

# ColdFusion Cookbook

## Introduction

Welcome to the ColdFusion Cookbook. The ColdFusion Cookbook was created by Raymond Camden ([ray@camdenfamily.com](mailto:ray@camdenfamily.com)) and is administrated by him along with Jeremy Petersen ([jer@petersenfam.com](mailto:jer@petersenfam.com)). The purpose of the site was to provide a way for ColdFusion developers to quickly find solutions to common problems. The site contains content by many authors, and is always open to submissions from the public.

This document was generated on July 28, 2006 at 7:14 PM. It contains 105 entries.

## Copyright

The text of each entry was written by it's author. You may not copy this text without attributing the author.

## Does CFFILE have a file size limit?

There is no specific limit on file size for CFFILE. However, CFFILE loads the file into the server's memory, so you will get an error if the file size exceeds the amount of free RAM.

Also note, you can place a server wide limit on the size of uploads in the CF administrator. Under the "settings" link, see: Maximum size of post data (MB), Request throttle threshold (MB), and Request throttle memory (MB).

This question was written by [Jacob Munson](#).

It was last updated on July 13, 2006 at 4:36:25 PM EDT.

## How can ColdFusion cache a database query?

ColdFusion query caching is used to keep frequently accessed query results in memory rather than having to pull the results from the database again and again.

Cached queries work by using either the `cachedAfter` or the `cachedWithin` attributes of the `<cfquery>` tag. In order to use either form of cached queries, query caching must be enabled in the ColdFusion administrator.

The `cachedAfter` attribute is used to cache a query after a certain date has passed by passing in a specific date.

```
<cfquery
name="qAfterTest"
datasource="myDs"
cachedAfter="10-10-2005">
select name
from recipes
</cfquery>
```

The `cachedWithin` attribute is used to cache a query within a specified date/time range by passing in a valid data/time range.

```
<cfquery
name="qWithinTest"
datasource="myDs"
cachedWithin="#createTimeSpan(0, 5, 0, 0)#">
select name
from recipes
</cfquery>
```

It is important to note that cached queries are identified by the exact SQL code and `<cfquery>` tag attributes (`datasource`, `name`, etc.) used in the `<cfquery>` tag call that created them. The only exceptions to this rule are the `cachedAfter` and the `cachedWithin` attributes themselves can be altered without effecting the identification of a cached query.

This question was written by [Jeremy Petersen](#).  
It was last updated on March 8, 2006 at 11:13:23 AM EST.

### CFML Referenced

[<cfquery>](#)

## How can ColdFusion generate an excel file?

Because recent Excel products support HTML table format, getting ColdFusion to generate an Excel file can be as simple as creating your HTML tables and then using the `<cfcontent>` tag to set the mime type of your newly generated Excel file.

The following code sample taken from the CF 7 documentation shows a sample of using `<cfheader>` and `<cfcontent>` to push a dynamic Excel file to the browser (prompt the user whether to save the Excel file or open it in a browser).

```
<cfheader name="Content-Disposition" value="inline; filename=acmesalesQ1.xls">
<cfcontent type="application/vnd.msexcel">

<table border="1">
<tr><td> Month</td><td> Quantity</td><td> $ Sales</td></tr>
<tr><td> January</td><td> 80</td><td> $245</td></tr>
<tr><td> February</td><td> 100</td><td> $699</td></tr>
<tr><td> March</td><td> 230</td><td> $2036</td></tr>
<tr><td> Total</td><td> =Sum(B2..B4)</td><td> =Sum(C2..C4)</td></tr>
</table>
```

Another option to consider is the Jakarta POI project: <http://jakarta.apache.org/poi/>. As per the project home page: The POI project consists of APIs for manipulating various file formats based upon Microsoft's OLE 2 Compound Document format using pure Java. In short, you can read and write MS Excel files using Java.

This question was written by [Jeremy Petersen](#).  
It was last updated on July 28, 2006 at 4:17:11 PM EDT.

### CFML Referenced

[<cfcontent>](#)  
[<cfheader>](#)

## How can I access a query column if I have the column name stored in a variable?

Use structure notation to access the recordset:

```
<cfset colName="lastname">
<cfoutput>
#myQuery[colName][1]#
</cfoutput>
```

Keep in mind that with this notation you need to provide a row index as well, even within a query loop:

```
<cfset colName="lastname">
<cfoutput query="myQuery">
#myQuery[colName][myQuery.currentrow]# <br />
</cfoutput>
```

This question was written by Christoph Schmitz.  
It was last updated on March 23, 2006 at 6:48:39 AM EST.

### CFML Referenced

[<cfoutput>](#)

## How can I automate cached queries to update at an exact time each day?

While ColdFusion gives you the ability to choose how long query data is cached, and even couple of options to clear a cached query by hand, you may still find a situation that requires more precise control over your cached query updates.

For example, say you have a query that generates a list of the newest recipes submitted to your recipe site. Because this query is used so often, you choose to cache it. However, you would like to update the cached query at exactly noon each day to reflect a daily cutoff for new recipe entries. How could you do this?

The solution lies in using the scheduling engine of your choice (ColdFusion server has one built in) to run a ColdFusion template that will refresh your cached query.

If you were using the following cached query in your pages:

```
<cfquery
name="qWithinTest"
datasource="myDs"
cachedwithin="#createTimeSpan(1,0,0,0)#">
select name
from recipes
</cfquery>
```

You could create a template that flushes the query cache using the following code:

```
<cfquery
name="qWithinTest"
datasource="myDs"
cachedwithin="#createTimeSpan(0,0,0,-1)#">
select name
from recipes
</cfquery>
```

It is then a simple matter of using your scheduling engine to run this query flush template at noon each day.

This question was written by [Jeremy Petersen](#).

It was last updated on March 15, 2006 at 11:52:42 AM EST.

### CFML Referenced

[<cfquery>](#)

## How can I cache a ColdFusion page on both the client machine and the ColdFusion server?

Use the [<cfcache>](#) tag with the action attribute set to either cache or optimal. Using the [<cfcache>](#) tag for combination caching uses a combination of both client-side and server-side caching. In this model, first the client browser will be checked for a cached copy of the page, if this check fails, then the server will try to get the data from its own cache. Combination caching optimizes ColdFusion server performance and is recommended over server-side only caching.

Note:

- 1) The [<cfcache>](#) tag should be placed at the top your ColdFusion template.
- 2) The [<cfcache>](#) tag treats each distinct URL combination as its own page. So the output of foo.cfm?key=1 and foo.cfm?key=10 would be cached as separate files.
- 3) Combination caching should not be used for caching client specific pages.

```
<cfcache  
action = "cache"  
directory = "C:/temp/cache"  
timespan = "#createTimeSpan(0,1,0,0)#">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on March 2, 2006 at 11:58:23 AM EST.

### CFML Referenced

[<cfcache>](#)

## How can I cache a ColdFusion page on the client machine?

Setting the action attribute of the [<cfcache>](#) tag equal to "clientCache" gives you programmatic control over whether a client browser should reload a ColdFusion page, or if it can use its local cached copy of the ColdFusion page.

Because the page is being stored on the client side, you can cache user specific versions of a dynamic web page without fear of client data being shown to the wrong client. This is a great technique to help speed up user specific pages that either have heavy access patterns, or take a while to display. It is also nice because client-cached files do not take up resources on the ColdFusion server.

Note:

- 1) The [<cfcache>](#) tag should be placed at the top your ColdFusion template.
- 2) The [<cfcache>](#) tag treats each distinct URL combination as its own page. So the output of foo.cfm?key=1 and foo.cfm?key=10 would be cached as separate files.

A code sample for using the [<cfcache>](#) tag to cache a ColdFusion page on the client machine would look as follows:

```
<cfcache  
action = "clientCache"  
directory = "C:/temp/cache"  
timespan = "#createTimeSpan(0,0,1,0)#">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on February 22, 2006 at 10:54:46 AM EST.

### CFML Referenced

[<cfcache>](#)

## How can I cache a ColdFusion page on the ColdFusion server?

The `<cfcache>` tag also gives you the ability to cache pages on the ColdFusion server by setting the action attribute equal to "serverCache". This type of caching is perfect for non-personalized pages that have high usage.

Note:

- 1) The `<cfcache>` tag should be placed at the top your ColdFusion template.
- 2) The `<cfcache>` tag treats each distinct URL combination as its own page. So the output of `foo.cfm?key=1` and `foo.cfm?key=10` would be cached as separate files.
- 3) ServerCache caching should not be used for caching client specific pages.

A code sample would look as follows:

```
<cfcache  
action = "serverCache"  
directory = "C:/temp/cache"  
timespan = "#createTimeSpan(0,1,0,0)#">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on March 2, 2006 at 12:18:40 PM EST.

### CFML Referenced

[<cfcache>](#)

## How can I cache the results of a block of ColdFusion code?

The ColdFusion `<cfsavecontent>` tag is a convenient way to store the results of a block of ColdFusion code. The `<cfsavecontent>` tag is called with both a beginning `<cfsavecontent>` and an ending `</cfsavecontent>` tag. `<cfsavecontent>` has a single required attribute called `variable`. This attribute will hold the cached results of the `<cfsavecontent>` tag.

So now that you know how to call the `<cfsavecontent>` tag, what does it do? When you wrap a block of ColdFusion code with a `<cfsavecontent>` tag, the block of code executes, but any output that is generated by the block of code doesn't display to the screen. Instead, the output is stored in the variable that you choose in the `variable` attribute of the tag.

```
<cfsavecontent variable="cachedOutput">
I am going to count to 10!<p>
<cfloop index="loopOn" from="1" to="10">
  <cfoutput>#loopOn#<br></cfoutput>
</cfloop>
</cfsavecontent>
```

If you want to see the output, you would need to do the following:

```
<cfoutput>#cachedOutput#</cfoutput>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on February 8, 2006 at 11:27:43 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[<cfloop>](#)  
[<cfsavecontent>](#)

## How can I clear a client-side or server-side cache that was created by the <cfcache> tag?

The [<cfcache>](#) tag provides an easy way to clear the contents of a [<cfcache>](#) cached page before the cached page would time out on its own. By using the action="flush" attribute, you can flush the contents of the cached page so that it can be recached with current data. The simplest way to do this is to run the following code:

```
<cfcache action = "flush">
```

The above line of code would clear all cached files in the same directory as the template you called it from. If you need more control over flushing a [<cfcache>](#) cached page, you can use the following optional attributes: directory and expireURL.

The directory attribute lets you specify the directory that contains the cached files you wish to clear.

The expireURL attribute is a URL reference that you can use to choose what specific cache files will be deleted. You can use an \* character as a wildcard. Some sample URL's would be "foo.cfm?Key=1" or "foo.cfm?\*". The first sample would only delete the cached copy of foo.cfm with the URL parameter key = to 1, where as the second sample would delete the cached copy of any foo.cfm page with any URL parameters.

This question was written by [Jeremy Petersen](#).  
It was last updated on March 7, 2006 at 10:47:18 AM EST.

### CFML Referenced

[<cfcache>](#)

## How can I clear the cache of a recordset created with <cfquery>?

Following the post on "How can ColdFusion cache a database query?", it is often necessary to display the results of an updated record instantly after the update transaction completes. If the `cachedwithin` attribute of [<cfquery>](#) is used in the SQL, the updates will not show until after the time span stipulated has expired.

In order to have the results show immediately, you can simply run another sql statement immediately after your update sql has completed.

Take your original query used to obtain your recordset. It contains the `cachedwithin` attribute with a time set to cache the query for 3 hours:

```
<cfquery name="myQry" datasource="myDatasource" cachedwithin="#createTimeSpan(0,3,0,0)#">
select foo
from tblfoobar
order by bar desc
</cfquery>
```

Run your update SQL (the 'name' attribute is optional, but a good practice to include it anyway):

```
<cfquery name="myInsertQry" datasource="myDatasource">
insert into foobar (foo)
values (thisfoo)
where id = <cfqueryparam value="1" cfsqltype="cf_sql_integer">
</cfquery>
```

To clear the cache of the 'myQry' query that was generated previously, run another SQL statement immediately after the update and set the `cachedwithin` attribute to a past time:

```
<cfquery name="myQry" datasource="myDatasource" cachedwithin="#CreateTimeSpan(0,0,0,-1)#">
select foo
from tblfoobar
order by bar desc
</cfquery>
```

Here's the full update code that can be used:

```
<cftransaction>
<cfquery name="myInsertQry" datasource="myDatasource">
insert into foobar (foo)
values (thisfoo)
where id = <cfqueryparam value="1" cfsqltype="cf_sql_integer">
</cfquery>
<cfquery name="myQry" datasource="myDatasource" cachedwithin="#createTimeSpan(0,0,0,-1)#">
select foo
from tblfoobar
order by bar desc
</cfquery>
</cftransaction>
```

Remember, the 'clear' query must be the exact same syntax as the original query or it will not work. It's also a good practice to wrap your SQL transactions in the [<cftransaction>](#) tag. In case some sort of error occurs during the db write, the chance of chance of data corruption is reduced.

This question was written by Brian Moss.  
It was last updated on March 9, 2006 at 1:45:34 PM EST.

### CFML Referenced

[<cfquery>](#)  
[<cftransaction>](#)

## How can I convert ColdFusion variables into JavaScript variables?

Use the [toScript\(\)](#) function to create JavaScript variables from a ColdFusion variable. This function can convert ColdFusion strings, numbers, arrays, structures, and queries to JavaScript syntax that defines equivalent variables and values.

```
<cfset thisString="hello world">
<script type="text/javascript" language="JavaScript">
<cfoutput>
var #toScript(thisString, "jsVar")#;
</cfoutput>
</script>
```

When ColdFusion runs this code, it sends the following to the client:

```
<script type="text/javascript" language="JavaScript">
var jsVar = "hello world";
</script>
```

This question was written by [Jeremy Petersen](#).

It was last updated on March 23, 2006 at 3:56:18 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[toScript\(\)](#)

## How can I detect if the browser accepts cookies?

This script must be placed in an empty page, without any content. Since the page will never be displayed to the browser. The "real" pages are yescookie.cfm and nocookie.cfm.

Please note we use server-side redirect (forward) instead of client-side redirect ([<cflocation>](#)) since search engine's spiders tend to penalize websites that perform client-side redirect.

As an added bonus, using [getPageContext\(\).forward\(\)](#) we keep the same url visible inside the browser's bar, allowing a better user experience and proper bookmarking.

```
<cfif structKeyExists(cookie, "tmtCookieTest")>
<cfset getPageContext().forward("yescookie.cfm")>
<cfelseif NOT structKeyExists(url, "tmtCookieSend")>
<!-- First time the user visit the page, set the cookie --->
<cookie name="tmtCookieTest" value="Accepts cookies">
<!-- The cookie was send, redirect and set the tmtCookieSend flag as an url variable --->
<cfset getPageContext().forward("#cgi.script_name#?tmtCookieSend=true")>
<cfelseif structKeyExists(url, "tmtCookieSend")>
<!-- We tried sending the cookie, no way, cookies are disabled, get out of here --->
<cfset getPageContext().forward("nocookie.cfm")>
</cfif>
```

This question was written by [Massimo Foti](#).

It was last updated on May 22, 2006 at 11:03:47 AM EDT.

### CFML Referenced

[<cflocation>](#)

[<cfif>](#)

[getPageContext\(\)](#)

## How can I display a message on a long-running page?

By default, ColdFusion will not return any HTML until the entire page has rendered. For a long-running page, this may make the user think nothing is happening, resulting in the user hitting reload multiple times.

ColdFusion provides a tag that will flush out the current data to the screen: [<cfflush>](#). For example:

```
<p>
This is a slow page, please stand by...
</p>

<cfflush>

<cfloop index="x" from="1" to="999999">
  <cfset doNothing = x*2/3>
</cfloop>

<p>
I'm finally done.
</p>
```

In this example, the user will see the "please stand by" warning immediately. After the page has processed they will then see the final message. A few warnings about [<cfflush>](#). When used - the following tags and functions may no longer be used: [<cfcontent>](#), [<cfcookie>](#), [<cfform>](#), [<cfheader>](#), [<cfhtmlhead>](#), [<cflocation>](#), and [SetLocale\(\)](#). Also, some browsers, like the wonderful Internet Explorer, will ignore your text unless you have "enough" text. If you see nothing in Internet Explorer, you can simply pad your output by adding: `repeatString(" ", 100)`. This fools the browser into thinking enough text has been sent to render.

This question was written by [Raymond Camden](#).  
It was last updated on February 23, 2006 at 7:11:59 AM EST.

### CFML Referenced

[<cfloop>](#)  
[<cfif>](#)  
[<cfcookie>](#)  
[<cfflush>](#)  
[<cfhtmlhead>](#)  
[<cflocation>](#)  
[<cfcontent>](#)  
[<cfheader>](#)  
[setLocale\(\)](#)

## How can I dynamically find what form fields have been posted into a page?

ColdFusion provides two easy ways to obtain a list of all form variables that have been posted to a page.

