ColdFusion Article

An overview of Ajax features in ColdFusion 8

Brian Szoszorek
BrianSzoszorek.com

Steve Rittler
countermarch.com/blog

Among the many enhancements to Adobe ColdFusion 8 are easy-to-use tools that enable you to quickly create Ajax applications. Ajax (Asynchronous JavaScript and XML) combines a set of technologies that you can use for creating more interactive web applications.

This article gives an overview of using new Ajax features in ColdFusion 8 for application interface layout and data binding. The Ajax features in ColdFusion 8 also help you work with JSON, SPRY, and proxies, but that will be covered in future articles.

Requirements

To complete this tutorial you will need to install the following software and files:

ColdFusion 8

Try (www.adobe.com/go/devcenter_cf_try) Buy (www.adobe.com/go/devcenter_cf_buy)

An overview of Ajax user interface layout tags

The basic elements that make up an AJAX application include:

- HTML and CSS for the presentation and formatting of the application.
- JavaScript which enables you to script client-side behaviors.
- Asynchronous communication is leveraged through the `XMLHttpRequest` function. This is an API used by JavaScript to transfer XML, JSON, and text from a web server using the HTTP protocol.
- XML or JSON is used as a format for transferring data.

The new Adobe ColdFusion Ajax features can be divided into three categories:

- Application interface layout
- Data binding.
- Server-side components (to be covered in a future article)

If you already have experience building web applications using Ajax technologies, you will find these new tags to be a simpler and more productive way of constructing your applications. If you have not been exposed to Ajax before, these new features are a great way to get started creating richer experiences.

Before diving into examples of how these new tags work, let's take a quick tour of new ColdFusion tags and existing ColdFusion tags that now are able to leverage Ajax capabilities.

The following tags enable you to quickly lay out your Ajax user interface:

- `CFLAYOUT`: Allows four different types of layout:

- Horizontal
- Vertical
- Free form
- Tabbed display

- CFLAYOUTAREA: Use this tag inside of your CFLAYOUT tag to define different regions.
- CFPOD: Creates a pod on the screen with a title bar in which you have the option of specifying a label.
- CFWINDOW: If you have used Flex before, this Ajax component is similar the to <mx:TitleWindow/>. It acts as a pop-up window that can be shown or hidden using scripting. CFWINDOW can also be set to enable resizing.
- CFMENU: Create a horizontal or vertical menu with this easy-to-use tag.
- CFMENUITEM: Use this tag inside your CFMENU tag to add items to your menu.
- CFTOOLTIP: Easily create tool tips in your application to enable a better user experience. You can even add HTML formatting to the tooltip for a better appearance.
- CFAJAXPROXY: Creates a JavaScript proxy between your client side JavaScript and ColdFusion components. It takes care of serializing values into JSON format returning from a CFC call.
- CFINPUT: This is an existing tag with a great new feature called autosuggest. You can program a static list of options, or bind the autosuggest attribute to a CFC to get suggestion data from a data source.

Moreover, there is a new ColdFusion/Ajax Application Wizard. Similar to the ColdFusion/FLEX wizard, the ColdFusion/Ajax wizard plug-in for Eclipse enables you to rapidly create rich, data-driven applications that require no coding from you. Once you have RDS configured in Eclipse with connection to your Adobe ColdFusion 8 server, you can build CRUD (Create Read Update Delete) based applications that leverage the Adobe ColdFusion 8 Ajax capabilities

New Ajax controls for creating user interfaces

The following sections explain the new Ajax controls in Adobe ColdFusion 8 that will help you lay out richer interfaces that provide a better experience for the user.

Laying out the page: CFLAYOUT

Four types of layouts are possible with the CFLAYOUT tag: border, tab, HBox, and VBox. The use of this tag reduces the amount of work you have to do to create flexible, browser-agnostic layouts. The CFLAYOUT tag can have as many CFLAYOUTAREA tags within it as you want. The CFLAYOUTAREA tags can contain any other CFML you want to render on the page.

