

# Item Renderers in Flex

**Josh Tynjala**

**Flash and Flex Engineer, Yahoo!**

**360Flex Atlanta**

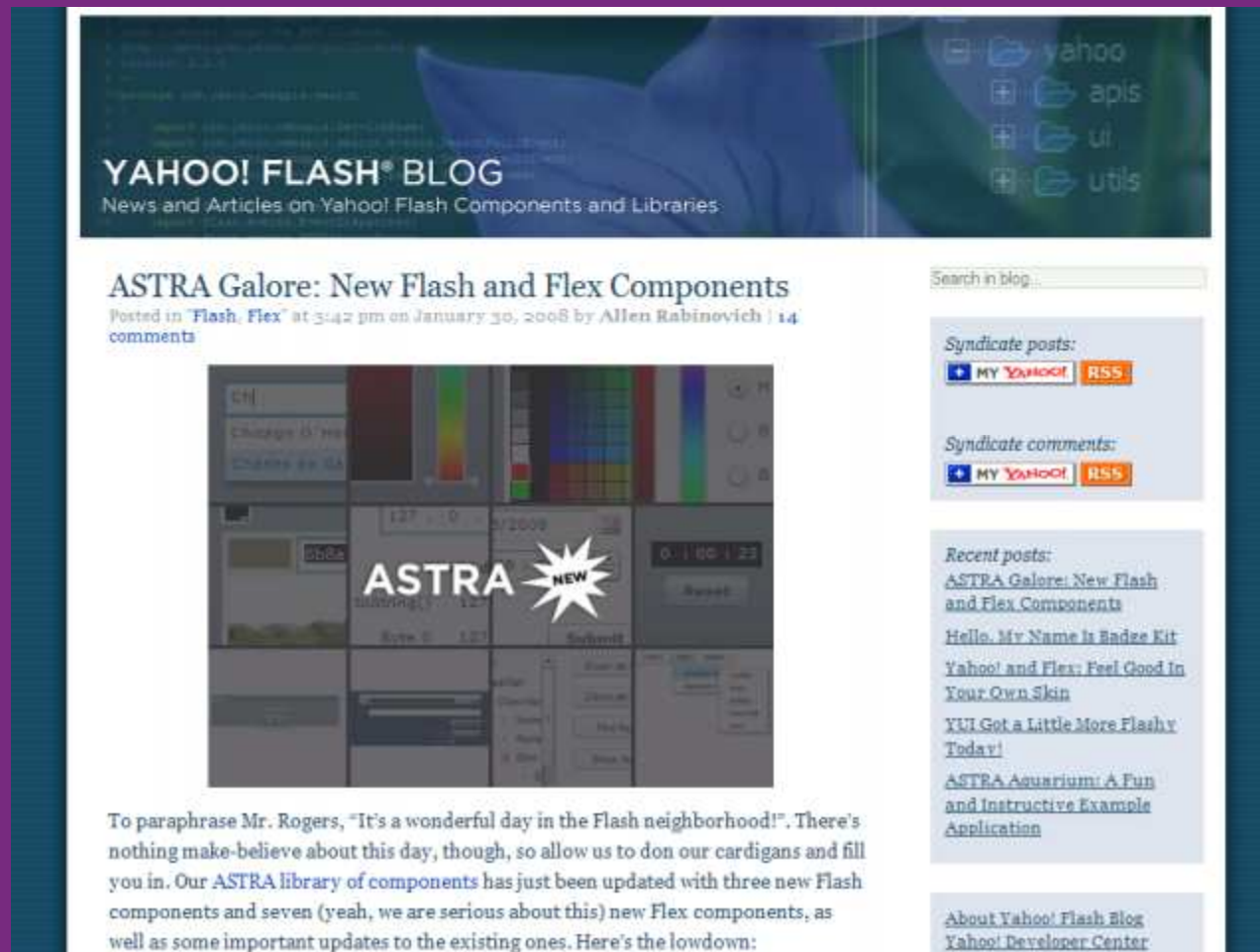
# Yahoo! Flash Platform

The screenshot shows the Yahoo! Developer Network Flash Developer Center. The page has a purple header with the text "YAHOO! DEVELOPER NETWORK" and "FLASH DEVELOPER CENTER". The main content area is divided into several sections:

- Welcome Flash Developers:** A text block explaining the center's purpose and providing links to open source tools and the ydn-flash Yahoo! Group.
- ASTRA: ActionScript Toolkit for Rich Applications:** A section featuring a video player showing an aquarium application. Text describes ASTRA as a collection of Flash and Flex components, code libraries, toolkits, and utilities developed by Yahoo! for ActionScript developers. A "DOWNLOAD" button is visible.
- Flash Components:** A section listing various components and resources. Components include Tree, Menu, TabBar, AutoComplete, Charts, AlertManager, AudioPlayback, and MenuBar. Resources include Overview, API Documentation, and License.
- Flex Components:** A section describing ASTRA Flex components for use in Adobe Flex 3 applications, noting they are fully documented and come with multiple examples.
- Featured Application: Aquarium:** A section with a video player showing an aquarium application. Text describes it as a fun application using ASTRA components.
- Flash Theater:** A section with a video player showing a screencast on ASTRA components.
- Articles:** A section for news and updates.
- Yahoo! Flash Blog:** A section with a list of recent blog posts, including "ASTRA Galop: New Flash and Flex Components", "Hello, My Name is Badge Kit", "Yahoo! and Flex: Feel Good in Your Own Skin", "YUI Got a Little More Flashy Today!", "ASTRA Aquarium: A Fun and Instructive Example Application", and "YDN-Flash Yahoo! Group".

<http://developer.yahoo.com/flash/>

# Yahoo! Flash Blog



The screenshot shows the Yahoo! Flash Blog interface. At the top, there's a navigation menu with links for 'yahoo', 'apis', 'ui', and 'utils'. The main header reads 'YAHOO! FLASH® BLOG' with the subtitle 'News and Articles on Yahoo! Flash Components and Libraries'. The featured article is 'ASTRA Galore: New Flash and Flex Components', posted in the 'Flash, Flex' category at 3:42 pm on January 30, 2008, by Allen Rabinovich, with 14 comments. The article's thumbnail image displays a grid of UI components, with the word 'ASTRA' and a 'NEW' starburst prominently featured. To the right of the article, there are sections for 'Syndicate posts:' and 'Syndicate comments:', each with 'MY YAHOO!' and 'RSS' buttons. Below these are 'Recent posts:' and 'About Yahoo! Flash Blog' sections, each containing a list of links to other blog entries.

**YAHOO! FLASH® BLOG**  
News and Articles on Yahoo! Flash Components and Libraries

## ASTRA Galore: New Flash and Flex Components

Posted in [Flash, Flex](#) at 3:42 pm on January 30, 2008 by Allen Rabinovich | 14 comments



To paraphrase Mr. Rogers, "It's a wonderful day in the Flash neighborhood!". There's nothing make-believe about this day, though, so allow us to don our cardigans and fill you in. Our [ASTRA library of components](#) has just been updated with three new Flash components and seven (yeah, we are serious about this) new Flex components, as well as some important updates to the existing ones. Here's the lowdown:

**Syndicate posts:**  
[MY YAHOO!](#) [RSS](#)

**Syndicate comments:**  
[MY YAHOO!](#) [RSS](#)

**Recent posts:**  
[ASTRA Galore: New Flash and Flex Components](#)  
[Hello, My Name is Badge Kit](#)  
[Yahoo! and Flex: Feel Good in Your Own Skin](#)  
[YUI Got a Little More Flashy Today!](#)  
[ASTRA Aquarium: A Fun and Instructive Example Application](#)

**About Yahoo! Flash Blog**  
[Yahoo! Developer Center](#)

<http://www.yswfblog.com/>

# Zeus Labs

The screenshot shows a blog post on the Zeus Labs website. The post title is "Juice Analytics Features Flex TreeMap Component" dated January 31, 2008. The author is Josh Tynjala. The main content describes how Juice Analytics featured a TreeMap component for Flex, which visualizes Internet traffic data by combining treemap, comScore data, and Alexa thumbnails. A treemap visualization is shown, displaying a grid of colored rectangles representing different website categories. The text below the image explains that the treemap shows the most popular website categories based on unique visitors from August to November 2007, and notes that Yahoo! properties are prominent. The post concludes with a disclaimer and a mention of recently released Flex components.