The first way is to use the `form.fieldNames` variable. The `form.fieldNames` variable is automatically available to any ColdFusion template that has received a form post and contains a comma-delimited list of form-field names that have been posted to the current template.

The second way is to use the form structure. The form structure is a special ColdFusion structure that contains each form-field name and its associated value. The following is a code sample for displaying the content of the form structure. Note how the `form.fieldNames` is filtered out of the result set:

```
<cfloop collection="#form#" item="theField">
<cfif theField is not "fieldNames">
<cfoutput>
#theField# = #form[theField]#<br>
</cfoutput>
</cfif>
</cfloop>
```

This question was written by [Jeremy Petersen](#).

It was last updated on April 17, 2006 at 9:35:54 AM EDT.

### CFML Referenced

[<cfoutput>](#)

[<cfloop>](#)

[<cfif>](#)

## How can I ensure there are no leading or trailing spaces in my variable?

When working with data that you have no control over, you sometimes need to do a bit of cleanup on strings. For example, a web form that accepts user input may send data with extra spaces at the beginning or end of the value.

ColdFusion provides three functions to address the issue of leading or trailing spaces in ColdFusion: [trim\(\)](#), [rTrim\(\)](#), and [lTrim\(\)](#). The functions [rTrim\(\)](#) and [lTrim\(\)](#) will strip spaces from the end or beginning of a string. Normally you want to strip white space from both the beginning and end of a string. For that you would use the [trim\(\)](#) function.

```
<cfset str = " Jacob turned six years old on Wednesday ">
<cfset str = trim(str)>
<cfoutput>#str#</cfoutput>
```

This question was written by [Charlie Grier](#).  
It was last updated on January 27, 2006 at 12:32:45 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[rTrim\(\)](#)  
[lTrim\(\)](#)  
[trim\(\)](#)

## How can I generate static HTML from a dynamic ColdFusion page?

One very powerful caching technique is to run your ColdFusion code to generate a dynamic web page, and then write the contents of the dynamic page to a static HTML file. The static page can then be loaded time and time again without the expense of having to rebuild the page.

To accomplish this task, you can use a combination of the ColdFusion [<cfsavecontent>](#) and [<cfile>](#) tags, or the ColdFusion scheduling engine to write the HTML generated from running a dynamic page to a static HTML file.

The first method to generate and static documents from dynamic content is to use a combination of the ColdFusion [<cfsavecontent>](#) and [<cfile>](#) tags. The [<cfsavecontent>](#) tag stores the generated output from a block of ColdFusion and HTML code into a variable. So once you have that variable, it is a simple matter of using the [<cfile>](#) tag to write the data to a static HTML file. The following code shows a sample of this technique:

```
<!-- Create cached contents (HTML) -->
<cfsavecontent variable="cachedOutput">
<html>
<head>
<title>Cached File Example</title>
</head>
<body>
<h3>Cached File Example</h3>

<cfloop index="loopOn" from="1" to="10">
  <cfoutput>The loop is on: #loopOn#<br></cfoutput>
</cfloop>

</body>
</html>
</cfsavecontent>

<!-- Write cached contents (HTML) to file -->
<cfile action="write"
file="C:\temp\cachedFile.html"
output="#cachedOutput#>
```

A second way to generate static documents from dynamic content is to use the ColdFusion scheduling engine. Along with the ability to generate static documents from dynamic content, using the ColdFusion scheduling engine gives you the added ability to automatically update your static documents on whatever schedule you may require.

There are two main options for working with the ColdFusion scheduling engine. The first option is by way of the ColdFusion Administrator, and the second is by using the [<cfschedule>](#) tag. For our example, we will use this second method.

The real magic of using the ColdFusion scheduling engine to generate static documents from dynamic content lies in the publish attribute. If this attribute is set to yes, then the results of the page the scheduled task runs will be saved to disk (as per the file and path attributes). Beyond this, we are simply telling the ColdFusion scheduling engine what page to run, and how often the page should be ran. Some sample code using this technique would look as follows:

```
<cfschedule action = "update"
task = "createStaticPage"
operation = "HttpRequest"
file = "cachedFile.html"
path = "C:\inetpub\wwwroot\"
startDate = "2/13/2006"
startTime = "12:00 PM"
url = "http://127.0.0.1/createPage.cfm"
publish = "yes"
interval = "3600"
resolveURL = "yes">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on February 13, 2006 at 11:27:30 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[<cfschedule>](#)  
[<cfloop>](#)  
[<cfile>](#)  
[<cfsavecontent>](#)

## How can I get a list of files in a directory?

Use the [<cfdirectory>](#) tag. Don't forget to use the `recurse="true"` attribute if you want the list to include the contents of subdirectories.

```
<cfdirectory action="list"
directory="#getDirectoryFromPath(getTemplatePath())#"
name="currentDir">

<cfoutput query="currentDir">
#name#<br>
</cfoutput>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on February 28, 2006 at 8:46:38 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[<cfdirectory>](#)

## How can I get user and Search Engine Friendly URLs?

Due to the dynamic nature of ColdFusion websites, you may find that some of your pages end up with a long string of url parameters that make the pages less then intuitive and user friendly. Along with looking out for your human visitors, you may also want to optimize your dynamic URLs for more favorable search engine treatment. While this recipe is not the place to debate search engine optimization techniques, we are running under the assumption that search engines seem to prefer more "human readable" URLs over dynamic URL strings.

Lets take a look at some sample URLs:

`http://www.coldfusioncookbook.com/index.cfm?event=faq`

works much better as:

`http://www.coldfusioncookbook.com/faq`

and

`http://www.coldfusioncookbook.com/index.cfm?event=showentry&id=1`

works much better as:

`http://www.coldfusioncookbook.com/entry/1/How-do-I-mail-the-contents-of-a-form?`

So how can you make your URLs more readable? 1) Write some custom code using the `cgi.path_info` variable, or 2) Have your web server do the work for you.

1) By using `cgi.path_info` and a little custom code, you can have ColdFusion parse down your more complex URLs in favor of something more simple.

It's really two parts. First we have to recognize the "weird" URL form - and once we do - we then parse it.

To begin with - whenever a URL comes in with the form, `http://host/filename.cfm/stuff/at/the/end`, your web server will recognize that "filename.cfm" is the file you want. It will then take the "extra" stuff and store it in a CGI variable, `path_info`. Sometimes - this CGI variable will also contain the filename. Luckily, Michael Dinowitz wrote a nice little article showing sample regex to "clean" this value. I don't seem to see a "direct" link to his article, but it on the House of Fusion website. (Look for the article, "Search Engine Safe (SES) URLs.") In this article he has a full blown UDF for dealing with the values, but I'm going to focus just on the regex. This example below shows it in action:

```
<cfset pathInfo = reReplaceNoCase(trim(cgi.path_info), "+\.cfm/? *", "")>
<cfoutput>
cgi.path_info=#cgi.path_info#<br>
stripped: #pathInfo#
</cfoutput>
```

You don't have to worry too much about the regex, it basically just handles removing any potential filename from the CGI variable. I'm not seeing any filename on my Apache or IIS server, but I know I've seen it in the past.

At this point we have a `pathInfo` variable that will store any information that added to the end of our filename. How do we parse this? Obviously you have a ColdFusion list using the `/` character are a delimiter. In my example above, `http://host/filename/stuff/at/the/end`, my `pathInfo` variable would have: `"/stuff/at/the/end"`. How I parse that is up to the application. In BlogCFC, I check the length of the value (using `listLen` and `/` as the delimiter) to make sure the length is 4. The first three values refer to the date and the last item refers to the alias.

You may want to use a format that is like typical URL variables. Something like: `http://host/filename.cfm/product/323`. In this form, the URL is simply another way of saying: `http://host/filename.cfm?product=323`. To parse this form, I would have to loop over the list and set URL variables. Here is a sample that will do that:

```
function parseSES() {
var pathInfo = reReplaceNoCase(trim(cgi.path_info), '+\.cfm/? *', '');
var i = 1;
var lastKey = "";
var value = "";

if(not len(pathInfo)) return;

for(i=1; i lte listLen(pathInfo, "/"); i=i+1) {
value = listGetAt(pathInfo, i, "/");
if(i mod 2 is 0) url[lastKey] = value;
else lastKey = value;
}
//did we end with a "dangler"? if((i-1) mod 2 is 1) url[lastKey] = "";
return;
}
```

What are we doing here? As I mentioned before, we begin by looking for stuff after the final slash. If we find nothing, we exit the function. (Normally a UDF returns something. A return statement by itself just means to leave the function without returning anything at all.)

Next we treat the value as a list and loop over it. We want to do things in twos - in other words, the first item is a variable, the second is a value. We simply check our list counter, `i`, and on odd numbers, we store the value as "lastKey", and on even numbers, we write to the URL scope. (UDFs should never directly access variables outside their own scope. Except when they should. :) This code assumes an even number of values. So what happens if the `pathInfo` variable is odd? (Ex: `/products/5/fo`) We treat this then as a "empty" variable and create the value in the URL scope with an empty string. This could be used as a flag. So for example, `/productid/5/short`, could mean set `url.productid` to 5, which is the database record to load, and "short" simply means show the shorthand version of the content.

2) As far as having your web server do the work for you, The solution is configure your web server so that the actual URL is intercepted and then displayed it as a more readable or friendly URL.

Apache

Apache has mod rewrite capabilities. You can set the rewrite rules in a `.htaccess` file. This file is good for whatever folder you place it in. So by placing the following code in a `.htaccess` file located at the root of your website, you would accomplish the url rewrites as shown in the sample URLs above.

```
RewriteEngine on
RewriteRule faq /?event=faq [PT]
RewriteRule entry/([0-9]+)/.* /?event=showentry&id=$1 [PT]
```

IIS

IIS does not have built in rewrite functionality, but you can add it with Ionic's free ISAPI Rewrite Filter : <http://cheeso.members.winisp.net/IIRF.aspx>

Credit Note: Raymond Camden helped write part of this entry.

This question was written by [Jeremy Petersen](#).  
It was last updated on July 6, 2006 at 11:03:44 AM EDT.

### **CFML Referenced**

[<cfoutput>](#)

## How can I limit the size of a file when uploading with cfile?

You can use the `cgi.content_length` to find the file size. If the `cgi.content_length` is less than or equal to the maximum size specified the file is uploaded (saved) to the server.

```
<cfset fileSizeLimit = 60000 />
<cffif cgi.content_length LTE fileSizeLimit>
  <cfile action="UPLOAD"
  filefield="Filename"
  destination="#thisDirectory#"
  nameconflict="OVERWRITE"
  accept="image/gif, image/jpeg">
  <center>SUCCESS!</center>
</cffif>
<cfelse>
  <cfoutput>
  <center>
  Your file size of #cgi.content_length# is too big!
  <br>
  The maximum size allowed is #fileSizeLimit#.
  </center>
  </cfoutput>
</cfabort>
</cffif>
```

Please note with this example the file is still uploaded as a temporary file, but the file is not saved. File sizes can also be restricted in the ColdFusion Administrator by using the setting, "Maximum size of post data." However, the example above gives more granular control.

This question was written by [Stan Winchester](#).

It was last updated on January 30, 2006 at 7:28:06 AM EST.

### CFML Referenced

[<cfoutput>](#)

[<cfelse>](#)

[<cfabort>](#)

[<cffif>](#)

## How can I make a form submit to itself without hardcoding the file name?

The ColdFusion CGI variable CGI.SCRIPT\_NAME contains the name of page being executed, so you can use:

```
<cfoutput>  
<form action="#CGI.SCRIPT_NAME#" method="post">  
</cfoutput>
```

Note the use of `<cfoutput>` around the form tag. This is required unless you are using `<cfform>`.

This question was written by [Ben Forta](#).

It was last updated on January 9, 2006 at 7:09:24 AM EST.

### CFML Referenced

[<cfoutput>](#)

[<cfform>](#)

## How can I prevent a browser from caching my page?

By default, browsers will try to cache the contents of a page. Because of their dynamic nature, most ColdFusion pages will automatically force a browser to reload them on each visit. However, you may still want to keep the contents of a more static page (for security or other reasons) from being cached. In these types of situations, you can use the [<cfheader>](#) tag or the html `<meta>` tag.

Place the following three [<cfheader>](#) tags at the top of a page to keep it from being cached:

```
<cfheader name="cache-control" value="no-cache, no-store, must-revalidate">
<cfheader name="pragma" value="no-cache">
<cfheader name="expires" value="#getHttpTimeString(now())#>
```

You can also use set a meta tag for content expiration

```
<meta http-equiv="expires" content="#<cfoutput>#getHttpTimeString(now())#</cfoutput>#>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on February 7, 2006 at 10:50:01 AM EST.

### CFML Referenced

[now\(\)](#)  
[<cfoutput>](#)  
[<cfheader>](#)

## How can I prevent empty query values from creating empty table cells?

If you place the contents of a ColdFusion query in an HTML table, any blank query value may show up as an empty cell with no borders in the table. This is not a bug in ColdFusion, but rather a result of how the browser renders the a cell that looks like this:

```
<d></d>
```

To prevent this, you can check for an empty value and display a non-breaking space instead:

```
<d>  
<cfif theValue is "">&nbsp;</cfif><cfelse>#theValue#</cfif>  
</d>
```

This question was written by [Raymond Camden](#).  
It was last updated on May 23, 2006 at 10:57:32 PM EDT.

### CFML Referenced

[<cfelse>](#)  
[<cfif>](#)

## How can I prevent SQL injection attacks?

SQL injection attacks occur when a client manipulates a web page to pass invalid data to a query. This can be down to force errors, bypass security, or even delete data. The [<cfqueryparam>](#) tag prevents SQL injection by binding values into the query; the bound values cannot be interpreted as SQL. It also results in faster queries.

```
<cfquery name="QCheckUser" datasource="blahblah">
SELECT *
FROM USERS
WHERE username = '#FORM.username#'
AND password = '#FORM.password#'
</cfquery>
```

becomes

```
<cfquery name="QCheckUser" datasource="blahblah">
SELECT *
FROM USERS
WHERE username = <cfqueryparam cfsqltype="cf_sql_varchar" value="#FORM.username#">
AND password = <cfqueryparam cfsqltype="cf_sql_varchar" value="#FORM.password#">
</cfquery>
```

In general, [<cfqueryparam>](#) should be used whenever a dynamic attribute is specified in a query.

This question was written by [James Holmes](#).

It was last updated on January 19, 2006 at 4:14:46 PM EST.

### CFML Referenced

[<cfquery>](#)

[<cfqueryparam>](#)

## How can I read a simple text file, processing each line of the file?

You can use ColdFusion lists for this, using the newline characters as the delimiter. Here is an example after a text file has been read using `<cffile>` where the variable is "myList":

```
<cfloop list="#myList#" index="aRow" delimiters="#chr(10)#chr(13)#">
The current row is #aRow# <br />
</cfloop>
```

Different operating systems will use different delimiters for files, but ColdFusion will treat multiple delimiters (as used in the example above) as multiple *possible* delimiters. Therefore this code should be able to read any text file.

This question was written by [Hal Helms](#).  
It was last updated on February 19, 2006 at 2:54:28 PM EST.

### CFML Referenced

[<cfloop>](#)  
[<cffile>](#)

## How can I read and write ColdFusion .log files?

ColdFusion log files are a form of text files that end in the .log extension and must be saved inside of the ColdFusion log directory. Although you can use [<cffile>](#) to read and write ColdFusion log files by hand, ColdFusion provides tools designed just for this specific task. To read the files, use the ColdFusion administrator (under Debugging & Logging), and to write/update the files, use the [<cflog>](#) tag. These handy shortcuts will save you most of the work associated with manipulating ColdFusion log files.

Using [<cflog>](#) is easy. You set the contents of the message to be written to the log file by way of the required text attribute. If you wish to use one of the default ColdFusion log files, you set the log attribute equal to either application or scheduler, depending on which of the two logs you wish to write to. If you wish to write to a custom log file, then you set the file attribute to the name (not including the .log extension) of the log file you wish to use. The type attribute is used to set the severity of the log entry. Valid values for this attribute include: information (default), warning, error, and fatal Information. Finally, the Optional application attribute can be set to either yes (default), or no. If application is set to yes, then the application name will be added to the log file.

The following sample code could be used to add a dynamic entry (based on form data) to a custom log file every time a new member registered on your site:

```
<cflog file="newUserLog"
application="No"
text="User #form.userName# joined the site.">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on February 2, 2006 at 5:28:20 PM EST.