If you have any experience with Flex, you'll recognize HBox and VBox as the two types of layout containers that will orient their child elements horizontally or vertically. If you're not familiar with the relevant CSS properties for positioning elements on the screen, using HBox and VBox is a quick way to work around that. Figure 1 shows a simple horizontal layout.

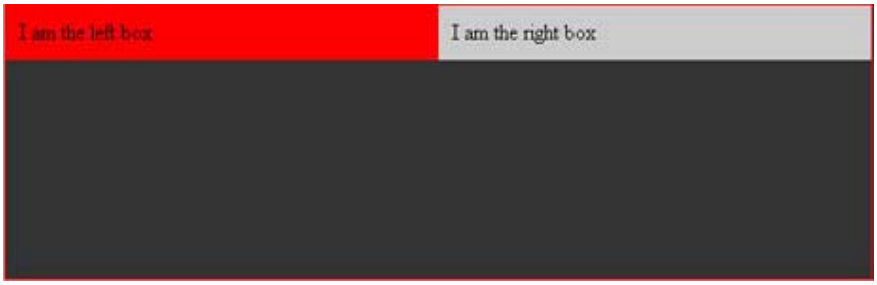


Figure 1. A simple horizontal layout using the CFLAYOUT tag

I used the following code to create the layout shown in Figure 1:

```
<cflayout type="hbox" style="border:1px solid red;height:200px;width:640px;">
  <cflayoutarea name="lbox" title="left box"style="width:300;padding:10px;
     background-color:##f00;">
  HI! I'm in the left cflayout area
  </cflayoutarea>
  <cflayoutarea name="rbox" title="right box"style="width:300;padding:10px;
     background-color:##ccc;">
  HEY! I am in the right cflayout area!
  </cflayoutarea> </cflayout>
```

Figure 2 shows a simple vertical layout.

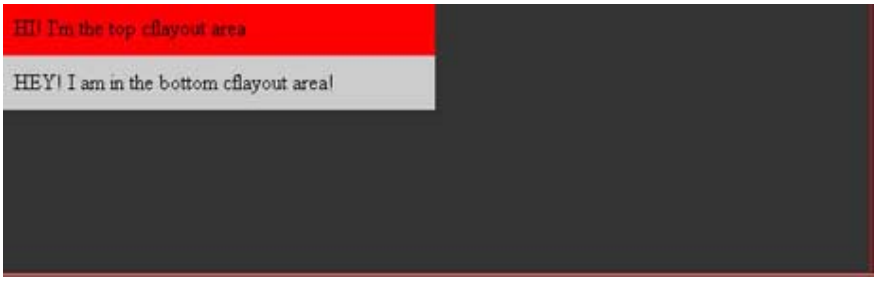


Figure 2. A simple vertical layout using the CFLAYOUT tag

The layout in Figure 2 uses this code:

```
<cflayout type="vbox" style="border:1px solid red;height:200px;width:640px;">
     <cflayoutarea name="tbox" title="top box" style="width:300;padding:10px;
        background-color:red;">
          HI! I'm the top cflayout area
     </cflayoutarea>
     <cflayoutarea name="bbox" title="bottom box" style="width:300;padding:10px;
        background-color:grey;">
          HEY! I am in the bottom cflayout area!
     </cflayoutarea>
</cflayout>
```

A slightly more sophisticated type of layout is the border layout (see Figure 3). This layout is possibly the easiest way to create a generic site layout with collapsible navigation, header, and footer regions, and a content zone. This type of layout is the same layout that the ColdFusion administrator uses.