**zeuslabs**  
JOSH TYNJALA EXPLORES FLASH, FLEX, AND ACTIONSCRIPT

### Juice Analytics Features Flex TreeMap Component

JANUARY 31, 2008



Today, I noticed my [TreeMap component for Flex](#) got featured by [Juice Analytics](#) on their blog. They created an application that combines a [treemap with comScore data and Alexa thumbnails](#) to make an interesting little visualization of Internet traffic.



At a high level, the application shows what categories of websites are the most popular online (based on a comparison of unique visitors between August and November 2007). Digging deeper, one can discover other mysteries across the categories. For instance, you might notice that various Yahoo! properties are sprinkled throughout the treemap because, obviously, the company is a major Internet portal that covers a wide-variety of interests. Disclaimer: I [work for Yahoo!](#) and I'm not ashamed to point out cool stuff about my employer. By the way, did you see the [Flex components](#) we released yesterday? 😊

**About the Author**  
WHO THE HELL IS THIS GUY?  
Josh Tynjala is a professional ActionScript Engineer living in Silicon Valley. He works for [Yahoo!](#) where he develops Flash and Flex components.

**Recommendations**  
GOOD FLASH BOOKS THAT I'VE READ

-  [Essential ActionScript 3.0](#) by Colin Moock
-  [ActionScript 3.0 Design Patterns](#) by Sanders and Cumararatunge

Free cell phones all carriers.  
\$3.99/month Hosting by 1&1

**Search**  
CAN I HELP YOU FIND SOMETHING?  
 GO

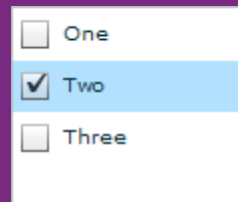
Recent Posts

<http://www.zeuslabs.us/>

# Agenda

- Custom Renderers for Existing Components
- Build an Item Renderer System

# Custom Renderers for Existing Components



# Setting Custom Renderers in MXML

- **Option 1**

```
<mx:List itemRenderer="mx.controls.Label">
```

- **Option 2 (more in a bit)**

```
<mx:List>  
  <mx:itemRenderer>  
    <mx:Component>  
      <mx:Label />  
    </mx:Component>  
  </mx:itemRenderer>  
</mx:List>
```

# Setting Custom Renderers in ActionScript

- `mx.core.IFactory`
- `mx.core.ClassFactory`

```
import mx.controls.Label;
```

```
itemRenderer = new ClassFactory( Label );  
itemRenderer.properties = { minWidth: 400 }
```

```
var instance:IListItemRenderer =  
    itemRenderer.newInstance();
```

# Creating Custom Renderers in MXML

- Uses special `<mx:Component>` tag.
- **Advantages:**
  - No extra classes.
  - Easy MXML layout.
- **Disadvantages:**
  - Can be hard to maintain.
  - Must understand scope.

# MXML Item Renderer Scope

```
<mx:Array id="dp">  
  <mx:String>One</mx:String>  
  <mx:String>Two</mx:String>  
  <mx:String>Three</mx:String>  
</mx:Array>
```

```
<mx>List dataProvider="{dp}">  
  <mx:itemRenderer>
```

```
    <mx:Component>
```

```
      <!-- this is like a new MXML file! -->
```

```
      <mx:CheckBox/>
```

```
    </mx:Component>
```

```
  </mx:itemRenderer>
```

```
</mx>List>
```