### CFML Referenced

[<cflog>](#)  
[<cffile>](#)

## How can I read, set, or delete Windows system registry data?

Anyone who has worked with the Windows platform has probably encountered the system registry at one time or another. The system registry is a database of sorts, organized in a tree like fashion. The system registry contains data on just about everything in the system, including hardware, software, and users. It is important to note that because the system registry is Windows platform specific, ColdFusion only supports the [<cfregistry>](#) tag on the Windows platform.

The registry hierarchy tree is organized into two basic units: keys and values. Keys make up the branches in the tree, and values are the actual data (in the form of name value pairs). The [<cfregistry>](#) tag allows you to read, set (insert or update), and delete registry data.

[<cfregistry>](#) gives you two methods for reading (retrieving) keys and values from the registry. The first method will allow you to retrieve all of the key and value information for an entire branch. This is accomplished by setting the action attribute to getAll. The second method will only get the information for the specific entry you are targeting. This is accomplished by setting the action attribute to get.

The following example will look up information about all of the ODBC connections (the HKEY\_LOCAL\_MACHINE\Software\ODBC\ODBC.INI key) setup on the ColdFusion server machine:

```
<cfregistry action="getAll"
branch="HKEY_LOCAL_MACHINE\Software\ODBC\ODBC.INI"
name="GetODBC"
type="any"
sort="entry asc">

<table border="1">
<tr>
<td>Entry</td><td>Type</td><td>Value</td>
</tr>
<cfoutput query="getODBC">
<tr>
<td>#Entry#</td><td>#Type#</td><td>#Value#</td>
</tr>
</cfoutput>
</table>
```

Setting registry keys and values is accomplished by setting the action attribute to set. Note that the type attribute can be set to key, dword, or string (default) depending on what type of registry data you need to set. The following code shows how to use [<cfregistry>](#) to set a registry value:

```
<cfregistry action="set"
branch="HKEY_LOCAL_MACHINE\Software\Macromedia\ColdFusion\CurrentVersion"
type="key"
entry="test">
```

Deleting registry key or values is accomplished by setting the action attribute to delete. The following code shows how to use [<cfregistry>](#) to delete a registry value:

```
<cfregistry action="delete"
branch="HKEY_LOCAL_MACHINE\Software\Macromedia\ColdFusion\CurrentVersion"
entry="test">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on February 2, 2006 at 5:30:13 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[<cfregistry>](#)

## How can I share cookies between ColdFusion and JavaScript?

Sharing Cookies between ColdFusion and JavaScript is an easy way to pass data back and forth between the 2 technologies. A cookie is a cookie- regardless of if it was set by ColdFusion, Java, .net, or JavaScript. As long as you know the cookie name (including exact case in many situations) you can access and manipulate the cookie. With that said, the biggest trick to sharing cookies between ColdFusion and JavaScript is to remember that ColdFusion ignores case, but JavaScript does not. To use a JavaScript cookie in ColdFusion case does not matter. But to use a ColdFusion cookie in JavaScript, you need to reference the cookie name all in caps.

The following code is broken into two pages. The first page sets two cookies. The first cookie is set via JavaScript, and the second cookie is set via ColdFusion. The Second page then uses ColdFusion and JavaScript to display the contents of the cookie that was set by the other language:

```
<!-- Page1.cfm -->
<script language=javascript>
<!--
//This function will set a JavaScript cookie function setCookie(name, value) {
//build an expiration time 1 hour into the future var expDate = new Date()
expDate.setTime(ExpDate.getTime() + 60*60*1000);

//set the cookie document.cookie = name + "=" + escape(value) + ";" + expDate.toGMTString();
}

setCookie('jSCookie', 'JavaScript!');
// --> </script>
<cfcookie name="cFCookie" value="ColdFusion!" expires="never">

<!-- Page2.cfm -->
<script language=javascript>
<!--
// This function will return the value of a JavaScript cookie function getCookie(name) {
//init output var output = null;
//append ; to end so we can calculate end of cookie text var myCookie = "" + document.cookie + ";";
//append = to cookie name so any additional text is the cookie value var search = "" + name + "=";
// init search start location var begin = myCookie.indexOf(search);
//init search end location var end;
//loop over cookie text and pull out the value we want if (begin != -1) {
begin += search.length;
end = myCookie.indexOf(";", begin);
output = unescape(myCookie.substring(begin, end));
}
return output;
}

alert(getCookie('CFCOOKIE'));
// --> </script>
<cfoutput>#cookie.jSCookie#</cfoutput>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on April 3, 2006 at 6:25:11 PM EDT.

### CFML Referenced

[<cfoutput>](#)

## How can I tell if a user has JavaScript enabled?

Because ColdFusion is a server side technology and JavaScript is a client side technology, you will need to use a 2 page check to see if JavaScript is enabled. Your first page will perform the "is JavaScript enabled" test; your second page will display or record the results (save in a session variable etc).

One way to accomplish this task would be to set a cookie in JavaScript, and then test for the existence of this cookie via ColdFusion. If the cookie exists, then you know JavaScript is enabled.

```
<!-- Page1.cfm -->
<script language="JavaScript">
<!--
function setCookie(name, value) {
var expDate = new Date()
expDate.setTime(expDate.getTime() + 60*60*1000);
document.cookie = name + "=" + escape(value) + ";" + expDate.toGMTString();
}

setCookie('JSCookie', 'true!');
// --> </script>
<!-- Page2.cfm -->
<cfoutput>#cookie.JSCookie#</cfoutput>
```

Another way to test for JavaScript is to use a JavaScript redirect. If the browser supports JavaScript, it will be redirected. You could also use a HTML Meta redirect to catch all instances that ignored the JavaScript redirection, and redirect them to a set of non-JavaScript enabled pages. The code would look as follows:

```
<script language="JavaScript">
<!-- Begin script
window.location.replace("hasJS.cfm");
// End script --> </script>
<html>
<head>
<META HTTP-EQUIV=REFRESH CONTENT="0;URL=noJS.cfm">
</head>
</html>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on June 13, 2006 at 10:31:38 AM EDT.

### CFML Referenced

[<cfoutput>](#)

## How can I use ColdFusion to check a mail account?

ColdFusion comes with a [<cfpop>](#) tag that allows you to both read and delete email from a POP based server. For example:

```
From: #from#  
Subject: #subject#  
Sent: #date#  
  
#paragraphformat(body)#
```

---

The [<cfpop>](#) can also be used to download attachments and delete email from the server.

This question was written by [Raymond Camden](#).  
It was last updated on June 12, 2006 at 4:58:54 PM EDT.

### CFML Referenced

[<cfpop>](#)

## How can I use session variables to determine the date of a user's last visit?

While the client scope has this functionality built in (`client.lastVisit`), with the session scope you will need to set your own variable to track a user's last visit date. On your `Application.CFC` page's `onRequestStart()` method, set a session variable with the current date time. This session variable will then be updated on each new page request.

```
<cffunction name="onRequestStart" returnType="boolean">
  <cfset session.lastVis = now()>
</cffunction>
```

Remember that session variables will usually timeout much sooner than client variables (based on server or local settings), and once a session is timed out, you will lose that user's last visit session variable. With this in mind, you may want to consider adding some code to your `Application.CFC` page's `onSessionEnd()` method to write the session data to a database or file on session end.

```
<cffunction name="onSessionEnd" returnType="boolean">
  -insert code to persist session.lastVis to DB or File-
</cffunction>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on June 30, 2006 at 1:46:48 PM EDT.

### CFML Referenced

[now\(\)](#)  
[<cffunction>](#)

## How can I work with remote HTML forms?

The nature of working with HTML form submission (or handler) pages is such that by default they do not make any distinction in regards to the source of the form data. In other words, if a form handler page is expecting to receive the posting of a name and an address value, it does not know (or care) if the form data came from a page on your server, or from another server. Take for example many of the web based credit card merchant account systems. The typical scenario for these systems is for you to provide a form on your website that contains the form data you have collected for a shopping transaction. You collect information such as the customer's name, billing address, and credit card number in a form. You then post this form data to a specifically designed page on the credit card processing company's website. This processing page performs all the magic of the credit card transaction, and then informs the user of the outcome. The big problem with this scenario is that your customer has now left your website, and this is not the ideal solution or user experience.

Amongst other functionality, the ColdFusion [<cfhttp>](#) tag provides a way for you to post form data to a remote form, and then retrieve the results that are outputted from the form results page into a ColdFusion variable. You can then process the results variable (the form submission outcome) within your ColdFusion application. So with our above credit card processing example, we could let [<cfhttp>](#) leave our website, post the form data, retrieve the results of the credit card transaction, then seamlessly display the outcome to the user, all from within our website.

In order to use [<cfhttp>](#) to post to a remote form, you need to know the full address of the template you will be posting to. You also need to know each expected form parameter, and include a [<cfhttpparam>](#) tag for each of these parameters. A sample [<cfhttp>](#) form post would look as follows:

```
<cfhttp url="http://127.0.0.1/formHandler.cfm" method="post">
<cfhttpparam type="formField" name="creditCardNumber" value="#localCCNum#">
<cfhttpparam type="formField" name="userName" value="#localName#">
</cfhttp>
```

Notice how we can pass in any value (in this case we are using the dynamic contents of ColdFusion variables) to the remote form via the [<cfhttpparam>](#) value parameter.

The results of the form post will be contained in the `cfhttp.fileContent` variable. You can access this variable to parse out and redisplay any data. In our credit card processing scenario, we would probably want to parse out the transaction status message so we could dynamically show the user a success or failure message.

[<cfhttp>](#) is a very powerful tool, and it can be used for many things. Some other common [<cfhttp>](#) tasks include grabbing current weather and stock quotes. One final thing to remember about [<cfhttp>](#): you are not limited to ColdFusion pages. A form is a form, so feel free to use [<cfhttp>](#) to communicate with any other flavor of form, including those made with JSP, ASP, PHP, etc.

This question was written by [Jeremy Petersen](#).  
It was last updated on February 1, 2006 at 9:23:43 AM EST.

### CFML Referenced

[<cfhttpparam>](#)  
[<cfhttp>](#)

## How do I access one row in a query?

You can access elements of a query result set with array notation:

```
<cfoutput>
#myQuery.columnName[5]#
</cfoutput>
```

This will give you the value of "columnName" in the 5th row of the record set. If you leave off the array notation, and are not within a query-based loop, then the result will be from the first row.

This question was written by Christoph Schmitz.  
It was last updated on March 23, 2006 at 6:49:19 AM EST.

### CFML Referenced

[<cfoutput>](#)

## How do I add an individual entry to a Verity collection?

As you know, the [<cfindex>](#) tag allows you to populate a Verity collection. This data can come from the file system (by using either a directory or just a file) or with a custom query.

Once the collection is populated, you do not need to clear the collection to do minor updates. You can use [<cfindex>](#) to add content as well. So for example, this code block will add a new file to the collection:

```
<cfindex action="update" collection="docs" type="file" key="c:\mydocs\new.pdf">
```

You can also use [<cfindex>](#) to remove an item as well. If the file used above was deleted, the following code should be used to keep the collection in sync:

```
<cfindex action="delete" collection="docs" type="file" key="c:\mydocs\new.pdf">
```

This question was written by [Raymond Camden](#).  
It was last updated on April 18, 2006 at 6:58:41 AM EDT.

### CFML Referenced

[<cfindex>](#)

## How do I alternate row colors in a table?

Two functions are used in one expression which allow for a clean implementation of alternating row colors. [Iif\(\)](#) takes three parameters (condition,expression1,expression2) and evaluates either the first expression or the second expression depending if the current row is even or odd. [DE\(\)](#) takes one parameter which is an expression and delays the evaluation of the expressions.

The following example uses CSS to style the alternating rows of content:

```
<cfquery name="myTest" datasource="testDatasource">
  SELECT *
  FROM table
</cfquery>

<style type="text/css">
tr.lightrow td {
background-color: #FFFFFF;
}
tr.darkrow td {
background-color: #EFEFEF;
}
</style>

<table>
<cfoutput query="myTest">
  <tr class="#iif(currentrow MOD 2,DE('lightrow'),DE('darkrow'))#">
    <td> 1 </td>
    <td> myField </td>
  </tr>
</cfoutput>
</table>
```

This question was written by Cory Toth.  
It was last updated on January 9, 2006 at 12:10:49 PM EST.

### CFML Referenced

[<cfquery>](#)  
[<cfoutput>](#)  
[iif\(\)](#)  
[de\(\)](#)

## How do I create a vCard/ vCalender link and have it download to MS Outlook?

Use `<cfcontent>` and `<cfheader>` to set the appropriate MIME and header types, then include the appropriate fields and data. If MS Outlook is installed on the client, it will receive the data as a vCard!

The following sample static vCard was found out on the internet. It can be used as a template to replace with your own dynamic data.

```
<cfcontent type="text/x-vCalendar">
<cfheader name="Content-Disposition" value="inline; filename=newAppointment.vcs">
BEGIN:VCARD
VERSION:3.0
N;charset=iso-8859-1;Geuzaine,Christophe
BDAY,value=date:1973-02-06
EMAIL,type=HOME;geuz@geuz.org
URL,type=HOME;http://www.geuz.org
TITLE,charset=iso-8859-1;Postdoctoral Scholar
ORG,charset=iso-8859-1;Caltech, Applied and Computational Mathematics
ADR,type=WORK,charset=iso-8859-1;;;1200 E California Blvd,Pasadena,CA,91125;USA
TEL,type=WORK;1 626 395 4552
URL,type=WORK;http://www.acm.caltech.edu
END:VCARD
BEGIN:VCARD
VERSION:3.0
N;charset=iso-8859-1;Knudson,Donald Ernest
BDAY,value=date:01-29
TEL,type=HOME;+01-(0)2-234.56.78
EMAIL,type=HOME;duck@novosi.umi.gnu
NOTE,charset=iso-8859-1;1952 Permafrost Press Award winner
END:VCARD
BEGIN:VCARD
VERSION:3.0
N;charset=iso-8859-1;Knudson,Daffy Duck (with Bugs Bunny and Mr. Pluto)
NICKNAME,charset=iso-8859-1;gnat and gnu and pluto
BDAY,value=date:02-10-11-05;01-01
TEL,type=HOME;+01-(0)2-765.43.21
TEL,type=CELL;+01-(0)5-555.55.55
ACCOUNT,type=HOME;010-1234567-05
ADR,type=HOME,charset=iso-8859-1;;;Haight Street 512;Novosibirsk;;80214;Gnoland
TEL,type=HOME;+01-(0)2-876.54.32
ORG,charset=iso-8859-1;University of Novosibirsk, Department of Octopus Parthenogenesis
END:VCARD
BEGIN:VCARD
VERSION:3.0
N;charset=iso-8859-1;Knudson;Bip B.
NICKNAME,charset=iso-8859-1;road runner
BDAY,value=date:02-27
ADR,type=HOME,charset=iso-8859-1;;;Somewhere close to a falling rock;;;
EMAIL,type=HOME;bip_bip@free.prov.gnu
END:VCARD
BEGIN:VCARD
VERSION:3.0
N;charset=iso-8859-1;Microknud Corp.;
ADR,type=WORK,charset=iso-8859-1;;;Haight Street 513;Novosibirsk;;80214;Gnoland
TEL,type=WORK;+01-(0)2-465.83.99
TEL,type=FAX;005.79.00
URL;http://foe.com/index.html
END:VCARD
BEGIN:VCARD
VERSION:3.0
N;charset=iso-8859-1;The Knudsoft Company;
EMAIL;knud@knudsoft.com
URL;http://foe.com/index.htm
END:VCARD
BEGIN:VCARD
VERSION:3.0
N;charset=iso-8859-1;Knudsoft (RS.2 Computer Room);
TEL,type=WORK;+01-(0)2-434.23.23
END:VCARD
```

This question was written by [Jeremy Petersen](#).

It was last updated on June 2, 2006 at 5:31:52 PM EDT.

### CFML Referenced

[<cfcontent>](#)

[<cfheader>](#)

## How do I create an array with more than three dimensions?

ColdFusion lets you directly create arrays with up to three dimensions using the [arrayNew\(\)](#) function. If you want to create a larger array, you can use multiple [arrayNew\(\)](#) statements.

```
<cfset foo = arrayNew(3)>
<cfset foo[1][1][1] = arrayNew(3)>
<cfset foo[1][1][1][1][1] = "this is a test">

<cfoutput>#foo[1][1][1][1][1]#</cfoutput>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on May 15, 2006 at 12:35:55 PM EDT.

### CFML Referenced

[<cfoutput>](#)  
[arrayNew\(\)](#)

## How do I delete a folder and all files and subfolders beneath it?

The tag has a delete action, but this will throw an error if there is anything in the directory. To remove a directory and everything inside it, simply use the recurse attribute:

```
<cfset dir = "c:\temp">  
<cfdirectory action="delete" directory="#dir#" recurse="true">
```

This question was written by [Raymond Camden](#).  
It was last updated on April 6, 2006 at 5:56:26 PM EDT.

## How do I determine if a number is even or odd?

An even number is any number that can be divided by 2 with no remainder. ColdFusion provides a function, `mod`, that returns the remainder of a division operation. To determine if a number is even, simply see if the value, modded by 2, returns 0:

```
<cfset x = 5>
<cfif x mod 2 is 0>
    The number is even.
</cfif>
<cfif x mod 2 is 1>
    The number is odd.
</cfif>
```

If the remainder is 1, then the number is odd.

This question was written by [Raymond Camden](#).

It was last updated on February 3, 2006 at 8:01:44 AM EST.

### CFML Referenced

[<cfelse>](#)  
[<cfif>](#)

## How do I display query results in an N-column table layout?

Displaying query information in a N-column table can be done with just a bit of logic. This first bit of code just creates a query object for the example, you don't need this if you have your own data to work with.

