

**founder & ceo**

Fuat Kircaali fuat@sys-con.com

**group publisher**

Jeremy Geelan jeremy@sys-con.com

**advertising**

**senior vp, sales & marketing**

Carmen Gonzalez carmen@sys-con.com

**vp, sales & marketing**

Miles Silverman miles@sys-con.com

Robyn Forma robyn@sys-con.com

**advertising manager**

Megan Mussa megan@sys-con.com

**associate sales manager**

Kerry Mealia kerry@sys-con.com

Lauren Orsi lauren@sys-con.com

**sys-con events**

**president, events**

Carmen Gonzalez carmen@sys-con.com

**customer relations**

**circulation service coordinator**

Edna Earle Russell edna@sys-con.com

**sys-con.com**

**vp, information systems**

Robert Diamond robert@sys-con.com

**web designers**

Stephen Kilmurray stephen@sys-con.com

Paula Zagari paula@sys-con.com

**accounting**

**financial analyst**

Joan LaRose joan@sys-con.com

**accounts payable**

Betty White betty@sys-con.com

**accounts receivable**

Gail Naples gain@sys-con.com

**subscriptions**

**Subscribe@sys-con.com**

**Call 1-888-303-5282**

For subscriptions and requests for bulk orders, please send your letters to Subscription Department

Cover Price: \$8.99/issue

Domestic: \$89.99/yr (12 issues)

Canada/Mexico: \$99.99/yr

All other countries \$129.99/yr

(U.S. Banks or Money Orders)

Back issues: \$12 U.S. \$15 all others

# Five Cool Things I've Done with ColdFusion

## Mailing labels, bar codes, credit cards, and graphing



By Jeffrey Houser

I'm at my best when I'm challenged. In my consulting business I tend to gravitate towards small businesses with delusions of grandeur. I want to be the one to help them realize their vision and turn their delusion into reality.

Looking back, this has been an interesting week. I thought I'd share some of the interesting things I've done and how ColdFusion helped make them possible.

### Monday: Bar Codes

I've been working over the past year to Web enable an MS Access based scheduling system. One item of the application generates attendance cards with a bar code on it for all users. When a user shows up for a class, that user will scan in, and the information is used to track attendance. The client wanted to be able to generate the attendance cards (with bar codes) from the Web. How do you do that?

First, you have to realize what bar codes are. They are just text, or data of some sort, that looks like a bunch of parallel lines. If you don't know what I mean, just flip over your copy of ColdFusion: The Complete Reference, and you'll find one. (Or you can look at any book or CD). These bar codes are generated using a special font. The client provided me with the font, named "3 of 9 Barcode." A simple Web search will reveal many places where you can get this font. I recommend downloading it straight from the source at <http://www.squaregear.net/fonts/free3of9.shtml>.

Now you're probably thinking that since you have the font name, you can just use the HTML font tag like this:

```
<font name="3 of 9 Barcode">My Bar Code</font>
```

And you don't even have to use ColdFusion, right? Well, you're on the right track, but things aren't that simple. The first mistake is that you need to put an asterisk (\*) before and after the text you want to be able to scan as a bar code. Even with that tidbit of information, you still aren't in the clear. For this approach to work, the bar code font must be installed on the client computer.

When you use a font in HTML like that, that font must be installed on the client computer (not the server), or else a default font is used. The resulting bar code won't scan if it's generated using Helvetica or Times New Roman. (You can trust me on that one.) This font is not a standard font installed on most users computers. Expecting it to be isn't a realistic expectation for Web development. Most people don't have uses for Bar Code fonts in their daily activities. Getting the bar code font installed on all relevant clients of your system would bring you back to the old days of client/server technology where you have to walk through a big dark office building with fluorescent lighting and spend months installing things so the requirements change by the time you're finished with the roll out. There must be a better way, right? Yes!

The answer is to use the ColdFusion Report Builder to generate the attendance cards with bar codes. First, install the bar code font on your development machine and on your production (and/or testing) server. Then, create a new report (or open an existing report). Create a field and give it an expression something like this:

```
 "*" & barcodedata & "*" 
```

In the properties dialog, you should see the 3 of 9 Barcode font listed in the Font name list, as shown in Figure 1. Select that



font and your field will look like a bar code. Above the font selection, there is another option for “embedded font.” Be sure to select true for this option. Normally, you wouldn’t want to embed fonts in a PDF because it makes the overall file size larger, but then you have to ensure that the client computer has the font installed, or it will default itself. Since we don’t want to make sure that the client computer has the bar code font installed (for reasons discussed earlier), we embedded it here.