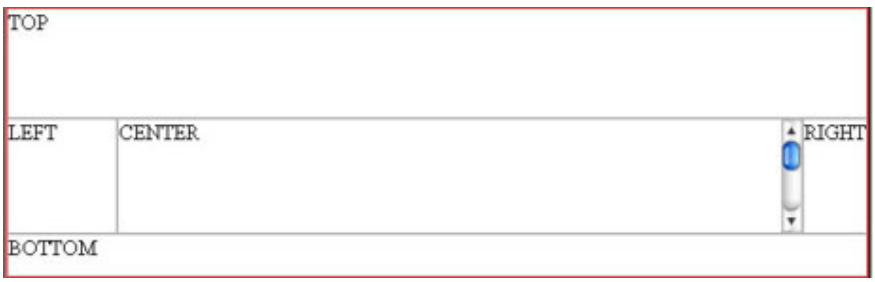


Figure 3. A simple layout using top, bottom, left, right, and center positioned panels

The layout in Figure 3 uses the following code:

```
<cflayout type="border" style="border:1px solid red;height:200px;width:640px;">
   <cflayoutarea position="top" style="height:80px;">
   TOP
   </cflayoutarea>
   <cflayoutarea position="left" style="width:80px;">
   LEFT
   </cflayoutarea>
   <cflayoutarea position="center" style="height:300px;width:400px">
   CENTER
   </cflayoutarea>
   <cflayoutarea position="right" style="height:80px;">
   RIGHT
   </cflayoutarea>
   <cflayoutarea position="bottom" style="height:30px;">
   BOTTOM
   </cflayoutarea>
</cflayout>
```

Some other interesting things that can be done with the border-type layout include collapsible and closable panels. Say you wanted to make the navigation collapsible, it's as simple as adding `collapsible="true"` attribute to the `CFLAYOUTAREA` tag. Note that you must also specify the title attribute for this feature to work (see Figure 4). Similarly, closable panels are easy to implement by specifying a `closable="true"` attribute. One of the nice touches about this feature is that the motion of the panels is a gradual transition from open to closed and vice versa. The effects are very smooth and provide a better, less abrupt interaction for the user than traditional methods. Using the `CFLAYOUT` tag saves the developer a significant amount of time that would have been spent fussing with HTML and CSS to get the panels aligned "just right" in every possible browser.

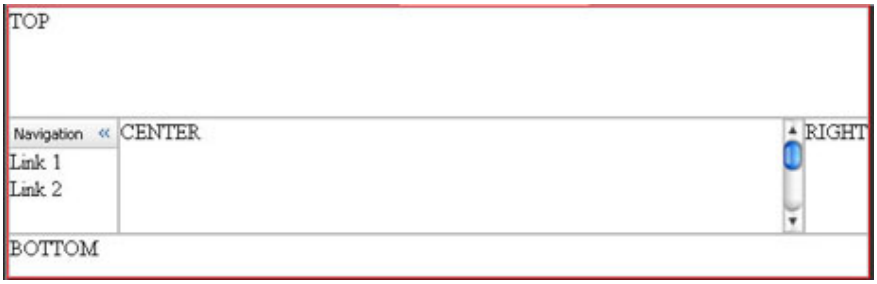


Figure 4. The left-positioned layout tag from Figure 3 with a titled, collapsible navigation added.

For the added feature in Figure 4, I used the following code:

```
<cflayoutarea position="left" style="width:80px;" collapsible="true" title="Navigation">
      nav 1<br/>
      nav 2<br/>
</cflayoutarea>
```

Clicking the header with the Navigation label will cause the left positioned panel to collapse (as shown in Figure 5) and expand.