```
<cfscript>
stateList =
"Alabama,Alaska,Arizona,Arkansas,California,Colorado,Connecticut,Delaware,Florida,Georgia,Hawaii,Idaho,Illinois,Indiana,Iowa,Kansas,Kentucky,Louisiana,Maine,Maryland,Massachusetts,Michigan,Minnesota,Mississippi,Missouri,Montana,Nebraska,Nevada,New Hampshire,New Jersey,New Mexico,New York,North Carolina,North Dakota,Ohio,Oklahoma,Oregon,Pennsylvania,Rhode Island,South Carolina,South Dakota,Tennessee,Texas,Utah,Vermont,Washington,West Virginia,Wisconsin,Wyoming";
myQuery = QueryNew("stateName");
for(i=1;LTE ListLen(stateList);i = i + 1) {
    queryAddRow(myQuery);
    querySetCell(myQuery,"stateName",ListGetAt(stateList,i));
}
</cfscript>
```

Here is where you set the number of columns you want your data to be outputted in, you can adjust it very easily:

```
<!-- adjust the number of columns to your needs -->
<cfset columnCount = 4>
```

Finally the table output:

```
<table border="1">
<tr>
<cfoutput query="myQuery">
<td>#stateName#</td>
<cfif currentRow MOD columnCount EQ false AND currentRow NEQ recordCount>
<tr>
<tr>
<cfelseif currentRow MOD columnCount EQ false AND currentRow EQ recordCount>
<tr>
<cfelseif currentRow MOD columnCount AND currentRow EQ recordCount>
<cfset columnsLeft = (Ceiling(currentRow/columnCount)*columnCount) - currentRow>
#RepeatString("<td> &nbsp;</td>",columnsLeft)#
<tr>
</cfif>
</cfoutput>
</table>
```

Basically the main thing going on here is math, specifically, determining if we have completed a "row" based on our current query row and the number of columns we specified. Note at the end we "fill" the table with blank cells. This completes our table and will help it render properly.

This question was written by [Erik Goodlad](#).

It was last updated on February 22, 2006 at 7:21:18 AM EST.

### CFML Referenced

[<cfoutput>](#)

[<cfif>](#)

[<cfscript>](#)

## How do I do a server-side relocation?

ColdFusion provides a tag to handle relocating the user, [<cflocation>](#). This tag returns header information to the browser that tells it to load a new URL. If you want to do a completely server-side relocation, you must use one of the underlying Java methods available in ColdFusion:

```
<cfset getPageContext().forward('url_here') />
```

This question was written by [Hal Helms](#).  
It was last updated on February 19, 2006 at 2:43:58 PM EST.

### CFML Referenced

[<cflocation>](#)  
[getPageContext\(\)](#)

## How do I find a value in a list?

ColdFusion provides four functions that can help you find a value in a list. The first two are related: [listFind\(\)](#) and [listFindNoCase\(\)](#). Both functions will search a list for a value. The first is case sensitive while the second will ignore case.

```
<cfset list = "Raymond,Jacob,Lynn,Noah,Jeanne">  
<cfif listFindNoCase(list, "jacob")>  
  Jacob is in the list.  
</cfif>
```

The above code snippet will find a match on the word "jacob" even though the case does not match.

The next two related functions are [listContains\(\)](#) and [listContainsNoCase\(\)](#). These functions allow for partial matches. So for example:

```
<cfset list = "Raymond,Jacob,Lynn,Noah,Jeanne">  
<cfif listContainsNoCase(list, "Ray")>  
  There is a Ray in the list.  
</cfif>
```

This code snippet will display a result since "Ray" partially matches "Raymond" in the list. In general you will probably never use [listContains\(\)](#) since you almost always want to match an entire list item, not a partial one.

This question was written by [Raymond Camden](#).

It was last updated on January 18, 2006 at 3:21:27 PM EST.

### CFML Referenced

[listContains\(\)](#)  
</cfif>  
[listContainsNoCase\(\)](#)  
[listFindNoCase\(\)](#)  
[listFind\(\)](#)

## How do I find a value in an array?

ColdFusion does not provide a built-in way to search an array for values. However, you have a few options. You can convert the array to a list using [arrayToList\(\)](#). Once you have done that, you can use [listFind\(\)](#), [listFindNoCase\(\)](#), [listContains\(\)](#), or [listContainsNoCase\(\)](#) to search the array.

You can also find UDFs at [CFLib](#) that will search an array. You can find both an [arrayFind\(\)](#), as well as an [arrayFindNoCase\(\)](#).

This question was written by [Raymond Camden](#).

It was last updated on January 18, 2006 at 3:27:07 PM EST.

### CFML Referenced

[listContains\(\)](#)  
[listContainsNoCase\(\)](#)  
[listFindNoCase\(\)](#)  
[listFind\(\)](#)  
[arrayToList\(\)](#)

## How do I find out if a specific file or directory exists on my ColdFusion server?

The [directoryExists\(\)](#) function takes an absolute path as its only parameter. It will then test for the existence of that absolute path on the server. The function returns YES or NO. Sample code for the [directoryExists\(\)](#) function would look as follows:

```
<cfset testDirectory = "C:\foo">
<cfif directoryExists(testDirectory)>
Yes, #testDirectory# exists on the server.
<cfelse>
No, #testDirectory# does not exist on the server.
</cfif>
</cfoutput>
```

The [fileExists\(\)](#) function works the same way as the [directoryExists\(\)](#). You pass it in an absolute path, and it returns YES or NO depending on if the path exists. It is important to note that if you are testing for the existence of a file, you many first want to make sure the directory exists. Sample use of the [fileExists\(\)](#) function would be as follows:

```
<cfset testFile = "C:\foo\foobar.cfm">
<cfoutput>
<cfif fileExists(testFile)>
Yes, #testFile# exists on the server.
<cfelse>
No, #testFile# does not exist on the server.
</cfif>
</cfoutput>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on January 26, 2006 at 9:13:55 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[<cfelse>](#)  
[<cfif>](#)  
[fileExists\(\)](#)  
[directoryExists\(\)](#)

## How do I find the size of a directory?

Use the [<cfdirectory>](#) tag, and then do a query of queries on the results:

```
<cfdirectory
  directory="c:\cfusionmx"
  action="list"
  name="cDir"
  recurse="true">

<cfquery dbtype="query" name="dirSize">
  select sum(size) as size from cDir
</cfquery>

<cfset sizeMb = dirSize.size/1000000>

<cfoutput>#numberFormat(sizeMb, ".99")#</cfoutput>
```

In the example above, the result is modified to return a value in megabytes. Also note the use of `recurse="true"` in the [<cfdirectory>](#) tag. This will return all the files including those beneath the directory specified. If you only want the size of the files in the directory itself, change `recurse` to `false`.

This question was written by [Jacob Munson](#).

It was last updated on March 31, 2006 at 1:54:23 PM EST.

### CFML Referenced

[<cfquery>](#)

[<cfoutput>](#)

[<cfdirectory>](#)

## How do I force a file to download instead of displaying inline in IE, Firefox and other browsers?

You can force a file to download by using a combination of the [<cfheader>](#) and [<cfcontent>](#) tags. We'll use a jpg for this example. It will work for any file type however.

```
<cfheader name="content-disposition" value="attachment;filename=example.jpg">  
<cfcontent type="image/jpeg" file="C:\files\example.jpg">
```

Note the use of filename in the [<cfheader>](#) tag. This allows you to give another name to the file being downloaded.

This question was written by Emmet McGovern.

It was last updated on January 24, 2006 at 7:02:18 AM EST.

### CFML Referenced

[<cfcontent>](#)

[<cfheader>](#)

## How do I format an Active Directory Timestamp?

Time in Active Directory is stored in a 64 bit integer that keeps track of the number of 100-Nanosecond intervals which have passed since January 1, 1601 (not to be confused with EPOCH, or with Active Directory's Generalized Time String). The 64 bit value uses 2 32bit parts to store the time.

This number, e.g 127944393687163952 can be converted to a local timestamp such as:

```
adTime = "127944393687163952";  
cfTime = DateConvert("utc2Local",DateAdd('n',adTime / (60*10000000),1/1/1601) );
```

This question was written by [Tariq Ahmed](#).

It was last updated on July 12, 2006 at 12:00:53 PM EDT.

## How do I get around a lack of constructors in CFCs?

A widely adopted practice is to create an "init" method that returns the object.

Example:

```
<cfcomponent displayName="Person">
<cffunction name="init" access="public" output="false">
<cfreturn this />
</cffunction>
</cfcomponent>
```

The init method can accept arguments and perform object initialization, if needed. Now, you can always create your objects like this:

```
<cfset joe = createObject('component', 'Person').init() />
```

This question was written by [Hal Helms](#).  
It was last updated on June 12, 2006 at 5:13:04 PM EDT.

### CFML Referenced

[<cfcomponent>](#)  
[<cffunction>](#)

## How do I get the directory for the current template?

Use the two ColdFusion functions [getCurrentTemplatePath\(\)](#) and [getDirectoryFromPath\(\)](#).

```
<cfset currentPath = getCurrentTemplatePath(>  
<cfset currentDirectory = getDirectoryFromPath(currentPath)>  
<cfoutput>This directory is #currentDirectory#</cfoutput>
```

The function, [getCurrentTemplatePath\(\)](#), will return the full path of the current template. By path we mean the directory and file name. The function, [getDirectoryFromPath\(\)](#), will then get just the directory from the path.

This question was written by [Stan Winchester](#).

It was last updated on January 26, 2006 at 6:57:12 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[getDirectoryFromPath\(\)](#)  
[getCurrentTemplatePath\(\)](#)

## How do I get the last modified date on a file?

Use [<cfdirectory>](#) with the filter extension:

```
<cfdirectory action="list" directory="C:\myDirectory\" name="myResult" filter="myFile.extension">  
<cfdump var="#myResult.dateLastModified#">
```

This question was written by [Jeff Houser](#).  
It was last updated on February 15, 2006 at 9:20:37 AM EST.

### CFML Referenced

[<cfdirectory>](#)

## How do I get the SQL used to generate a query?

Most queries written in ColdFusion will contain one or more dynamic portions. If you want to get the SQL that was actually passed to the database, use the result attribute of the [<cfquery>](#) tag:

```
<cfset name = "e">
<cfquery name="getIt" datasource="cfartgallery" result="result">
select artistid
from artists
where lastname like <cfqueryparam cfsqltype="cf_sql_varchar" value="%#name%" maxlength="255">
</cfquery>
<cfdump var="#result#">
```

The result attribute returns a structure with keys that relate to the query. The keys that you would be concerned about are SQL and SQLParameters. The SQL key will contain the sql that was passed to the database. Each and every [<cfqueryparam>](#) tag that was used will be replaced with a question mark. The SQLParameters key will contain an array of values that correspond to each question mark.

This question was written by [Raymond Camden](#).  
It was last updated on May 20, 2006 at 3:11:44 PM EDT.

### CFML Referenced

[<cfquery>](#)  
[<cfqueryparam>](#)

## How do I get the username or domain from an email address?

The obvious use of lists in ColdFusion is to work with a lists of data. However what's cool about list functions is that they can be used for quick extraction of data.

You can consider an email address as a list which uses the @ character as a delimiter. Because of this, you can then use [listFirst\(\)](#) and [listLast\(\)](#) to quickly grab the two portions of the email address.

```
<cfset userName = listFirst(emailAddress, "@")>  
<cfset domainName = listLast(emailAddress, "@")>
```

ColdFusion's list functions use the comma character as the default delimiter. Notice in the code above that we explicitly tell ColdFusion to use the @ character instead.

This question was written by [Tariq Ahmed](#).

It was last updated on February 2, 2006 at 7:56:23 AM EST.

### CFML Referenced

[listFirst\(\)](#)

[listLast\(\)](#)

## How do I get the values from one column in a query?

If you need to retrieve the values from one column in a query, ColdFusion provides the [valueList\(\)](#) function. It will return every value from that column. Here is an example:

```
<cfquery name="getPeople" datasource="people">
select name, age, rank
from people
</cfquery>

<cfset allNames = valueList(getPeople.name)>
```

Note that valueList does not take a string, but the actual query and column variable.

This question was written by [Raymond Camden](#).

It was last updated on March 23, 2006 at 6:55:54 AM EST.

### CFML Referenced

[<cfquery>](#)  
[valueList\(\)](#)

## How do I know when a user's session ends?

You can run code when a user session ends by using the `onSessionEnd` method of the `Application.cfc` file. The following example will log to a file:

```
<cffunction name="onSessionEnd" returnType="void" output="false">
<cfargument name="sessionScope" type="struct" required="true">
<cfargument name="appScope" type="struct" required="false">

<cflog file="#arguments.appScope.applicationName#" text="Session ended.">
</cffunction>
```

Note that within the `onSessionEnd` method you cannot address the session or application scopes directly. Instead you reference them via the arguments automatically passed to the method. You can't output anything from this method since, obviously, no user is around to see the output.

Lastly - if you use J2EE session variables, you will notice that your session ends when you close the browser. Actually, your session is not ending. Instead, when you relaunch your browser, a new session is created. Therefore you cannot close your browser as a means to test this code. Instead, you must wait the complete amount of time it takes for sessions to expire in your application. (If not specified, it will default to a value set in the ColdFusion administrator - usually 20 minutes.)

This question was written by [Raymond Camden](#).  
It was last updated on January 16, 2006 at 12:58:29 AM EST.

### CFML Referenced

[<cffunction>](#)

## How do I mail the contents of a form?

One of the most common things a web site may require is a simple "Contact Us" or other form. Normally all you want to do is take the results of the form and email them to the site owner. If you want to quickly deploy a script to do this without all the fancy formatting, you can use the fact that ColdFusion treats form data as a structure. Because of this - there are some simple structure functions we can use to email the contents of the form.

```
<cfmail to="someone@yourorganization.com" from="someone@yourorganization.com" subject="Form Foo Submitted" wraptext="80">
<cflump item="field" collection="#form#">
<cfif field is not "fieldnames">
#field# = #form[field]#
</cfif>
</cflump>
</cfmail>
```

The code snippet above begins with a cfmail tag. Obviously you would change the addresses to match those of the people you want to mail. Next we use cflump with the item and collection attributes. These tell cflump to iterate over all the keys of the structure. In this case it will be the fields of the form. Notice that we skip the form field, "fieldnames." This is a special field that ColdFusion creates. It contains all the fields of the form. Since we don't need this, we don't print it.

This question was written by [Raymond Camden](#).

It was last updated on January 9, 2006 at 6:13:20 AM EST.

### CFML Referenced

[<cflump>](#)

[<cfif>](#)

[<cfmail>](#)

## How do I make a 301 redirect (permanently) from url1 to url2?

Permanently redirecting traffic using [<cfheader>](#) with statuscode="301" is the best way to ensure that your web-visitors and search engine spiders continue to find content that has permanently moved to a new location.

```
<cfheader statuscode="301" statustext="Moved permanently">  
<cfheader name="Location" value="http://www.newUrl.com">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on June 19, 2006 at 12:59:09 PM EDT.

### CFML Referenced

[<cfheader>](#)

## How do I make a template pause(sleep)?

There is no native sleep or pause function in CFML. But with CFMX you can easily leverage Java and take advantage of Java's sleep function. The following code snippet will sleep for 1,000 milliseconds (one second) between calls to [getTickCount\(\)](#).