To test the bar codes, you have to scan them. I used a CueCat. The CueCat company has a long disastrous history which results in CueCat scanners being an extremely cheap buy on eBay. Perform a Web search for all the gory details; they are beyond the scope of this article.

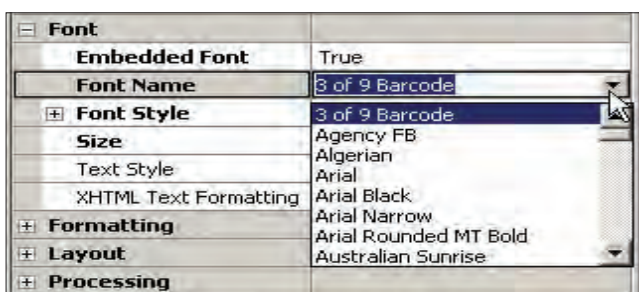


Figure 1

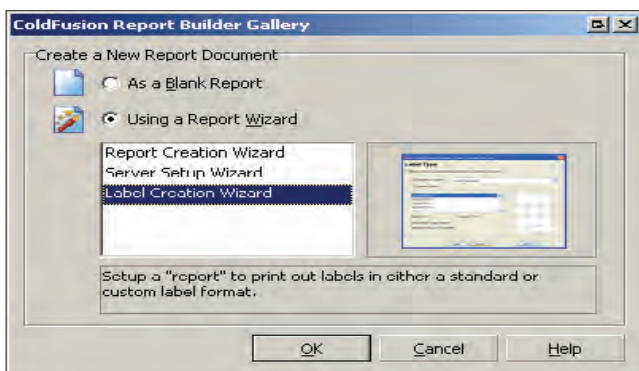


Figure 2: New on the CF Report Builder

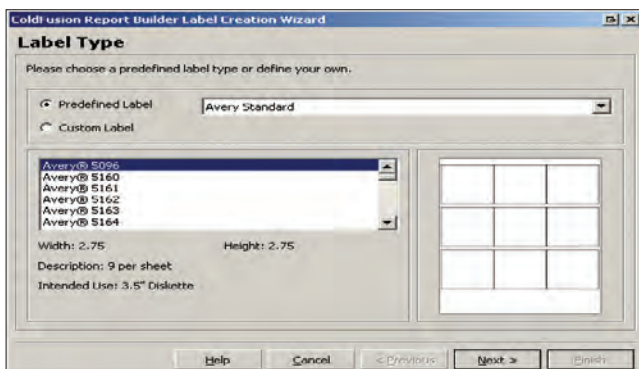


Figure 3: The first step in the Label Creation Wizard

## Tuesday: Mailing Labels

The mid-week point brought another challenge from above. A client wanted to be able to generate mailing labels. Doing this sort of task with HTML is grueling, tedious, and prone to errors. Especially since different browsers can display the same HTML code in different ways. HTML is not designed for print, and forcing what is on screen to line up with what is on your label sheet is going to be the cause some serious headaches. So, what are the options?

I turned once again to my new best friend, the CF Report Builder. The nice man, Dean, at cfreport.org (no last name is known, but I get the impression he works for Adobe) has created a label wizard for the ColdFusion Report Builder. You can download the wizard at <http://www.cfreport.org/index.cfm?mode=entry&entry=38BBAD81-3048-2D03-0A777CDC2FB60D74>. You’ll need to have at least 7.01 of the Report Builder installed (and I recommend getting the 7.02 update). In my case, I already had the wizard installed. Since I don’t remember installing it, it may have been included in the 7.02 update of the Report Builder. Check your install and you may not have to worry about manually downloading and installing the wizard.

You can find out if you have the label wizard installed by selecting File → New. You should see the Label Creation Wizard as an option in the new dialog, as shown in Figure 2. Select the label creator wizard and click next.

You’ll see the first step in the Label Creation Wizard, as shown in Figure 3. Select the label template you want to use. Most labels are based on some form of Avery label template (My client asked for 5160). Select that and click next.

