudioUI).append('<div class="iwmf_fn">' + title + '</div>');  
$(hAudioUI).append('<div class="iwmf_fn">' + album + '</div>');  
$(hAudioUI).append('<div>' + contributor + '</div>');
```

Now append your containers to the main UI and increment your counter:

```
$(hAudioContainer).append(hAudioUI);  
$(hAudios).append(hAudioContainer);  
hAudioCount++;
```

And you are done with the javascript!

8. On to the CSS. Create a new class for your css or else add the following classes to an existing css file.

Be sure to change the location where the css is downloaded in the javascript in the insertCSS() function. For this example, I created a new css file and then put a link to that file in the html.

```
#iwmf #iwmf_hAudios
{
    width:auto;
    margin:0 10px 10px 10px;
    padding:10px;
    border:#333333 solid 2px;
    background-image:url('images/50x50_95.png');
    background-repeat:repeat;
    min-height:160px;
}
```

```
#iwmf #iwmf_mainhAudioContainer
{
    float:left;
    display:inline;
    width:32%;
}
```

```
#iwmf #iwmf_hAudioHead
{
    height:15px;
    margin: 0 0 5px 15px;
    font-size:1em;
    font-weight:bold;
    text-transform:uppercase;
}
```

Then, change the following CSS width % so that your new module will fit in the real estate. In this case, there are three containers, so I have things set at 32%. If you were to add a fourth it would be 24%, and a 5th 19%.

```
#iwmf #iwmf_mainEventContainer
{
    float:left;
    display:inline;
    width:32%;
}

#iwmf #iwmf_mainhCardContainer
{
    float:left;
    display:inline;
    width:32%;
}
```

An example of this can be seen in oomphx.js, which is implemented in hAudio.html.