```
<cfoutput>
Before Sleep: #getTickCount()#<br>
</cfscript>
<cfscript>
go_to = createObject("java", "java.lang.Thread");
go_to.sleep(1000); !sleep time in milliseconds
</cfscript>
After Sleep: #getTickCount()#
</cfoutput>
```

This question was written by [Steve Gustafson](#).  
It was last updated on February 22, 2006 at 12:00:25 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[<cfscript>](#)  
[getTickCount\(\)](#)

## How do I make sure a string is safe to use with JavaScript?

If you are dynamically populating a JavaScript variable, you may find that your code breaks with "unterminated string constant" or similar error messages. This is probably a case of your JavaScript variables containing characters that are considered to be "special" characters by JavaScript. You will need to "escape" these special characters so that JavaScript can process them.

Some common characters you need to be wary of include: newlines, carriage returns, and quotes. In order for JavaScript to handle these special characters, they must be escaped, or converted to JavaScript safe alternatives. JavaScript makes use of the \ characters to escape most special characters.

The following code sample from the CF Docs shows how to use the ColdFusion [jsStringFormat\(\)](#) function to make a string JavaScript safe:

```
<cfset stringValue = "An example string value with a tab chr(9), a newline (chr(10) and some ""quoted"" `text`">
<p>This is the string we have created:<br>
<cfoutput>#stringValue#</cfoutput>
</p>
<cfset jsStringValue = jsStringFormat(#stringValue#)>
<!-- Generate an alert from the JavaScript string jsStringValue. ---->
<script>
s = "<cfoutput>#jsStringValue#</cfoutput>";
alert(s);
</script>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on March 29, 2006 at 10:41:14 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[jsStringFormat\(\)](#)

## How do I output a query result set grouped by a specific field?

To generate this type of display, there are two key things to note in the code sample below. First, the field you wish to group by must appear in the ORDER BY clause of your query and this same field must be used as the "group" attribute for the first [<cfoutput>](#) tag in addition to the "query" attribute which tells the tag to loop.

Sample Code:

```
<cfquery datasource="bluedragon" name="q_getemployees">
SELECT employee.employeefirstname, employee.employeelastname, department.departmentname
FROM employee INNER JOIN department ON employee.departmentid =
department.departmentid
ORDER BY department.departmentname
</cfquery>

<cfoutput query="q_getemployees" group="departmentname">
<h1>#q_getemployees.departmentname#</h1>
<cfoutput>
#q_getemployees.employeefirstname# #q_getemployees.employeelastname#<br />
</cfoutput>
</cfoutput>
```

Output:

```
<h1>SAMPLE RESULT:</h1>
<h2>Development</h2>
Darin Kohles<br />
Eric Jones<br />
Colleen Cox<br />

<h2>Management</h2>
Steve Nation<br />
Ben Wakeman<br />

<h2>Sales</h2>
David Taylor-Klaus<br />
Beth Cooper<br />
```

This question was written by [Ben Wakeman](#).  
It was last updated on June 13, 2006 at 2:25:06 PM EDT.

### CFML Referenced

[<cfquery>](#)  
[<cfoutput>](#)

## How do I perform an XSLT transform?

Here's a simple example of transforming an XML file using an XSLT stylesheet that are both located in the same directory as the running ColdFusion script:

```
<cfset xmldoc = xmlParse(expandPath("input.xml"))>
<cffile action="read" file="#expandPath('transform.xml')#" variable="xmltrans">
<cfoutput>#xmlTransform(xmldoc, xmltrans)#</cfoutput>
```

This question was written by [Bif](#).  
It was last updated on June 13, 2006 at 1:33:21 PM EDT.

### CFML Referenced

[<cfoutput>](#)

## How do I perform trigonometric calculations?

ColdFusion provides functions for all of the common trigonometric calculations.

The [pi\(\)](#) function returns the mathematical constant Pi, accurate up to 15 digits:

```
<cfset testVar = pi()>
<cfoutput>#testVar#</cfoutput>
3.14159265359
```

The [sin\(\)](#) function takes an angle (in radians), and returns the sine of the angle (in radians):

```
<cfset testVar = sin(100)>
<cfoutput>#testVar#</cfoutput>
-0.50636564111
```

The [cos\(\)](#) function takes an angle (in radians), and returns the cosine of the angle (in radians):

```
<cfset testVar = cos(100)>
<cfoutput>#testVar#</cfoutput>
0.862318872288
```

The [tan\(\)](#) function takes an angle (in radians), and returns the tangent of the angle (in radians):

```
<cfset testVar = tan(100)>
<cfoutput>#testVar#</cfoutput>
-0.587213915157
```

The [asin\(\)](#) function takes a number between -1 and 1, and then returns the arcsine of that number.

```
<cfset testVar = asin(1)>
<cfoutput>#testVar#</cfoutput>
1.57079632679
```

The [acos\(\)](#) function takes a number between -1 and 1, and then returns the arccosine of that number.

```
<cfset testVar = acos(1)>
<cfoutput>#testVar#</cfoutput>
0
```

The [atan\(\)](#) function takes a number, and then returns the arctangent of that number.

```
<cfset testVar = atan(1)>
<cfoutput>#testVar#</cfoutput>
0.785398163397
```

To convert degrees to radians, multiply degrees by  $\pi/180$ .

```
<cfset myDegreesVar = .5>
<cfoutput>Radians = #myDegreesVar * pi()/180#</cfoutput>
Radians = 0.00872664625997
```

To convert radians to degrees, multiply radians by  $180/\pi$ .

```
<cfset myRadiansVar = .5>
<cfoutput>Degrees = #myRadiansVar * 180/pi()#</cfoutput>
Degrees = 28.6478897565
```

This question was written by [Jeremy Petersen](#).  
It was last updated on January 24, 2006 at 4:44:12 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[aCos\(\)](#)  
[pi\(\)](#)  
[sin\(\)](#)  
[atan\(\)](#)  
[cos\(\)](#)  
[aSin\(\)](#)  
[tan\(\)](#)

## How do I prevent a file from becoming corrupt due to simultaneous access?

While working with files, it is important to lock them for single threaded access. If you do not, it is possible that your application will try to perform simultaneous read or write operations on the file. This could cause all kinds of undesirable results, including file corruption. By wrapping all file access code in a [<cflock>](#) tag that is uniquely named for each file, you insure that your file data only be accessed by one process at a time.

The following code shows a sample use of a [<cflock>](#) tag that could be used to protect a file transaction:

```
<cflock name="dataFileLock" type="exclusive" timeout="30">
<!-- Insert file transaction -->
</cflock>
```

Note the name of the lock. By setting this to something related to the particular file you will be working with, you can better insure that the next portion of code that will work with this file will use the same lock.

This question was written by [Jeremy Petersen](#).

It was last updated on January 31, 2006 at 9:17:50 AM EST.

### CFML Referenced

[<cflock>](#)

## How do I process CF code contained in a string (eg. text field in DB)?

Unfortunately CFMX 7 does not have a built-in feature to accomplish this task. You can however pull this off with a few manual steps: 1) save the code to a temp file 2) [<cfinclude>](#) the temp file 3) Delete the temp file.

Be advised, this is not a recommend best practice! Why add dynamic file read writes if you can avoid it- this is a bad move for performance. Also consider the possible security hole of someone inserting malicious code into your processing engine. In other words, use this recipe at your own risk.

```
<!-- get cfml code string from DB -->
<cfquery
datasource="foo"
name="mQuery">
SELECT code
FROM myCode
WHERE id = 1
</cfquery>

<cfset tempFile = getTempFile("C:\CFusionMX7\wwwroot\test", "code")>

<!-- write file out -->
<cffile action="write"
file="#tempFile#"
output="#mQuery.code#">

<!-- include file -->
<cfinclude template="#GetFilePath(tempFile)#">

<!-- delete file -->
<cffile action="delete"
file="#tempFile#">

<p>This is after</p>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on May 31, 2006 at 2:31:30 PM EDT.

### CFML Referenced

[<cfquery>](#)  
[<cfinclude>](#)

## How do I read and write binary files?

Working with binary files is very similar to working with text files. The main difference is that by default, binary files are in a more complex format than standard text files. This means in order for ColdFusion to manipulate binary file data, you may need to use the [toBase64\(\)](#) and [toBinary\(\)](#) functions to convert data to and from binary format into a more manageable format.

In order to use [<cffile>](#) to read a binary file, you need to set the action attribute to readBinary. For example:

```
<cffile action="readBinary"
file="C:/button.gif"
variable="myBinaryFile">
```

This would store the contents of the binary file in the myBinaryFile variable. You could then use the [toBase64\(\)](#) function to convert the binary file into a format that ColdFusion could output and manipulate.

If you want to use [<cffile>](#) to write a binary file, you first need to make sure the data is in the proper format. If the data is not already in binary format, you can use the [toBinary\(\)](#) function to convert the data to binary. Once you have binary data, it is a simple matter of using [<cffile>](#) with the action attribute set to write and the output attribute to the variable holding your binary data:

```
<cffile action="write"
file="C:/newButton.gif"
output="myBinaryFile">
```

Notice that you do not do anything special in the action attribute to disclose the fact that the file is binary. ColdFusion is smart enough to figure this out by itself.

This question was written by [Jeremy Petersen](#).  
It was last updated on February 2, 2006 at 9:06:25 AM EST.

### CFML Referenced

[<cffile>](#)  
[toBinary\(\)](#)  
[toBase64\(\)](#)

## How do I remove HTML from a string?

There are many applications that allow visitors to enter content that will then get displayed on screen. This can potentially lead to issues if the user enters HTML. The HTML may break the layout of your site, or even serve as a way for someone to steal information from other users on your site. In general, you almost always want to remove HTML from user input. ColdFusion provides a few ways to do that.

The simplest method is to use either [htmlCodeFormat\(\)](#) or [htmlEditFormat\(\)](#). These two functions will find any HTML in a string and escape it. So if the user entered `<b>TEXT</b>`, the `<` and `>` characters will be escaped. Normally [htmlEditFormat\(\)](#) is used as [htmlCodeFormat\(\)](#) automatically inserts `<pre>` tags around the string.

While this method will escape the HTML, you may prefer to remove it all together. Luckily this is also rather easy using ColdFusion's regular expression support. The following UDF (user-defined function) from CFLib will do just that:

```
<cfscript>
function stripHTML(str) {
    return REReplaceNoCase(str,"<[^>*"'">","ALL");
}
</cfscript>
```

So to remove the HTML from a form value, you could do this:

```
<cfset cleanStr = stripHTML(form.input)>
```

This question was written by [Raymond Camden](#).  
It was last updated on May 7, 2006 at 1:35:12 PM EDT.

### CFML Referenced

[htmlCodeFormat\(\)](#)  
[<cfscript>](#)  
[htmlEditFormat\(\)](#)

## How do I reset the ColdFusion Administrator password?

If you forget the password to your ColdFusion MX Administrator, navigate to the cfusionmx7/lib folder and find the file named neo-security.xml. Create a copy of this file. Then open the original and find this node in the XML:

```
<var name="admin.security.enabled">  
<boolean value="true"/>  
</var>
```

Change true to false, restart your ColdFusion server, then immediately login and set up a new password.

This question was written by [Raymond Camden](#).

It was last updated on March 2, 2006 at 7:19:08 AM EST.

## How do I re-sort a query?

If you have a database query that is not sorted, or is sorted by the wrong column, you may have a need to re-sort the query by a new column. ColdFusion's query of query functionality makes this simple. The following code sample shows an example:

```
<cfquery name="original" datasource="foo">
select name, age, rank
from people
order by age asc
</cfquery>

<!-- Resort by name -->
<cfquery name="newQuery" dbtype="query">
select name, age, rank
from original
order by name asc
</cfquery>
```

When using query of queries, notice that we do not provide a datasource, but rather we tell ColdFusion that the dbtype is query. Also note how the from portion of the sql refers to the variable (original) that has the original query.

This question was written by [Raymond Camden](#).

It was last updated on March 23, 2006 at 6:49:54 AM EST.

### CFML Referenced

[<cfquery>](#)

## How do I sort a 2 dimensional array?

ColdFusion does not provide a built-in way to sort a multi-dimensional array. However, there 2 ways to accomplish this task.

1) Turn the multi-dimensional array into a query and then use query of queries to sort it.

```
<!-- sample array -->
<cfset arr = arrayNew(2)>
<cfset arr[1][1] = "beta">
<cfset arr[2][1] = "bar">
<cfset arr[3][1] = "foo">
<cfset arr[4][1] = "alpha">
<cfset arr[1][2] = "car">
<cfset arr[2][2] = "boat">
<cfset arr[3][2] = "bike">
<cfset arr[4][2] = "car">

<!-- convert array to CF query -->
<cfscript>
myQuery = QueryNew("one,two");

for (i=1; i LTE ArrayLen(arr); i=i+1) {
    newRow = QueryAddRow(myQuery);
    QuerySetCell(myQuery, "one", #arr[i][1]# );
    QuerySetCell(myQuery, "two", #arr[i][2]# );
}
</cfscript>

<cfdump var= #myQuery#>

<!-- Sort -->
<cfquery name="qSort"
dbtype = "query">
SELECT *
FROM myQuery
ORDER BY one
</cfquery>

<cfdump var= #qSort#>
```

2) You can sort single dimension arrays use the built in ColdFusion function [arraySort\(\)](#). So with this in mind, you can pull the array dimension you want to sort by into its own single dimension array and then use [arraySort\(\)](#), on this single dimension array.

```
var sortArray = ArrayNew(1);
for (i=1; i LTE ArrayLen(arrayToSort); i=i+1) {
    ArrayAppend(sortArray, arrayToSort[i][sortColumn]);
}
```

It is then a simple matter of reordering the rest of your multi-dimensional array based on the new sort order of the single dimension array.

```
theList = ArrayToList(sortArray);
ArraySort(sortArray, type, order);
for (i=1; i LTE ArrayLen(sortArray); i=i+1) {
    thePosition = ListFind(theList, sortArray[i]);
    theList = ListDeleteAt(theList, thePosition);
    for (j=1; j LTE ArrayLen(arrayToSort[thePosition]); j=j+1) {
        arrayToReturn[counter][j] = arrayToSort[thePosition][j];
    }
    ArrayDeleteAt(arrayToSort, thePosition);
    counter = counter + 1;
}
```

The above code was taken from Robert West's [ArraySort2D\(\)](#) function that can be found at: <http://www.cflib.org/udf.cfm?ID=390>

This question was written by [Jeremy Petersen](#).

It was last updated on May 8, 2006 at 5:24:26 PM EDT.

### CFML Referenced

[<cfquery>](#)  
[<cfscript>](#)  
[arraySort\(\)](#)

## How do I sort a single dimension array?

Use the built in ColdFusion [arraySort\(\)](#) function.

```
<cfset arr = arrayNew(1)>
<cfset arr[1] = "beta">
<cfset arr[2] = "bar">
<cfset arr[3] = "foo">
<cfset arr[4] = "alpha">

<!-- sort array ascending alphabetically -->
<cfset isSuccess = ArraySort(arr, "textnocase", "asc")>

<cfdump var="#arr#">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on May 8, 2006 at 2:57:02 PM EDT.

### CFML Referenced

[arraySort\(\)](#)

## How do I sort a structure?

Use the built in ColdFusion [structSort\(\)](#) function.

```
<cfscript>
foo = structNew();
StructInsert(foo, "a", "this");
StructInsert(foo, "b", "is");
StructInsert(foo, "c", "a");
StructInsert(foo, "d", "test");
</cfscript>
```

[StructSort\(\)](#) returns an array of top-level key names (strings).

```
<cfoutput>#arrayToList(structSort(foo))#</cfoutput>
```

You can also sort parent structures based on their child structures. For example, the following example will sort all of the people in foo by their age:

```
<cfset foo = structNew()>
<cfset foo.raymond = structNew()>
<cfset foo.raymond.age = 9>
<cfset foo.raymond.lastname = "Camden">
<cfset foo.jeremy = structNew()>
<cfset foo.jeremy.age = 10>
<cfset foo.jeremy.lastname = "Petersen">
<cfset foo.joe = structNew()>
<cfset foo.joe.age = 12>
<cfset foo.joe.lastname = "Test">

<cfoutput>#arrayToList(structSort(foo, "numeric", "asc", "age"))#</cfoutput>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on May 15, 2006 at 5:29:35 PM EDT.

### CFML Referenced

[<cfoutput>](#)  
[<cfscript>](#)  
[structNew\(\)](#)  
[structSort\(\)](#)

## How do I trim the contents of a form?

This piece of code will trim the contents of a form. Since the form scope always exists in ColdFusion, you can either run it automatically, or only when a form post is submitted.