Now you have to define the query fields you want to use to display information on your label, as shown in Figure 4. I manually entered them (The query builder button didn’t work for me). Click the next button

The final step in the report builder is to define the layout of your label. Double-click the elements in the left window, and they’ll be entered in the right window. The right window is an expression window, but you don’t have to worry about writing an expression, because when you click the finish button, text (such as commas) will automatically be escaped when creating the finished template. Save your template and you’re good to go.

The last step in the Report Builder is just to run the report using the cfreport tag. Your template will look something like this:

```
<cfquery name="Getusers" datasource="myDSN">
  select firstName, LastName, State, City, Zip
  from users
</cfquery>

<cfreport format="PDF" template="LabelReport.cfr" query="Getusers">
</cfreport>
```

– CONTINUED ON PAGE 20

## Five Cool Things I've Done with ColdFusion

— CONTINUED FROM PAGE 9

Your query will most likely be different, depending on the information you want to print on a label, but you should be able to apply these concepts to your own development.

### Wednesday: Credit Card Swipes

One of my clients runs an e-commerce business, and the site implementation is about as classic as an on-line shopping cart can come. The client is in the process of opening up a retail store, and I spent some time modifying the shopping cart to accommodate retail transactions. The biggest difference between retail transactions and Web-based transactions is that you have a credit card to swipe. Swiped cards get a much better merchant rate than Internet or phone orders received, so that is an added benefit of the retail store vs. the Internet store.

The client provided me with a USB credit card swiper so that I could test the process. As far as the computer is concerned, the swiper is not much different than a keyboard; you just get a bunch of data in one scan instead of having to type it out manually. The scanned data is sent to the active program running on your machine. This can be notepad (which is great for testing) or the active form field on a Web browser. I used the latter to integrate into their Web based shopping cart.

The data contained on a credit card consists of the card number, some bank routing information, the customer's name, and other miscellaneous information. More details on the specifics of the credit card swipe data are at <http://money.howstuffworks.com/credit-card1.htm>. To process the credit card data you don't need to know the specifics of what it is. The PayFlow Pro Web API has a field for sending swipe data, so it's easy to send it off the transaction and receive information back just as you would a normal Web transaction. I would expect that most payment processors have a similar option.

### Thursday: Processing Bar Codes and Credit Card Swipes

Now that the bar codes were generated, the client wanted to be able to process scanned bar codes in the Web browser. The CueCat, like other bar code readers, operates in the same manner that the credit card scanner does. When you swipe a card or scan a bar code, the scanned (or swiped) information goes to the active screen. If you have a Web browser open, it will go to



Figure 4: Enter your query Fields

the selected form field. To start the input page, first we'll need the form:

```
<form action="BarCodeScanner.cfm" method="post"
      name="BarCodeScanner" id="BarCodeScanner">
  Scanned Info: <input type="text" name="ScannedInfo">
</form>
```

This is a form that will submit back onto itself. The form has one input field, with no submit button. For attendance purposes in the scheduling system, the scan had to be as painless as possible without any user interaction, so the form is submitted through JavaScript.

How does the browser know we've scanned something? How do we know when to submit? The answer is through the use of a JavaScript timer. The timer is a JavaScript that counts down from some number to 0. When it reaches 0, it checks whether there is data in the ScannedInfo field (or not) and then submits the form if there is.

I used the JavaScript code located at <http://www.mcfedries.com/JavaScript/timer.asp> as a base for this. (Much thanks to Paul McFedries' for his assistance). For space reasons, I'll refrain from copying the full script here. There are four variables to take into account:

- *Secs*: The secs variable specifies how many clock ticks until the timer reaches 0. This will count down from the initial value to 0.
- *Delay*: This is the amount of milliseconds before the timer checks its next status. 1000 milliseconds are equal to one second, so that is the default.
- *TimerID*: The timerID is a variable internal to the script. It specifies when the next clock tick will occur.
- *TimerRunning*: The timerRunning variable is a Boolean value that tells us whether the timer is running or not. For all intents and purposes, it is internal to the script and you don't need to worry about it.