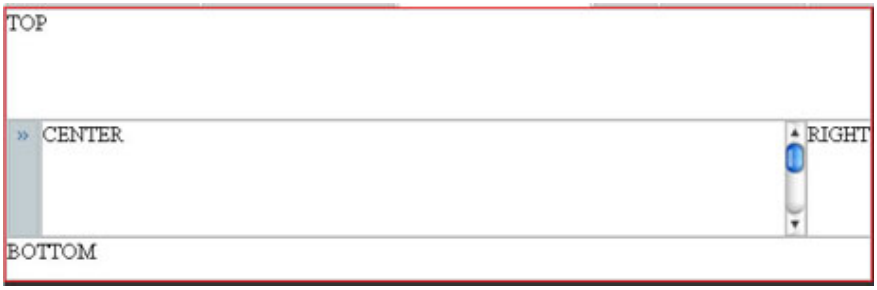


Figure 5. A collapsible panel in its collapsed state

Laying out the page: CFLAYOUT

In many applications, you might need to build a tabbed interface (see Figure 6). The Ajax features in ColdFusion 8 Ajax features simplify this process for you, while helping you shorten development time. You won't waste anymore time developing a cross-browser tabbed UI!

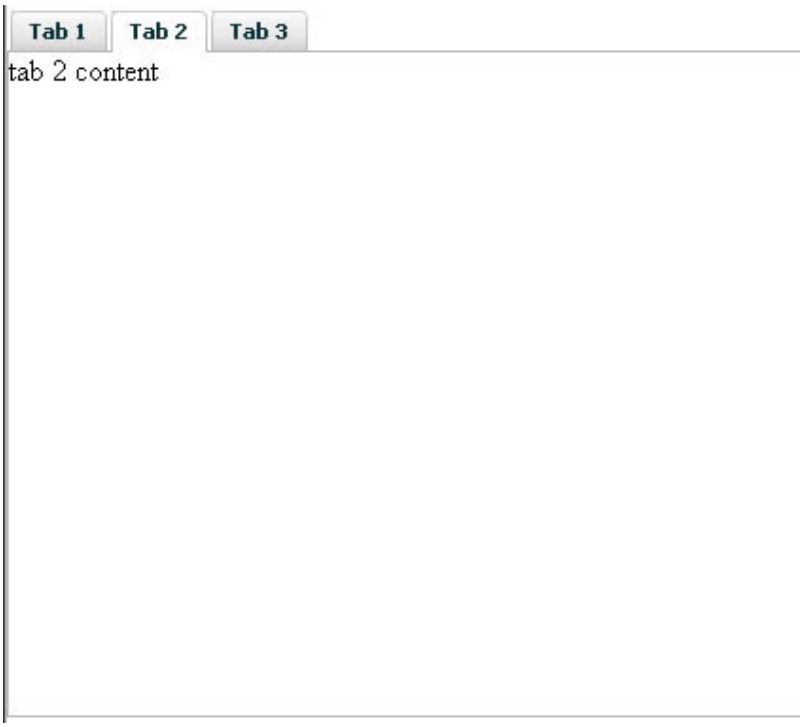


Figure 6. A simple tab-based layout using just a few lines of code

Here is the code for the layout shown in Figure 6:

```
<cflayout type="tab" tabheight="350">
 <cflayoutarea title="Tab 1">tab 1 content</cflayoutarea>
  <cflayoutarea title="Tab 2">tab 2 content</cflayoutarea>
  <cflayoutarea title="Tab 3">tab 3 content</cflayoutarea>
</cflayout>
```

All of the generated layout elements can be referenced and controlled by JavaScript in the rendered page, enabling you to hook in your own controls to drive the user interface as necessary. Stylistically, you can dress up the tabs as you see fit using CSS. Since this feature is driven by the Yahoo! user interface controls, you can refer to plenty of online guides to help you identify and tweak the necessary styles. Check out the *Yahoo! ColdFusion Developer Center* at the Yahoo! Developer Network..

Using CFWINDOW

This tag allows you create and control pop-up windows in your application. If you are familiar with Flex, this is similar to using the `TitleWindow` component. This pop-up widow has the following features:

- A title bar where you can specify a title
- The window can be placed anywhere in the application
- Can be a modal, this means while this window is on the screen, the user cannot interact with the application behind the window
- Drag behavior
- Resizable
- Closable
- You can control the pop-up window with JavaScript once it has been created

The following is an example of the CFWINDOW tag.

```
<cfwindow name="myWindow" title="ColdFusion Window Example" draggable="true"
   resizable="true"
   initshow="true"
   height="250"
   width="250"
   x=375
   y=0>
</cfwindow>
```

The drag and resize behaviors have been set to true. A width and height for the window along with the x and y coordinates dictating where the window should appear in the application when shown.