```
<cfloop collection="#form#" item="formfield">
  <cfset form[formfield] = trim(form[formfield])>
</cfloop>
```

Another suggestion would be to `htmlEditFormat` the data. This escapes any HTML tags the user may have entered into the form:

```
<cfloop collection="#form#" item="formfield">
  <cfset form[formfield] = trim(htmlEditFormat(form[formfield]))>
</cfloop>
```

This question was written by [Tjarko Rikkerink](#).  
It was last updated on January 13, 2006 at 7:00:55 AM EST.

### CFML Referenced

[<cfloop>](#)

## How do I upload a file to my ColdFusion application by way of a form?

Using a HTML form to upload files is a 2-step process. The first step involves the use of a HTML form to collect the file data from the user. Once you have the file on your web server, the second step involves the use of [<cffile>](#).

If you know that your HTML form will contain a type = "file" form-field, you need to be sure to set the <form> enctype parameter to "multipart/form-data". This vital step ensures that non-text file data can be transmitted with the form post. The second step in setting up a HTML form to accept a file is to include a type= "file" form-field. It is important to note that the file upload feature is browser specific and not supported by all browsers- especially older browsers. It is also important to note that different browsers and operating systems may render the file input form differently. If your application must support multiple browsers and operating systems, you will want to be sure to test your type = "file" form-field code with each browser and operating system. A sample file upload form would look as follows:

```
<form action="fileUpload.cfm" method="post" enctype="multipart/form-data">
<input name="fileField" type="file">
<input type="submit" value="upload">
</form>
```

As already mentioned, the next step in the process is to use some ColdFusion tags and functions to capture the now posted form data. This is accomplished by using the [<cffile>](#) tag with the action = "upload" attribute, and the field attribute set to the type = "file" form-field you just posted. This code would look as follows:

```
<cffile
action="upload"
destination="C:/Temp/"
nameconflict="overwrite"
filefield="fileField">
```

This question was written by [Jeremy Petersen](#).  
It was last updated on January 26, 2006 at 10:16:40 AM EST.

### CFML Referenced

[<cffile>](#)

## How do I work with logarithms?

The [log\(\)](#) function returns the natural logarithm of its single numeric parameter:

```
<cfset testVar = log(100)>
<cfoutput>#testVar#</cfoutput>
4.60517018599
```

The [exp\(\)](#) function is the inverse of the [log\(\)](#) function:

```
<cfset testVar = exp(4.60517018599)>
<cfoutput>#testVar#</cfoutput>
100
```

The [log10\(\)](#) function returns the logarithm of its single numeric parameter to base 10:

```
<cfset testVar = log10(100)>
<cfoutput>#testVar#</cfoutput>
2
```

This question was written by [Jeremy Petersen](#).  
It was last updated on January 23, 2006 at 11:15:11 AM EST.

### CFML Referenced

[exp\(\)](#)  
[<cfoutput>](#)  
[log10\(\)](#)  
[log\(\)](#)

## How do you copy a structure?

There are (at least) 3 ways to copy a structure. Assume an existing structure named myStruct.

1) Variable assignment: `<cfset myNewStruct = myStruct />`

This will create a shallow copy (or copy by reference) of myStruct. Any modifications made to myStruct will also affect myNewStruct.

2) [structCopy\(\)](#): `<cfset myNewStruct = structCopy(myStruct) />`

This built-in function will create a deep copy (or copy by value) of all top level keys and their values. This means any modifications to myStruct will NOT affect these values. However, any nested structs are shallow copies (by reference). This means that nested structures within myNewStruct will be affected by any change to nested structs within myStruct.

3) [duplicate\(\)](#): `<cfset myNewStruct = duplicate(myStruct) />`

This built-in function will create a deep copy (or copy by value) of the entire structure and any nested structures. There is no reference whatsoever to the original struct.

The bottom line: If you need a truly separate entity (a clone of the original struct), use [duplicate\(\)](#).

This question was written by [Charlie Griefer](#).

It was last updated on January 17, 2006 at 6:19:16 PM EST.

### CFML Referenced

[structCopy\(\)](#)

[duplicate\(\)](#)

## How do you determine if an array position exists?

ColdFusion does not have any built in function to determine if an array position is defined. The simplest way is to use ColdFusion's built-in exception handling. The following code will check for the second position in an array:

```
<cfset arr = arrayNew(1)>
<cfset arr[1] = "Jacob">
<cfset arr[3] = "Lynn">
<cfset arr[5] = "Noah">

<cftry>
  <cfset foo = arr[2]>
  <cfoutput>Something exists at position 2.</cfoutput>
<cfcatch>
  <cfoutput>Something does NOT exist at position 2.</cfoutput>
</cfcatch>
</cftry>
```

Another option is to use the [arrayToList\(\)](#) function. In an array with empty positions, this will return a list with empty values. However, ColdFusion's list functions will not correctly tell you if a position is empty.

For a simpler version of the code above, the user-defined function, [arrayDefinedAt\(\)](#), may be used.

This question was written by [Raymond Camden](#).

It was last updated on January 12, 2006 at 7:30:41 AM EST.

### CFML Referenced

[<cfoutput>](#)

[<cftry>](#)

[<cfcatch>](#)

[arrayToList\(\)](#)

## How do you determine the amount of free space on a volume?

Currently there is no native ColdFusion functionality to accomplish this task. If ColdFusion can't do it natively, the next best option for this kind of task is usually calling a Java class directly from ColdFusion. Unfortunately, there does not seem to be any build-in way for current versions of Java to perform this task. The good news on the Java front is that the Java 6 (Mustang) File class will have `getUsableSpace()` and `getTotalSpace()` methods.

If you are in a Windows environment, you can use COM to accomplish this task. Rob Brooks-Bilson created a custom functional on [CFLib](#) called [FreeSpace](#) that uses Windows COM.

```
<cfscript>
/**
 * Returns the amount of free space (in bytes) available to the ColdFusion server for a specified drive or network share. (Windows only)
 *
 * @param drvPath Drive letter (c, c:, c:) or network share (\\computer\share).
 * @return Returns a simple value.
 * @author Rob Brooks-Bilson (rbils@amkor.com)
 * @version 1, July 19, 2001
 */
function FreeSpace(drvPath)
{
    Var fso = CreateObject("COM", "Scripting.FileSystemObject");
    Var drive = fso.GetDrive(drvPath);
    Return drive.FreeSpace;
}
</cfscript>
```

This question was written by [Jeremy Petersen](#).  
It was last updated on March 1, 2006 at 7:24:03 AM EST.

### CFML Referenced

[<cfscript>](#)

## How do you determine the beginning of a given week?

When building online calendars it's often useful to know the first day of the week for a given date. The following code returns the Sunday of the date provided. The code can easily be adjusted to return another day if required (see code comment).

```
<cfscript>
request.startDate = now();
currentDayOfWeek = dayOfWeek(REQUEST.startDate);
offset = 1 - currentDayOfWeek; //change number to get a different weekday
variables.startDate = createODBCDate(dateAdd("d", offset, request.startDate));
</cfscript>
```

This question was written by [Oliver Merk](#).

It was last updated on June 13, 2006 at 1:28:08 PM EDT.

### CFML Referenced

[now\(\)](#)  
[<cfscript>](#)

## How do you dynamically set a variable and its value?

You need to set a value to a variable but the name of the variable is also to be set dynamically, for instance the variable name and variable value maybe stored in the database. This can be done in ColdFusion multiple ways. The first, and probably preferred way, is to use structure notation:

```
<cfset varname = "name">  
<cfset value = "Jacob">  
<cfset variables[varname] = value>
```

This code simply treats the local variables scope as a structure. The next way lets ColdFusion evaluate the left hand side of an expression to determine the variable name:

```
<cfset varname = "name">  
<cfset value = "Jacob">  
<cfset "#varname#" = value>
```

Lastly, you can use the [setVariable\(\)](#) function to create a variable:

```
<cfset varname = "name">  
<cfset value = "Jacob">  
<cfset setVariable(varname, value)>
```

This question was written by [Martin Thorpe](#).  
It was last updated on March 2, 2006 at 6:10:34 AM EST.

### CFML Referenced

[setVariable\(\)](#)

## How do you force an application to use SSL?

If you want to force your application (or a portion of it) to use SSL, you can simply check one of the CGI variables, `server_port_secure`.

```
<cfif not cgi.server_port_secure>
  <cflocation url="https://#cgi.server_name#cgi.script_name#cgi.query_string" />
</cfif>
```

The code block above makes use of four CGI variables. The first one, `cgi.server_port_secure`, will be true if the current request is on a secure server. (Technically it seems to return 0 or 1, which can be treated as false and true in ColdFusion.) The variable `cgi.server_name` represents the current server. The variable `cgi.script_name` will represent the current document. (However this will not be the case when ColdFusion is using a context root of anything but `/`.) Lastly, the variable `cgi.query_string` will represent anything after the `?` character in the URL. If blank, nothing will be passed.

As a general warning, CGI variables can behave differently between different web servers, versions of web servers, web browsers, and many other factors. In general, care should be taken when using CGI variables.

This question was written by [Terrence Ryan](#).

It was last updated on January 27, 2006 at 12:42:28 PM EST.

### CFML Referenced

[<cfif>](#)

## How do you highlight searched words in results?

This is a simple matter of finding the text you are looking for, and replacing this text with a highlighted version of the same text. This can be accomplished using the [replaceNoCase\(\)](#) function.

```
<cfset myText = "This is my text!">
<cfset myWord = "my">
<cfoutput>#replaceNoCase(myText,myWord,"<span style='background:yellow'>#myWord#</span>","all")#</cfoutput>
```

Also, Verity searches support this functionality right out of the box using the context column. In your Verity result set, the context column returns a context summary containing the search terms, highlighted in bold. (This is enabled if you set the contextpassages attribute to a number greater than zero.). If bolding your search term is not enough, you can use the contextHighlightBegin and contextHighlightEnd attributes to append custom HTML before and after your search term.

This question was written by [Jeremy Petersen](#).  
It was last updated on June 30, 2006 at 1:47:58 PM EDT.

### CFML Referenced

[<cfoutput>](#)  
[replaceNoCase\(\)](#)

## How do you loop over the values in a structure?

A structure is a complex object that is comprised of key-value pairs.

Let's say we have a structure that represents/describes a person. Keys are 'Name', 'Address', and 'PhoneNumber'...with corresponding values 'Harvey', '123 Main Street', and '555-1212'.

CF's [<cfloop>](#) tag provides us with a "collection" loop that is can loop over a structure.

```
<cfloop collection="#myStructure#" item="key">
#key#: #myStructure[key]#<br />
</cfloop>
```

The "collection" attribute is the name of your structure (in # signs so CF knows to evaluate it).

The "item" attribute is simply a variable to represent the key of your struct for each iteration. For this reason you'll often see it represented as the variable "key", but it could just as easily have been "i", "x", or "foo".

The output of the loop above would, for each iteration, output the key alone ('Name', 'Address', 'PhoneNumber'), and then the value of that key in the structure ('Harvey', '123 Main Street', '555-1212').

If you prefer [<cfscript>](#), you can use a for-in loop to loop over a structure:

```
<cfscript>
for (key in myStruct) {
writeOutput(key & ": " & myStruct[key] & "<br />");
}
</cfscript>
```

This question was written by [Charlie Grierfer](#).  
It was last updated on January 10, 2006 at 8:14:47 AM EST.

### CFML Referenced

[<cfloop>](#)  
[<cfscript>](#)

## How do you loop over the values of an array?

An array is a collection of data indexed by numbers. So for an example, an array of values may have data at position 1, 2, and 3. In order to loop over the items in an array, the [arrayLen\(\)](#) function should be used:

```
<!-- Arr is an array of values. -->
<cfloop index="x" from="1" to="#arrayLen(arr)#">
<cfoutput>#arr[x]#</cfoutput>
</cfloop>
```

Generally this is safe code to write. However, it is possible that an array may have a missing position. Consider this array:

```
<cfset arr = arrayNew(1)>
<cfset arr[1] = "Jacob">
<cfset arr[3] = "Lynn">
<cfset arr[5] = "Noah">
```

Even though there is clearly only three items in the array, ColdFusion's [arrayLen\(\)](#) function will return five.

This question was written by [Raymond Camden](#).

It was last updated on January 12, 2006 at 7:22:50 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[<cfloop>](#)  
[arrayLen\(\)](#)

## How do you return the value of a dynamic structure key?

If you need to return the value of a structure key where the key is dynamic, you must use bracket notation. Assume myStruct is a structure and you want to get the value of the key stored in a variable, keyname.

```
<cfset keyname = "foo">
<cfset value = myStruct[keyname]>
```

If the value of mystruct.foo was "Jacob", the variable value will be "Jacob."

Bracket notation should also be used when key names have a space or other invalid character. If myStruct have a key called "Raymond Camden", you would not be able to do this:

```
<cfoutput>
#myStruct.raymond camden#
</cfoutput>
```

Instead, bracket notation must be used:

```
<cfoutput>
#myStruct["raymond camden"]#
</cfoutput>
```

This question was written by [Ben Forta](#).  
It was last updated on January 25, 2006 at 2:03:11 PM EST.

### CFML Referenced

[<cfoutput>](#)

## How do you stop users from clicking the submit button more than once?

If there is a form submission that may take a while, many users will click the submit button multiple times. One way to prevent that from happening is to add a bit of javascript to a form button that disables the button on the first submit. For example:

```
<INPUT TYPE="Button" VALUE="Submit" onClick="if(this.value == 'Submit') this.form.submit(); this.value = 'Please Wait';this.disabled=true;">
```

Of course this depends on Javascript being turned on in the user's browser. Another option is to use ColdFusion's built in form handling with `cfbutton` and `validate`. The `validate` tag has a `validate` option "SubmitOnce", that will prevent users from submitting the form more than once.

```
<cfinput type="submit" name="submit" value="Submit" validate="SubmitOnce">
```

This question was written by [Larry C. Lyons](#).  
It was last updated on January 11, 2006 at 11:45:24 AM EST.

## How to I initialize a CFC at the same time as I create it?

In many examples using CFCs, the call that creates the CFC also calls an `init()` method. This is done to both create an instance of a CFC as well as initialize it with certain information. So for example, you may want to pass in a datasource name to a CFC so that the component can perform database queries. How is this done?

This is a 2 step process. The first is to call the CFC using syntax like the example below:

```
<cfset cachedQuery = createObject("component", "CachedQuery").init(dsn)>
```

This says to create a variable called `cachedQuery` which will contain a reference to a CFC. The CFC is created using the [createObject\(\)](#) function. At the same time that the CFC is created, a method of the CFC called `init()` is called. Whatever is returned from the `init()` method is what will actually be loaded into the `CachedQuery` variable. This is called method chaining. It is essentially the act of calling multiple functions in one line. The result of the first operation (`createObject`) is passed to the next operation (`init`).

The CFC's `init` method could look something like this:

```
<cffunction name="init" access="public" returntype="CachedQuery" output="false">
  <cfargument name="dsn" type="string" required="yes">
  <!-- Set the DSN to the local variables for the CFC -->
  <cfset variables.dsn = arguments.dsn>
  <!-- Return the query object -->
  <cfreturn this>
</cffunction>
```

Very simply, it is expecting a string to be passed in and that string to be set to a local variable for the CFC called `DSN`. The important point here is two-fold. First, we are setting a `returntype` for this method (`cffunction`) to the name of the CFC. This is not 100% needed and you can ignore the return type totally if you want, but it's a good idea to have it here. The second thing is the most important. We will be returning `THIS` from the method. `THIS` is a reference to the entire CFC. The end result is that a reference to the CFC is passed back to the original call and then assigned to the outside variable resulting in a variable that holds a reference to the CFC.

This question was written by [Michael Dinowitz](#).

It was last updated on April 17, 2006 at 12:54:45 PM EDT.

### CFML Referenced

[<cffunction>](#)  
[createObject\(\)](#)

## Is there a way to determine the name of a user authenticated by the web server?

If the web server authenticates a user, it should provide that information to ColdFusion which will then make the login name available to you in variable `CGIAUTH_USER`.

This question was written by [Ben Forta](#).

It was last updated on January 9, 2006 at 7:07:21 AM EST.

## What's the most effective way to clean text pasted from Microsoft Word?

A common problem in forms arise when content is pasted from Microsoft Word. Characters sometimes become corrupted and do not store themselves well in the backend. One way to correct it is with a simple UDF (user-defined function) called [deMoronize](#). This UDF will clean up the broken content and replace it with safer characters.

```
<!-- udf.cfm contains deMoronize, downloaded from cflib.org -->  
<cfinclude template="udf.cfm">  
<cfset cleanText = deMoronize(form.text)>
```

This question was written by [Raymond Camden](#).  
It was last updated on March 21, 2006 at 2:26:06 PM EST.

## You have a date and time in Epoch seconds that you would like to convert to a date/time object.

Use [DateAdd\(\)](#) to add the Epoch seconds to the Epoch date.

ColdFusion does not natively deal with dates based on the Epoch. However, as a developer, you may be faced with situations where you are provided with a date/time value stored in Epoch seconds. If this is the case, you can easily convert the value to a ColdFusion date/time object using the [DateAdd\(\)](#) function.