There are three functions that make up the timer:

- *InitializeTimer*: The initialize timer function sets the secs variable, stops the timer (if currently running), and then starts the timer.
- *StopTheClock*: The stop the clock function runs some script to prevent the clock from running.
- *StartTheTimer*: The StartTheTimer function is the meat of the execution. It counts down from the clock tick and executes certain code if the count down is done, or moves onto the next clock tick. Of these three functions, it is the StartTheTimer function that needs a more in depth examination.

```
function StartTheTimer(){
  if (secs==0){
    StopTheClock()
    if (document.BarCodeScanner.ScannedInfo.value != ''){
      document.BarCodeScanner.submit();
    } else {
```

```

    InitializeTimer();
}

}else {
secs = secs - 1
timerRunning = true
    timerID = self.setTimeout("StartTheTimer()", delay)
}
}
}

```

The StartTheTimer function starts by checking to see if the secs variable has counted down to zero yet. If it has, first it stops the clock. Then it checks to see if the form field has data in it. If the form field has data, submit the form. Otherwise, restart the clock.

If the time hasn't counted down to zero yet, subtract one from the counter, and set the function to execute again in one second. The function is recursive, calling itself. Load the page, and the timer waits, checking to see if something had been entered. Once it has, it submits the form, which does some further processing.

To start the ball rolling, you need to execute the Initialize-Timer function, like this:

```
InitializeTimer();
```

As long as that is somewhere on the page, you're all set. This example is a bit more simplified than what actually went into production (obviously since we didn't create a processing page). I did some Ajax-y stuff with iFrames to show a running status of what attendance was entered and when. Errors were also shown in the status list without interrupting future scans. It worked out well.

## Friday: Graphing Types

I was sitting with a client discussing a report. He wanted to view the report information in a graphical format, so we had decided to implement this report using cfchart. I made the mistake of asking if he would prefer a bar chart or a pie chart. The client, being like normal clients, asked if I could do both. It turns out I could.

There are three tags that are used to generate charts in ColdFusion: cfchart, cfchartseries, and cfchartdata. Cfchart defines overall attributes for graph, such as the format (PNG, JPG, or Flash), height, width, and other display attributes that affect the full chart. Full information is at the livedocs at <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000226.htm#2619630>. cfchartseries is used to define the chart style, such as bar or pie. There are eleven different types of charts supported. Full documentation on the tag is at <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000228.htm#2741830>. You can have multiple chart series inside a chart, but this tag must be nested inside a cfchart. cfchartdata is used to define a single point on the graph, and it is the simplest of the three tags, as it only defines the name of the point and its value. Cfchartdata must be located inside a cfchartseries. Its full documentation is located here: <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000228.htm#2741830>.

I didn't want to have to build multiple versions of the same

report, one with a bar chart and one with a pie chart. I decided to pass the chart type as a URL variable into the template, and use that to decide what type of chart to display. This gives me the option of easily adding any of the 11 types supported by cfchart when the client asks for more in the future.

First I cfparamed a charttype variable to default to:

```
<cfparam name="chartType" default="pie">
```

That should cover me in the case of nothing being passed in. The next step was to get the data. I decided to use a query. In this example, I'll pull from my MyFriend's RSS Aggregator database and display the number of items for each RSS Feed:

```

<cfquery name="getTotals" datasource="#request.dsn#">
select RSSFeeds.title, count(Items.ItemID) as total
from RSSFeeds join items on (RSSFeeds.RSSFeedID = Items.RSSFeedID)
group by RSSFeeds.title
</cfquery>

```

And finally the page can display the chart like this:

```

<cfchart format="flash">
<cfchartseries itemcolumn="title" valuecolumn="total"
    query="getTotals" type="#chartType#" />
</cfchart>

```

Since the chart is getting data from a query, the cfchartdata tag is not needed. The cfchart, simply enough, tells us that we're creating a flash chart. I can load the page using URLs like this:

```


ChartTest.cfm?chartType=pie
ChartTest.cfm?chartType=bar
ChartTest.cfm?chartType=line

```

(and so on)

It was a simple, elegant solution that gave me another happy client.

## Conclusion

It was a good week. I love dealing with ColdFusion as a language because of the way it makes complicated development tasks seem simple. I feel confident knowing that it will help me address the next "crazy" idea. What are your favorite "odd" requests you've had to deal with? How did you solve them? Just fill out the contact form on my blog ([www.jeffryhouser.com](http://www.jeffryhouser.com)) and let me know! 

## About the Author

Jeffry Houser has been working with computers for over 20 years and in Web development for over 8 years. He owns a consulting company and has authored three separate books on ColdFusion, most recently *ColdFusion MX: The Complete Reference* (McGraw-Hill Osborne Media).

[jeff@instantcoldfusion.com](mailto:jeff@instantcoldfusion.com)