## Using `outerDocument` in `<mx:Component/>`

- Refers to scope of MXML outside the renderer.

```
<mx:Component>
  <mx:CheckBox
    change="trace(outerDocument.dp.indexOf(this.data))">
  </mx:CheckBox>
</mx:Component>
```

# Creating Custom Renderers in ActionScript

- **Implement the required interface**
  - IListItemRenderer
  - Used by List, Tree, and DataGrid
- **Optionally, implement the drop-in interface**
  - IDropInListItemRenderer

# Drop-In Renderers

- A common data structure
- Promotes encapsulation
- The standard Flex components are ready for drop-in

# ListData Class

- The common data structure
- Properties
  - columnIndex
  - rowIndex
  - icon
  - label
  - labelField
  - uid
  - owner

# Custom Renderer Examples

- Font List (MXML)
- Color List (MXML)
- CheckBox List (ActionScript)

# Build an Item Renderer System



# Expose Common Renderer Properties

- **Examples from List:**
  - label
  - icon
- **Fields and Functions**
  - Field is property in data provider item.
  - Function is custom procedure to “generate” the value.

# Create Quick Conversion Functions

- **Examples from List**
  - itemToLabel()
  - itemToIcon()
  - itemToDataTip()
  - itemToItemRenderer()
  - itemRendererToIndex()

# itemToLabel() Example

```
public function itemToLabel( item:Object ):String
{
    if(!item) return "";

    if( this.labelFunction != null )
    {
        return this.labelFunction( item );
    }
    else if( item.hasOwnProperty( this.labelField ) )
    {
        return item[ this.labelField ];
    }

    // worse case scenario
    return item.toString();
}
```

# Creating, Removing, and Updating

- Reduce display list manipulations
- Reuse renderers with new data
- Recycling is fun

# Structure of Renderer Creation

```
override protected function commitProperties():void
{
    super.commitProperties();

    // save the current item renderers for reuse.
    this.createCache();

    // reuse or create new renderers to display the data.
    this.renderItems();

    // remove extra cached item renderers we don't need.
    this.clearCache();
}
```

# Renderer Recycling

- *createCache()*
- *renderItems()*
- *clearCache()*

# Creating the Cache

```
protected function createCache():void
{
    // copy the Array using concat() and start fresh
    this._rendererCache = this._itemRenderers.concat();
    this._itemRenderers = [];
}
```

# Renderer Recycling

- `createCache()`
- *`renderItems()`*
- `clearCache()`

# Requesting a Renderer

```
protected function renderItem():void
{
    var iterator:IViewCursor =
        this._dataProvider.createCursor();

    while( !iterator.afterLast )
    {
        // request a renderer. it may be new or from
        // the cache. we don't care.
        var renderer:ItemRenderer = this.getRenderer();
        renderer.data = iterator.current;
        iterator.moveToNext();
    }
}
```

# Using the Cache

```
protected function getRenderer():void
{
    // if there's anything in the cache, use it
    if( this._rendererCache.length > 0 )
    {
        // shift() removes the first item
        // may help to prevent extra redraws
        return this._rendererCache.shift();
    }
    else // otherwise create a new one
    {
        return this._itemRenderer.newInstance();
        // may want to add event handlers too
    }
}
```

# Renderer Recycling

- `createCache()`
- `renderItems()`
- *`clearCache()`*

# Clearing the Cache

```
protected function clearCache():void
{
    var cacheLength:int = this._rendererCache.length;
    for( var i:int = 0; i < cacheLength; i++ )
    {
        // remove the renderer from the cache
        var renderer:ItemRenderer =
            this._rendererCache.pop();

        // then (finally) remove it from the display list
        this.removeChild( renderer );

        // also, remove event handlers, if needed
    }
}
```

# Examples

- Flex TreeMap
- Flash CS3 TabBar (I know, I know...)

# Yahoo! Flash Platform

- **Yahoo! Flash Developer Network:**
  - <http://developer.yahoo.com/flash/>
- **Mailing List:**
  - <http://tech.groups.yahoo.com/groups/ydn-flash/>
- **Blog:**
  - <http://www.yswfblog.com/>