```
<cfset e=1041397199>
<cfoutput>
#DateAdd("s",e,DateConvert("utc2Local", "January 1 1970 00:00"))#
</cfoutput>
```

In this example, the Epoch is first converted to local time using the [DateConvert\(\)](#) function. Next, the number of Epoch seconds we have are added to the converted Epoch time using [DateAdd\(\)](#). The output looks like this:

```
{ts '2002-12-31 23:59:59'}
```

Using the [DateConvert\(\)](#) function is only necessary if you want to convert Epoch seconds to local time. You can leave it out if you simply want Epoch seconds converted to a date/time object in UTC.

This question was written by [Rob Brooks-Bilson](#).

It was last updated on January 6, 2006 at 1:10:39 PM EST.

### CFML Referenced

[dateConvert\(\)](#)

[<cfoutput>](#)

[dateAdd\(\)](#)

## You have a date, time, or both and you need to convert it to Epoch seconds.

Use the [DateDiff\(\)](#) function to do the conversion. If you are converting from local time, you'll first need to convert your date/time to GMT(Greenwich Mean Time) using [DateConvert\(\)](#).

If you have ever worked on a \*nix system before, or with other development languages such as Perl, Java, or even JavaScript, you may already be familiar with the concept of Epoch time. In the \*nix world, the Epoch is defined as January 1, 1970 00:00 (midnight) GMT. This date/time is used as the starting point for all date and time calculations. By converting all dates following the Epoch to seconds, it makes it easy to do things like date comparisons, as well as adding to or subtracting from dates.

ColdFusion does not natively use the Epoch for date/time calculations. However, there may be various instances where you find you need to convert a date/time value to Epoch seconds for use in another environment.

To convert a ColdFusion date/time object to Epoch seconds, you can use the [DateDiff\(\)](#) function to calculate the difference in seconds between the Epoch and the date/time you want to convert. If the date you want to convert to Epoch seconds is not in GMT (also called UTC or Universal Coordinated Time), you'll need to convert the Epoch to local time, or convert your local time to GMT. This is done using [DateConvert\(\)](#).

```
<cfset thedate = createdatetime(2002,12,31,19,0,0)>
<cfoutput>
#TheDate# (local) <br>
Epoch seconds (convert Epoch to local time): #DateDiff("s",
DateConvert("utc2Local", "January 1 1970 00:00"), TheDate)# <br>
Epoch seconds (convert local time to UTC): #DateDiff("s", "January 1 1970
00:00", DateConvert("Local2utc", TheDate))#
</cfoutput>
```

Running this code produces this output:

```
{ts '2002-12-31 19:00:00'} (local)
Epoch seconds (convert Epoch to local time): 1041379200
Epoch seconds (convert local time to UTC): 1041379200
```

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 6, 2006 at 1:09:31 PM EST.

### CFML Referenced

[dateConvert\(\)](#)  
[<cfoutput>](#)  
[dateDiff\(\)](#)

## You have a string, such as user input, that you need to convert to a date/time object.

The [ParseDateTime\(\)](#) function, and its locale specific sibling [LSParseDateTime\(\)](#) can be used to convert an arbitrary date/time string into a date/time object. Additionally, [ParseDateTime\(\)](#) (but not [LSParseDateTime\(\)](#)) can be used to convert the supplied date/time from local time to GMT (Greenwich Mean Time).

As developers, we don't always get to choose the format that dates and times make their way into our applications. This is especially true for web applications that involve user input. If you have a form that allows a user to freely input a date/time, chances are not all of those dates are going to come in the same form (unless perhaps you use JavaScript to validate the input or design your form to accept the date time values as individual form fields for each date/time part). In any case, there will be situations where it's desirable to convert an arbitrarily formatted date/time value into a date/time object. When this need arises, you can accomplish this using the [ParseDateTime\(\)](#) function:

```
<cfoutput>
#ParseDateTime("12/31/2002")#<br>
#ParseDateTime("12/31/2002 23:59:59", "Standard")#
</cfoutput>
```

Results in:

```
{ts '2002-12-31 00:00:00'}
{ts '2002-12-31 23:59:59'}
```

The [ParseDateTime\(\)](#) function accepts an optional parameter that allows you to specify whether the date you supplied is in "Standard" (the default) or "POP" format. POP format is for use with date/time values associated with SMTP generated email messages. If POP is specified, the date/time string is automatically converted to GMT time using the English (US) locale. No conversion is performed if the conversion type is Standard:

```
<cfoutput>
#ParseDateTime("Tue, 31 Dec 2002 23:59:29 +0400 (EDT)", "POP")#<br>
#ParseDateTime("Tue, 31 Dec 2002 23:59:29 -0400", "POP")#
</cfoutput>
```

Results in:

```
{ts '2002-12-31 19:59:29'}
{ts '2003-01-01 03:59:29'}
```

[LSParseDateTime\(\)](#) can be used to parse locale specific date strings and works in the same way as [ParseDateTime\(\)](#) function except there is no optional parameter for handling POP dates.

This question was written by [Rob Brooks-Bilson](#).

It was last updated on January 6, 2006 at 12:54:13 PM EST.

### CFML Referenced

[lsParseDateTime\(\)](#)

[<cfoutput>](#)

[parseDateTime\(\)](#)

## You have two dates/times you want to compare.

You have two dates/times you want to compare.

Use one of ColdFusion's comparison operators in an expression, or use the [DateCompare\(\)](#) function.

There are two ways you can compare dates and times in ColdFusion. The first way is to use the tag with a comparison operator such as IS, EQ, NEQ, GT, GTE, LT, LTE, etc:

```
<cfset date1 = "12/31/1999">
<cfset date2 = "12/31/2002">

<cfif date1 gte date2>
<cfoutput>#Date1# is greater than or equal to #Date2#</cfoutput>
<cfelse>
<cfoutput>#Date1# is less than #Date2#</cfoutput>
</cfif>
```

This technique allows you to perform a basic comparison between the two dates/times.

You can perform a more flexible comparison between two dates using the [DateCompare\(\)](#) function. This function takes three parameters:

```
DateCompare(date1, date2, [,datepart])
```

The function returns -1 if date1 is less than date2, 0 if both date1 and date2 are equal, or 1 if date1 is greater than date2.

What makes this a more flexible method for comparing dates is that the precision of the comparison can be specified using the optional datepart parameter. Valid attributes for datepart are: s (second), n (minute), h (hour), d (day), m (month), and yyyy (year). This means you can compare two dates/times and make the comparison precise to the second, hour, month, year, etc:

```
<cfset date1 = "12/31/2002 19:00:00">
<cfset date2 = "12/31/2002 21:30:00">

<cfif datecompare(date1, date2, "d") eq 0>
<cfoutput>#Date1# and #Date2# fall on the same day.</cfoutput>
<cfelse>
<cfoutput>#Date1# and #Date2# don't fall on the same day.</cfoutput>
</cfif>
```

This example compares two date/time objects, with the precision set to "day". Even though the date/time objects are not exactly equal because of the different time stamps, the example still evaluates True because the comparison's precision is set to "day", making the time stamp insignificant.

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 6, 2006 at 1:04:52 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[<cfelse>](#)  
[<cfif>](#)  
[dateCompare\(\)](#)

## You need to create a date/time object.

You can write the date/time value as a string, or use the [CreateDate\(\)](#), [CreateTime\(\)](#), [CreateDateTime\(\)](#), or [CreateODBCDate\(\)](#) function depending on your specific needs.

Internally, ColdFusion stores dates and times as real numbers. The date is stored as the integer part of the number, and the time is stored as the fractional part. 0 represents 12:00 am on 12/30/1899. 7:00 pm on 12/31/2002 would be 37621.79167. The whole part of the number represents 37621 days since 12/30/1899 while .79167, or 7 pm is obtained by dividing the hour (19) by the total number of hours in a day (24). Storing dates in this manner allows ColdFusion to quickly and efficiently store and manipulate dates and times.

From a human readability standpoint, this format isn't very friendly. Because of this, ColdFusion allows you to refer to date/time objects as strings. You can specify dates and times separately, or combined. Dates must be in the range 100 AD to 9999 AD and can be written as:

```
12/31/02
12/31/2002
12-31-2002
2002-12-31
December 31, 2002
Dec 31, 2002
{ts '2002-12-31 00:00:00'}
```

ColdFusion handles two-digit years from 00 to 29 as twenty-first century dates and two digit years from 30 to 99 as twentieth-century dates.

Times are accurate to the second and can be written as:

```
7pm
7:00p
7:00pm
07:00pm
19:00:00
{ts '1899-12-30 19:00:00'}
```

Combined date/time objects can be written as any combination of the above. For example, a combined ODBC formatted date/time object looks like this:

```
{ts '2002-04-01 21:51:50'}
```

Date/Time values can be assigned to variables using and :

```
<cfset x="12/31/2002">
<cfparam name="x" default="12/31/2002">
```

They can also be coded directly in tag attributes and function parameters:

```
<cfcookie name="ID" value="12" expires="12/31/2002">
or
<cfset x = dateFormat("12/31/2002")>
```

While this may seem like an easy way to create dates, it can be problematic if you ever need to represent your dates in a locale other than English (US). The main problem has to do with the month and day portion of the date. In the U.S., dates are usually written mm-dd-yyyy. The month is first, followed by the day, then the year. In many other locales, the month and day parts of the date are reversed. Because of this, it is recommended you always use the [CreateDate\(\)](#), [CreateTime\(\)](#), or [CreateDateTime\(\)](#) functions to create date/time objects. These functions can be used in any locale and will ensure that the dates and times you create can be used with any locale.

To create a date object, use [CreateDate\(\)](#):

```
<cfset x = createDate(2002,12,31)>
<cfoutput>#x#</cfoutput>
```

The function requires three parameters, the year, the month, and the day. This removes any ambiguity regarding the position and value of the day and month portions of the date object.

Likewise, you can create a time object using [CreateTime\(\)](#):

```
<cfset x=CreateTime(19,0,0)>
<cfoutput>#x#</cfoutput>
```

[CreateTime\(\)](#), requires you to pass the hour (using the 24-hour clock), minute, and second.

To create a combined date/time object, use [CreateDateTime\(\)](#):

```
<cfset x=CreateDateTime(2002,12,31,7,0,0)>
<cfoutput>#x#</cfoutput>
```

This function requires you to specify the year, month, day, hour (24-hour clock), minute, and second for the date/time object you want to create.

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 6, 2006 at 11:32:05 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[createODBCDate\(\)](#)  
[createDate\(\)](#)  
[createTime\(\)](#)  
[createDateTime\(\)](#)

## You need to determine the difference between two dates or times.

Use [DateDiff\(\)](#) to return the interval between two dates and/or times. The [DateDiff\(\)](#) function is used to return the interval between two dates or times. The function takes three parameters, the interval for the comparison, and the two dates you want to compare.

The following code compares the difference between two dates in years, and the difference between two times in minutes:

```
<cfoutput>
#DateDiff("yyyy", "12/31/1999", "12/31/2002"),#<br>
#DateDiff("h", "12:00:00", "15:00:00")#
</cfoutput>
```

This interval can be expressed in a variety of formats:

Datepart	Interval
s	Second
n	Minute
h	Hour
ww	Week
w	Weekday
d	Day
y	Day of year
m	Month
q	Quarter
yyyy	Year

[DateDiff\(\)](#) returns the interval in the unit you specify by which the second date/time is greater than the first. If the first date is greater than the second, a negative number is returned.

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 6, 2006 at 12:59:46 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[dateDiff\(\)](#)

## You need to determine whether a number is positive or negative, or you need to find the absolute value of a number.

The [sgn\(\)](#) function takes a number as its single parameter. The [sgn\(\)](#) function will return 1 if the number is positive, 0 if the number equals 0, and -1 if the number is negative:

```
<cfset testVar = sgn(-127)>
<cfoutput>#testVar#</cfoutput>
-1
```

The [abs\(\)](#) function also takes a number as its single parameter. The [abs\(\)](#) function will return the absolute value (the value of the number without its sign) of the number:

```
<cfset testVar = abs(-127)>
<cfoutput>#testVar#</cfoutput>
127
```

This question was written by [Jeremy Petersen](#).  
It was last updated on January 9, 2006 at 9:55:07 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[abs\(\)](#)  
[sgn\(\)](#)

## You need to determine whether a string or numeric value is a valid date object.

Use [IsDate\(\)](#) or [LSIsDate\(\)](#) for strings, or [IsNumericDate\(\)](#) for numeric values. The [IsDate\(\)](#) function returns True if the specified string can be converted to a valid date/time object. Be sure to enclose literal dates in quotation marks.

```
<cfoutput>
<cfset x="12-31-2002">

#isDate(x)#<br>
#isDate("12/31/2002")#<br>
#isDate("Dec 31, 2002")#<br>
#isDate("13/31/2002")#<br>
#isDate("19:00")#<br>
#isDate("7pm")#
</cfoutput>
```

You should be aware that the [IsDate\(\)](#) function only works with dates formatted for the U.S. locale. If you have a date formatted using a different locale, you should use the [LSIsDate\(\)](#) function. It returns True if the specified date can be converted to a date/time object in the current locale or False if not.

If you need to determine whether a numeric value (real number) is a valid date/time object, you can use the [IsNumericDate\(\)](#) function. Like [IsDate\(\)](#), it returns True if the value you pass to it can be converted to a date/time object.

```
<cfoutput>
#isNumericDate(37621.79167)#<br>
#isNumericDate(-1)#
</cfoutput>
```

Because [isNumericDate\(\)](#) operates on numeric values, it can be used in any locale.

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 5, 2006 at 6:00:50 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[isNumericDate\(\)](#)  
[LSIsDate\(\)](#)  
[isDate\(\)](#)

## You need to extract the day/month/year/hour/minute/second (DMYHMS), day of the week/year, week number, or quarter from a date/time object.

[DatePart\(\)](#) accepts two parameters, the datepart you want to extract, and the date you want to extract the date part from:

```
<cfset thedate = createdatetime(2002, 12, 31, 23, 59, 59)>
<cfoutput>
#DateFormat(TheDate, 'dddd mmmm dd, yyyy')# #TimeFormat(TheDate, 'hh:mm:ss tt')#
<p>
<b>Second:</b> #DatePart('s', TheDate)#<br>
<b>Minute:</b> #DatePart('n', TheDate)#<br>
<b>Hour:</b> #DatePart('h', TheDate)#<br>
<b>Week:</b> #DatePart('ww', TheDate)#<br>
<b>Day:</b> #DatePart('d', TheDate)#<br>
<b>Day of week:</b> #DatePart('w', TheDate)#<br>
<b>Day of year:</b> #DatePart('y', TheDate)#<br>
<b>Month:</b> #DatePart('m', TheDate)#<br>
<b>Quarter:</b> #DatePart('q', TheDate)#<br>
<b>Year:</b> #DatePart('yyyy', TheDate)#
</cfoutput>
```

Running the code produces this output:

```
Tuesday December 31, 2002 11:59:59 PM
Second: 59
Minute: 59
Hour: 23
Week: 53
Day: 31
Day of week: 3
Day of year: 365
Month: 12
Quarter: 4
Year: 2002
```

ColdFusion also has a separate function that corresponds to each date part available in the [DatePart\(\)](#) function. These functions are often used in lieu of [DatePart\(\)](#) as they offer "shorthand" syntax for extracting a date part.

```
<cfset thedate = createdatetime(2002, 12, 31, 23, 59, 59)>
<cfoutput>
#DateFormat(TheDate, 'dddd mmmm dd, yyyy')# #TimeFormat(TheDate, 'hh:mm:ss tt')#
<p>
Second: #Second(TheDate)#<br>
Minute: #Minute(TheDate)#<br>
Hour: #Hour(TheDate)#<br>
Day: #Day(TheDate)#<br>
Day of week: #DayOfWeek(TheDate)#<br>
Day of year: #DayOfYear(TheDate)#<br>
Week: #Week(TheDate)#<br>
Month: #Month(TheDate)#<br>
Quarter: #Quarter(TheDate)#<br>
Year: #Year(TheDate)#<br>
</cfoutput>
```

Running this code produces the exact same output as the [DatePart\(\)](#) example.

You can get the string representation for the day of week, and month parts of a date object by using the [DayOfWeekAsString\(\)](#) and [MonthAsString\(\)](#) respectively.