The attribute `initshow="true"`
means when the page loads, the window will automatically be initialized and shown on the screen. You can set this to `false` and open the window at a later time by calling `ColdFusion.Window.show` JavaScript functions. You can also create event handles for the window. In the code example below you will see in the `initApp` function we are creating two event handles for the window named win1. In the two handlers in this example, I simply show an alert in the browser.

In the following form, you have two buttons in which the `onClick` attribute specifies the JavaScript function to be invoked. This time, you use `ColdFusion.Window.show(windowname)` to control the window in your application.

```
<html>
<head>
<script>
        function initApp(){
                ColdFusion.Window.onShow("win1", openWindowHandler);
                ColdFusion.Window.onHide("win1", closeWindowHandler);
        }

        function closeWindowHandler(){
                alert("You closed me!");
        }

        function openWindowHandler(){
                alert("You opened me!");
        }

        function showMyWindow(){
                ColdFusion.Window.show("win1");
        }

        function hideMyWindow(){
                ColdFusion.Window.hide("win1");
        }

</script>
</head>
<body onload="initApp()">

<cfwindow name="win1"
                title="ColdFusion Window"
                draggable="true"
                resizable="true"
                initshow="false"
                height="250"
                width="250"
                x=375
                y=0/>

<form>
        <input type="button" value="Show my window" onClick="showMyWindow()">
        <input type="button" value="Hide my window" onClick="hideMyWindow()">
</form>
</body>
</html>
```

Creating a menu: CFMENU

An unordered list in HTML combined with a ridiculously complex style sheet seems to be the accepted way to develop a site navigation menu in the past (see Figure 7). Not anymore, Adobe has simplified that process as well with the new `CFMENU` tag.

Could this be any easier?

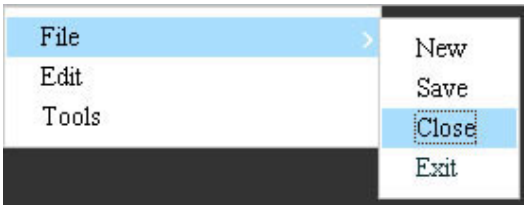


Figure 7. The CFMENU tag,showing a simple nested navigation tree

The following code creates the menu shown in Figure 9:

```
<cfmenu type="vertical" width="200">
      <cfmenuitem display="File">
            <cfmenuitem display="New"/>
            <cfmenuitem display="Save"/>
            <cfmenuitem display="Close"/>
            <cfmenuitem display="Exit"/>
      </cfmenuitem>
      <cfmenuitem display="Edit"/>
      <cfmenuitem display="Tools"/>
</cfmenu>
```

Note that the code contains no complex CSS, nor complex JavaScript; it just contains a simple menu that can render submenus that are easy to maintain and read.

Need a hint: CFTOOLTIP

Every now and then you may want to put a little bit of explanatory text in your web application to help guide users (see Figure 8). You could go find a JavaScript library to do this on your own, or you could use the very simple CFTOOLTIP tag. Just specify the text you want to show in the tooltip and wrap the content you want the tooltip to apply to with the CFTOOLTIP tag.