```
<cfset thedate = createdatetime(2002, 12, 31, 19, 30, 55)>
<cfoutput>
#DayOfWeekAsString(DayOfWeek(TheDate))#<br>
#MonthAsString(Month(TheDate))#
</cfoutput>
```

This returns:

```
Tuesday
December
```

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 6, 2006 at 1:06:52 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[monthAsString\(\)](#)  
[dayOfWeekAsString\(\)](#)  
[datePart\(\)](#)

## You need to format a date/time object.

To format a date for the English (US) locale, use the [DateFormat\(\)](#) function. For times using the U.S. format, use [TimeFormat\(\)](#). To format a date using the conventions of a different locale, use the [LSDateFormat\(\)](#) function. Locale specific time formatting can be accomplished using [LSTimeFormat\(\)](#).

Few situation calls for formatting your dates and times like this:

```
{ts '2002-04-01 21:51:50'}
```

More often than not, you'll want or need to format a date/time object in a way different from how ColdFusion or your database stores the value internally. This is easily handled using the [DateFormat\(\)](#) function:

```
DateFormat(date [, mask])
```

[DateFormat\(\)](#) returns date formatted according to mask. If no value is specified for mask, [DateFormat\(\)](#) uses the default dd-mmm-yy. Valid entries for mask are:

d	Day of the month as a number with no leading zero for single-digit days.
dd	Day of the month as a number with a leading zero for single-digit days.
ddd	Three-letter abbreviation for day of the week
dddd	Full name of the day of the week.
m	Month as a number with no leading zero for single-digit months.
mm	Month as a number with a leading zero for single-digit months.
mmm	Three-letter abbreviation for the month.
mmm	Full name of the month.
y	Last two digits of year with no leading zero for years less than 10.
yy	Last two digits of year with a leading zero for years less than 10.
yyyy	Four digit year.
gg	Period/era
short	Java short date format
medium	Java medium date format
long	Java long date format
full	Java full date format

There is a wide variety of ways you can format your dates using [DateFormat\(\)](#). Here are some examples:

```
<cfset thedate = now()>
<cfoutput>
TheDate = #DateFormat(TheDate)#
<p>
m/d/yy: #DateFormat(TheDate, 'm/d/yy')# <br>
mm/dd/yy: #DateFormat(TheDate, 'mm/dd/yy')# <br>
mm/dd/yyyy: #DateFormat(TheDate, 'mm/dd/yyyy')# <br>
dd/mm/yyyy: #DateFormat(TheDate, 'dd/mm/yyyy')# <br>
dd mmm yy: #DateFormat(TheDate, 'dd mmm yy')# <br>
```

```
</cfoutput>
```

Note that [DateFormat\(\)](#) supports U.S. date formats only. To use a locale specific date format, use the [LSDateFormat\(\)](#) function. [LSDateFormat\(\)](#) returns a locale specific date format according to the mask you provide. If no mask is specified, [LSDateFormat\(\)](#) uses the locale specific default. This can vary depending on the locale your server is set to. Valid date masks are the same as for the [DateFormat\(\)](#) function.

Why ColdFusion doesn't have a combined function for formatting dates and times is beyond me. If you need to format the time portion of a date/time object, you'll need to use the [TimeFormat\(\)](#) function:

```
TimeFormat(time [, mask])
```

[TimeFormat\(\)](#) returns time formatted according to the mask you provide. If no value is specified for mask, [TimeFormat\(\)](#) uses the default hh:mm tt. Valid entries for mask are:

h	Hours based on a 12-hour clock with no leading zeros for single-digit hours
hh	Hours based on a 12-hour clock with leading zeros for single-digit hours
H	Hours based on a 24-hour clock with no leading zeros for single-digit hours
HH	Hours based on a 24-hour clock with leading zeros for single-digit hours
m	Minutes with no leading zero for single-digit minutes
mm	Minutes with a leading zero for single-digit minutes
s	Seconds with no leading zero for single-digit seconds
ss	Seconds with a leading zero for single-digit seconds
t	Single character meridian, either A or P
tt	Multi character meridian, either AM or PM
short	Java short time format
medium	Java medium time format
long	Java long time format
full	Java full time format

Examples:

```
<cfset thetime = now()>
<cfoutput>
TheTime = #TimeFormat(TheTime)#
<p>
TimeFormat(TheTime, 'h:m:s'): #TimeFormat(TheTime, 'h:m:s')#<br>
TimeFormat(TheTime, 'h:m:t'): #TimeFormat(TheTime, 'h:m:s t')#<br>
TimeFormat(TheTime, 'hh:mm:ss'): #TimeFormat(TheTime, 'hh:mm:ss')#<br>
TimeFormat(TheTime, 'hh:mm:ss tt'): #TimeFormat(TheTime, 'hh:mm:ss tt')#<br>
TimeFormat(TheTime, 'H:M:ss'): #TimeFormat(TheTime, 'H:M:s')#<br>
TimeFormat(TheTime, 'HH:MM:ss'): #TimeFormat(TheTime, 'HH:MM:ss')#<br>
</cfoutput>
```

Locale specific times are formatted using [LSTimeFormat\(\)](#). This works identically to the [TimeFormat\(\)](#) function, and uses the same masks. If no mask is provided, the function reverts to the locale specific default.

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 6, 2006 at 12:52:26 PM EST.

## CFML Referenced

[now\(\)](#)  
[<cfoutput>](#)  
[dateFormat\(\)](#)

[IsTimeFormat\(\)](#)  
[IsDateFormat\(\)](#)  
[timeFormat\(\)](#)

## You need to format a non-currency number for output.

The [decimalFormat\(\)](#) function is similar to the [dollarFormat\(\)](#) function. You pass in a number, and it will return a formatted string. However, the string will only be formatted with two decimal places and a thousandths separator.

```
<cfset testNum = -537>
<cfoutput>#decimalFormat(testNum)#</cfoutput>
-537.00
```

The [numberFormat\(\)](#) function gives you much more control of your formatted output. Along with passing it a number to format, you also pass in a formatting mask. This mask can include information such as digit placeholders, commas, padding with 0's, and many other options. One common use for this extra formatting is to better organize display of different sized numbers.

Take the following example without [numberFormat\(\)](#):

```
<cfset testNum = -537>
<cfset testNum2 = 5735>
<cfset testNumTotal = testNum + testNum2>
<cfoutput>
#testNum#<br>
#testNum2#<br>
-----<br>
#testNumTotal#
</cfoutput>
```

Running this code produces this output:

```
-537
5735
-----
5198
```

And with [numberFormat\(\)](#):

```
<cfset testNum = -537>
<cfset testNum2 = 5735>
<cfset testNumTotal = testNum + testNum2>
<cfoutput>
#numberFormat(testNum,"-$____.____")#<br>
#numberFormat(testNum2,"-$____.____")#<br>
-----<br>
#numberFormat(testNumTotal,"-$____.____")#
</cfoutput>
```

Running this code produces this output:

```
-$ 537.00
$5,735.00
-----
$5,198.00
```

As you can see, the output from the [numberFormat\(\)](#) block is much easier to read.

This question was written by [Jeremy Petersen](#).  
It was last updated on January 9, 2006 at 9:28:56 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[decimalFormat\(\)](#)  
[dollarFormat\(\)](#)  
[numberFormat\(\)](#)

## You need to format a number as a US currency for output.

The [dollarFormat\(\)](#) function takes a number as its single parameter, and returns a formatted String. The following formatting is added to the string: two decimal places, thousandths separator, and a dollar sign. Furthermore, if the number is negative, it will be wrapped in parentheses.

```
<cfset testNum = -537>  
<cfoutput>#dollarFormat(testNum)#</cfoutput>  
(537.00)
```

This question was written by [Jeremy Petersen](#).  
It was last updated on January 9, 2006 at 9:29:46 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[dollarFormat\(\)](#)

## You need to generate a unique identification value to track a user.

A very common task that deals with number generation is creating an UUID. UUID stands for Universally Unique Identifier. A UUID is a 35-character string representation of a unique integer. This unique integer is generated from the ethernet "MAC" address built into the computer, along with the current time (in 100ns increments). This assures that the identifier will be unique.

It is important to note that ColdFusion uses a slightly different format for its UUIDs. ColdFusion UUIDs can be broken into 4 parts such as: B54D60CD-DF98-4869-9C3910ABE33E5112, while other companies such as Microsoft use a 5 part format such as: 4CFB048C-A19E-44E8-83E8-B842A3D43AA3. If you are interested in working with 5 part UUIDs in ColdFusion, you can refer to [www.cflib.org](http://www.cflib.org) for a ColdFusion UDF (User Defined Function) for creating MS Style UUIDs.

You create an UUID in ColdFusion by using the [CreateUUID\(\)](#) function:

```
<cfset userUUID = createUUID()>
<cfoutput>#userUUID#</cfoutput>
B54D60CD-DF98-4869-9C3910ABE33E5112
```

This question was written by [Jeremy Petersen](#).

It was last updated on January 13, 2006 at 9:06:00 AM EST.

### CFML Referenced

[<cfoutput>](#)  
[createUUID\(\)](#)

## You need to get the current date/time from the server.

The [Now\(\)](#) function can be used to obtain the current date/time from the ColdFusion server:

```
<cfoutput>#Now()#</cfoutput>
```

The current date/time is returned as a date/time object, formatted as a timestamp:

```
{ts '2002-04-01 10:04:17'}
```

The object is formatted as {ts 'yyyy-mm-dd HH:mm:ss'} where ts indicates a time stamp. The date is formatted as a four-digit year followed by a two-digit month and a two-digit day. The time portion of the object is formatted for a 24-hour clock. Hours, minutes, and seconds are all formatted using two digits.

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 5, 2006 at 5:55:31 PM EST.

### CFML Referenced

[now\(\)](#)  
[<cfoutput>](#)

## You need to round a number to an integer value.

ColdFusion comes with a number of built in functions to work with rounding whole numbers (integers):

The [round\(\)](#) function rounds a number to the closest integer. For example:

```
test1: <cfoutput>#round(99.5)#</cfoutput>
<br>
test2: <cfoutput>#round(-99.5)#</cfoutput>
test1: 100
test2: -100
```

The [fix\(\)](#) function's output depends on if the number you pass in to it is positive or negative. If the number you pass in is greater then or equal to 0, the function returns the closets integer less then the number passed in. If the number you pass in is less then 0, the function returns the closest integer greater then the number in question:

```
test1: <cfoutput>#fix(99.5)#</cfoutput>
<br>
test2: <cfoutput>#fix(-99.5)#</cfoutput>
test1: 99
test2: -99
```

The [int\(\)](#) function returns the closest integer that is smaller than the number you pass in:

```
test1: <cfoutput>#int(99.5)#</cfoutput>
<br>
test2: <cfoutput>#int(-99.5)#</cfoutput>
test1: 99
test2: -100
```

The [ceiling\(\)](#) returns the closest integer greater than the number you pass in:

```
test1: <cfoutput>#ceiling(99.5)#</cfoutput>
<br>
test2: <cfoutput>#ceiling(-99.5)#</cfoutput>
test1: 100
test2: -99
```

This question was written by [Jeremy Petersen](#).

It was last updated on January 10, 2006 at 9:59:31 PM EST.

### CFML Referenced

[round\(\)](#)  
[ceiling\(\)](#)  
[<cfoutput>](#)  
[fix\(\)](#)  
[int\(\)](#)

## You need to test a string to see if it is a valid numeric value.

The [isNumeric\(\)](#) function is used to directly test a variable to see if it is numeric.

```
<cfset testVar = "foo">
<cfoutput>#isNumeric(testVar)#</cfoutput>
NO
```

A Boolean response (True or False) is returned.

The [val\(\)](#) function on the other hand, will go beyond an all or nothing check. It will attempt to parse out a number from the beginning of a string, and return this number if it exists:

```
<cfset testVar = "100foo">
<cfoutput>#val(testVar)#</cfoutput>
100
```

If the beginning of the string contains a number, that number is returned. If not, a 0 is returned.

This question was written by [Jeremy Petersen](#).

It was last updated on January 12, 2006 at 1:54:46 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[isNumeric\(\)](#)  
[val\(\)](#)

## You need to work with non-US formatted numbers and currencies.

LS functions are locale specific. You can use the [setLocale\(\)](#) function to change the current locale you will be working with. Once this locale is set, all locale specific functions will use this local. For more information on what locals are supported in ColdFusion, you can consult your ColdFusion documentation, or access the `server.coldfusion.supportedLocales` variable.

As you begin working with locale specific functions, you will probably notice that many of these functions have a non-locale specific twin. As a general rule, aside from some minor locale specific differences, the functions are identical. The following are the locale specific versions of some functions we have already covered in this chapter:

```

ISIsNumeric() is used to directly test a variable to see if it is numeric.
<cfset newLocal = setLocale("Dutch (Belgian)")>
<cfset testVar = "foo">
<cfoutput>#ISIsNumeric(testVar)#</cfoutput>
NO

```

The [ISCurrencyFormat\(\)](#) function is used to format a number with the locale specific currency format. It is very similar to the [dollarFormat\(\)](#) function with one exception. The exception being that [ISCurrencyFormat\(\)](#) has an optional type parameter. Type can be: none, local, or international. The Type parameter controls what type of formatting is used with the basic decimal number. For our example, we will ignore this parameter allowing the type parameter to default to the "local" value:

```

<cfset newLocal = setLocale("Dutch (Belgian)")>
<cfset testNum = -537>
<cfoutput>#ISCurrencyFormat(testNum)#</cfoutput>
-537,00 BF

```

The [ISEuroCurrencyFormat\(\)](#) function is similar to [ISCurrencyFormat\(\)](#), with the exception that if the current local specific country accepts the euro as a local currency, it can also display the euro currency symbol (€), or the international euro sign (EUR).

The [ISNumberFormat\(\)](#) function works the same as [numberFormat\(\)](#), only the mask contents adapt to the local specific settings:

```

<cfset newLocal = setLocale("Dutch (Belgian)")>
<cfset testNum = "-537">
<cfset testNum2 = "5735">
<cfset testNumTotal = testNum + testNum2>
<cfoutput>
#ISNumberFormat(testNum,"-$_.____")#<BR>
#ISNumberFormat(testNum2,"-$_.____")#<BR>
-----<BR>
#ISNumberFormat(testNumTotal,"-$_.____")#
</cfoutput>

```

Running this code produces this output:

```

BF- 537,00
BF 5.735,00
-----
BF 5.198,00

```

This question was written by [Jeremy Petersen](#).  
It was last updated on January 12, 2006 at 1:53:30 PM EST.

### CFML Referenced

[<cfoutput>](#)  
[ISCurrencyFormat\(\)](#)  
[ISNumberFormat\(\)](#)  
[ISIsNumeric\(\)](#)  
[dollarFormat\(\)](#)  
[numberFormat\(\)](#)  
[ISEuroCurrencyFormat\(\)](#)  
[setLocale\(\)](#)

## You want to add to or subtract from a date/time object.

The [DateAdd\(\)](#) function can be used to add or subtract various units of time from a date/time object. For example, you can use [DateAdd\(\)](#) to add or subtract an arbitrary number of seconds/minutes/days/months, etc. to a specific date/time. [DateAdd\(\)](#) takes the form:

```
DateAdd(datepart, number, date)
```

Valid entries for datepart are: s (second), n (minute), h (hour), ww (week), w (weekday), d (day), y (day of year), m (month), q (quarter), and yyyy (year). Number specifies the number of datepart units to add to date. To subtract from the specified date, make number negative. Here are some examples:

```
<cfset mydatetime=now()>
<cfoutput>The original time and date is
#TimeFormat(MyDateTime,'hh:mm:ss tt')#, #DateFormat(MyDateTime,'mmm dd, yyyy')#

<br><b>Add 30 Seconds:</b>
#TimeFormat(DateAdd('s', 30, MyDateTime),'hh:mm:ss tt')#
<br><b>Subtract 10 minutes:</b>
#TimeFormat(DateAdd('n', -10, MyDateTime),'hh:mm:ss tt')#
<br><b>Add 2 hours:</b>
#TimeFormat(DateAdd('h', 2, MyDateTime),'hh:mm:ss tt')#
<br><b>Add 9 weeks:</b>
#DateFormat(DateAdd('ww', 9, MyDateTime),'mmm dd, yyyy')#
<br><b>Add 3 weekdays:</b>
#DateFormat(DateAdd('w', 3, MyDateTime),'mmm dd, yyyy')#
<br><b>Subtract 67 days:</b>
#DateFormat(DateAdd('d', -67, MyDateTime),'mmm dd, yyyy')#
<br><b>Add 45 days of the year:</b>
#DateFormat(DateAdd('y', 45, MyDateTime),'mmm dd, yyyy')#
<br><b>Subtract 7 months:</b>
#DateFormat(DateAdd('m', -7, MyDateTime),'mmm dd, yyyy')#
<br><b>Subtract 2 quarters:</b>
#DateFormat(DateAdd('q', -2, MyDateTime),'mmm dd, yyyy')#
<br><b>Subtract 5 years:</b>
#DateFormat(DateAdd('yyyy', -5, MyDateTime),'mmm dd, yyyy')#
</cfoutput>
```

This question was written by [Rob Brooks-Bilson](#).  
It was last updated on January 6, 2006 at 1:02:03 PM EST.

### CFML Referenced

[now\(\)](#)  
[<cfoutput>](#)  
[dateAdd\(\)](#)