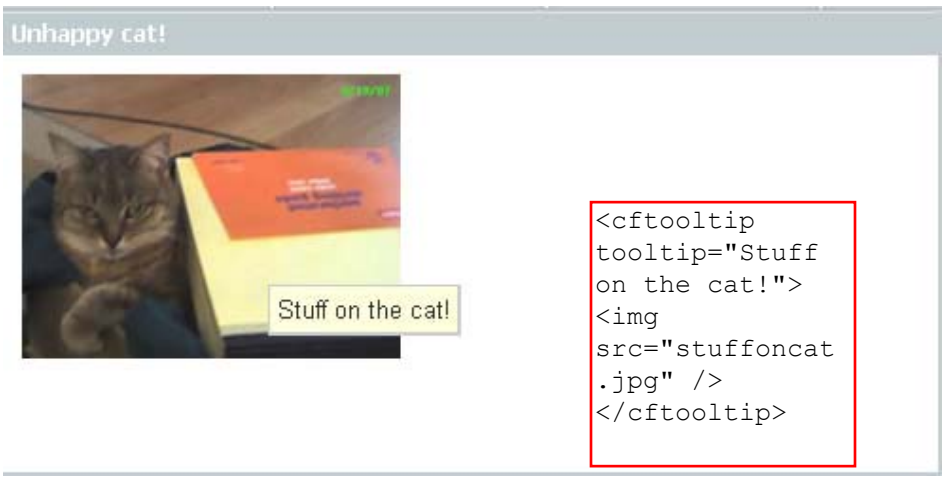


Figure 8. Using the CFTOOLTIP tag

```
<cftooltip tooltip="Stuff on the cat!">
  <img src="stuffoncat.jpg" />
</cftooltip>
```

Updating sections of the page: CFDIV

One of the reasons for using an Ajax user interface is to selectively update sections of the page. Adobe ColdFusion 8 has a new tag called CFDIV that defines a region enabled for update. The CFDIV tag uses the concept of “binding" to accomplish this task. If you have experience developing applications with Adobe Flex or Spry you’ll be very comfortable

with this approach.

You can choose from two methods of binding. The first method is a simple URL binding:

```
<cfdiv id="div1" bind="url:myPage.cfm"></cfdiv>
```

This will automatically include the content from myPage.cfm to be loaded into the div tag. The interesting aspect of this is that the template you're referring to can generate anything on the fly that you need it to. Naturally, you'll want to pass some data to this template which is where the other type of binding comes in—binding to controls on the page:

```
<cfmenu name="nav" type="horizontal" bgcolor="##336699" fontcolor="##fff">
   <cfmenuitem display="Page One"
     href="javaScript:ColdFusion.navigate('pageOne.cfm','myDiv')" style="color:##fff;">
   </cfmenuitem>
   <cfmenuitem display="Page Two"
     href="javaScript:ColdFusion.navigate('pageTwo.cfm','myDiv')" style="color:##fff;">
   </cfmenuitem>
</cfmenu>
<cfdiv id="myDiv" bind="url:pageOne.cfm">
</cfdiv>
```

In this example, the `div` tag first loads pageOne.cfm. Using the `CFMENU` tag to create a simple menu, you will notice the `href` attribute contains JavaScript. The function `ColdFusion.navigate` takes two arguments. The first is the page to load, and the second specifies to load the page into the `CFDIV` tag with the id `myDiv`. When a user clicks the menu item labeled `Page Two`, a request to load `pageTwo`
is initiated and a loading progress image is presented to the user while the page is loaded into the `CFDIV` area.

The `CFDIV` tag can also be bound to a CFC using the following syntax:

```
<cfdiv id="nickname" bind="cfc:states.getNickname({cbState})"></cfdiv>
```

This is much more convenient syntax for those of you who use CFCs for just about everything data-related in your ColdFusion applications.

New Ajax controls for data manipulation

We jumped the gun a little bit in the previous section on `CFDIV` when we mentioned data binding. It's really a fantastic, powerful tool when you're working with an Ajax-powered application. Many ColdFusion tags have inherited functionality for working with data binding in ColdFusion 8, making the rendering of updated information practically effortless on the part of the developer.

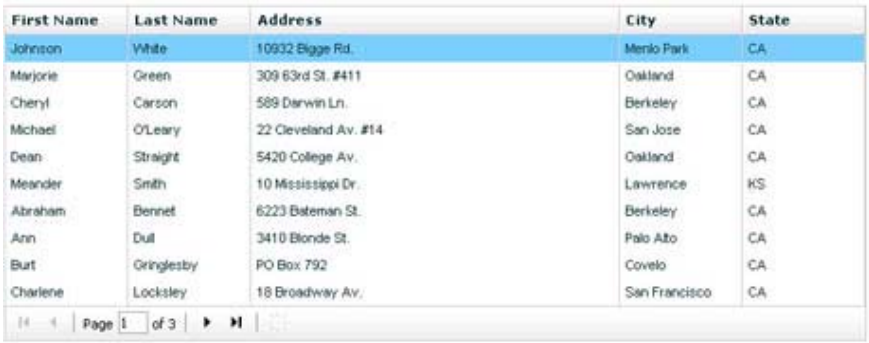
Tabular data: `CFGRID (this tag can only be used inside a <cfform> tag)`
Although this was probably one of those ColdFusion tags you have never used, its well worth another look in Adobe ColdFusion 8. The features worth your attention include the ability to bind to remote data sets, column sorting and reordering, and paging recordsets. In previous versions of ColdFusion, `CFGRID` was implemented as a Java Applet or Flash control. The HTML version has been updated with a helpful dose of JavaScript to make it perform in any environment.

The code to generate a `CFGRID`
control is as simple as always. In the following example, we're using the new CFC binding capability to specify the data that will render in the grid:

```
<cfform>

  <cfgrid name="authors" format="html" bind="cfc:authors.getAuthors({cfgridpage},

       {cfgridpagesize},{cfgridsortcolumn},{cfgridsortdirection})">

  <cfgridcolumn name="AU_FNAME" header="First Name" width="100"/>

  <cfgridcolumn name="AU_LNAME" header="Last Name" width="100"/>

  <cfgridcolumn name="ADDRESS" header="Address" width="300"/>

  <cfgridcolumn name="CITY" header="City" width="100"/>

  <cfgridcolumn name="STATE" header="State" width="100"/>

  </cfgrid>  </cfform>
```

Our back-end CFC (`authors`) has a single method (`getAuthors`) that returns the data that will populate the grid as a structure.

Executing the code results in the output shown in Figure 11.

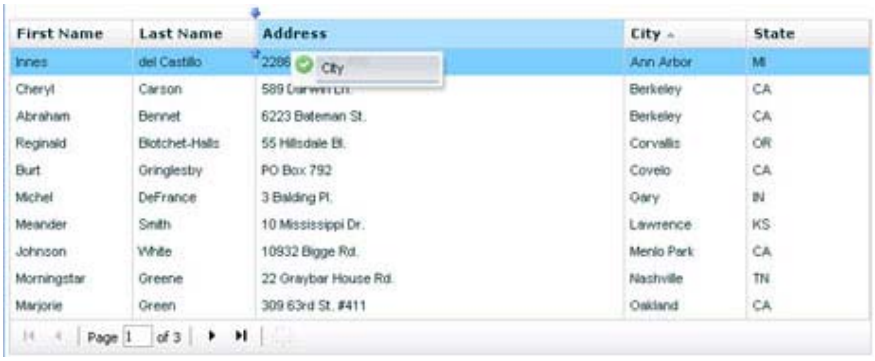


First Name	Last Name	Address	City	State
Johnson	White	10932 Bigge Rd.	Menlo Park	CA
Marjorie	Green	309 63rd St. #411	Oakland	CA
Cheryl	Carson	589 Darwin Ln.	Berkeley	CA
Michael	O'Leary	22 Cleveland Av. #14	San Jose	CA
Dean	Straight	5420 College Av.	Oakland	CA
Meander	Smith	10 Mississippi Dr.	Lawrence	KS
Abraham	Bennet	6223 Bateman St.	Berkeley	CA
Ann	Dull	3410 Blonde St.	Palo Alto	CA
Burt	Gringlesby	PO Box 792	Covelo	CA
Charlene	Locksley	18 Broadway Av.	San Francisco	CA

Page 1 of 3

Figure 9. A `CFGRID control`, populated using the new bind attribute

The resulting grid contains all the necessary controls to move from page to page of output. Clicking the column headings will sort the data according to that column and dragging the column headings will reorder their presentation on the screen (see Figure 12).



First Name	Last Name	Address	City	State
Innes	del Castillo	2286	Ann Arbor	MI
Cheryl	Carson	589	Berkeley	CA
Abraham	Bennet	6223 Bateman St.	Berkeley	CA
Reginald	Blotchet-Halls	55 Hillsdale Bl.	Corvallis	OR
Burt	Gringlesby	PO Box 792	Covelo	CA
Michel	DeFrance	3 Balding Pl.	Gary	IN
Meander	Smith	10 Mississippi Dr.	Lawrence	KS
Johnson	White	10932 Bigge Rd.	Menlo Park	CA
Morningstar	Greene	22 Graybar House Rd.	Nashville	TN
Marjorie	Green	309 63rd St. #411	Oakland	CA

City

Page 1 of 3

Figure 10. Reordering the columns in a `CFGRID` control

Although we only looked at the CFC binding capabilities of the `CFGRID` control, you can also bind it to a URL (like in our `CFDIV` example from above) or a JSON. Take a good long look at this for your data-intensive output needs.

Data binding in CFTREE

When you need to show hierarchal information the `CFTREE` tag is very helpful, and just like previous examples you can

use the `bind` attribute to populate the tree with dynamic data from a CFC.

```
<cfform name="testform">

 <cftree name="t1" format="html">

 <cftreeitem bind="cfc:makeTree.getNodes({cftreeitemvalue},{cftreeitempath})">

 </cftree>

 </cfform>
```

Using `CFRICHTEXTEDITOR`

Based on the FCKeditor, you can use this simple tag to present a rich text editor to your users. You can also customize how the editor is presented to the user.

```
<cftextarea richtext=true name="text01" />
```

Using `CFAUTOSUGGEST`

You too can have autosuggest functionality in your application with ease. When using the `CFINPUT` tag, you can specify a dynamic or static source where your suggest data will come from. You also set the `autosuggestMinLength` attribute, which specifies the number of characters that must be entered before it will display any suggestions.

To show static suggest data, you specify a comma-delimited list like in the following example:

```
<cfinput type="text" autosuggest="Alabama,Alaska,Arkansas,Arizona,Maryland,Minnesota,Mis
```

While the drop-down menu displays, a user can use his up and down arrows and press Enter to select the highlighted label.

You can also use dynamic data from a ColdFusion Component for your autosuggest values.

Using `CFAJAXIMPORT`

Sometimes you may pull other ColdFusion templates that contain Ajax controls dynamically into a `CFDIV` area. You have to make sure that the JavaScript libraries that support those controls are imported into your main page. You use the `CFAJAXIMPORT` tag to specify the libraries that ColdFusion needs to load into your page, so that the libraries are available to other pages you load into your layout. Suppose you had other pages that used the auto-suggest functionality, a date field, and a tree component that would you load into your main page into a div tag for example. You would need this tag in your main application page to instruct the ColdFusion server to include those Ajax libraries in your page. This can be compared to the import statement in ActionScript when you need to import libraries into a class.

```
<cfajaximport tags="CFINPUT-AUTOSUGGEST, CFINPUT-DATEFIELD, CFTREE"/>
```

As you can see, the Adobe ColdFusion 8 Ajax tags are nothing short of remarkable and provide a higher level of productivity for developers when creating richer user interfaces. From laying out your application, loading in sections of the screen without page refreshes, a rich set of form controls and easy connectivity to ColdFusion components, you have a full suite of productive tags to help you get the job done faster. Stay tuned to the *ColdFusion Developer Center (www.adobe.com/devnet/coldfusion)* for more on using new Ajax features in ColdFusion 8.