



A Little Bit About Me

I am the chief technical officer at InVision App, a prototyping and collaboration platform for designers, built by designers. I also rock out in JavaScript and ColdFusion 24x7. ([more](#))

Follow me: [Twitter](#) - [Facebook](#) - [LinkedIn](#)



The <http://t.co/JcViV510Ju> job board made its 148th Kiva donation on behalf of "Back-End Web Dev-IT" <http://t.co/etneGJ2XcY> #Jobs #FullTime

Latest via Twitter

[Home](#)

[About Me](#)

[Projects](#)

[Ask Ben](#)

[Contact](#)

[Job Board](#)



[RSS Feed](#)

| [Custom RSS Feed](#)

Robust CFScript Support For Tags In ColdFusion 11 Beta

Posted February 24, 2014 at 7:33 AM by [Ben Nadel](#)

Tags: [ColdFusion](#)

Over the past several releases of ColdFusion, the support for CFScript have been getting much better. In fact, I write most of my ColdFusion code in CFScript; but, the CFScript feature set does leaves something to be desired. With ColdFusion 11 Beta, however, it looks like Adobe has finally given us full (enough) CFScript support for CFML tags.

The syntax for the ColdFusion tag support in CFScript is rather straightforward. For tags that have bodies, the format is:

```
cfTagName( attribute-pairs* ) { ... }
```

... and for tags that do not have bodies, the format is:

```
cfTagName( attribute-pairs* );
```

These tags can be nested in the same way that you would nest CFML tags in your non-script code (see demo below).

For the ColdFusion tags that have bodies, the body of the tag may be associated with a new output buffer. Meaning, if you write output within the context of a CFScript-based tag, the content may or may not get written to the page buffer; the difference in outcome depends on the behavior of the given tag. CFSaveContent and CFThread are examples from prior releases of ColdFusion that have already demonstrated this behavior.

The following is a quick example that demonstrates a ColdFusion tag that consumes its output buffer (CFXml), a ColdFusion tag that consumes nested tags (CFHttp and CFHttpParam), and a ColdFusion tag that has no body (CFContent):

```
1 <!-- NOTE: ColdFusion 11 was in BETA at the time of this writing. -->
2 <cfscript>
3
4     // Define our friends to be converted into XML.
```

MY EXPERIENCE



AngularJS by Google

COLDFUSION

JOBS

Find Your Next
ColdFusion Job

One Man's
Search
For
Love
by
Ben Nadel



FORK ME



LIKE A BOSS

PROTOTYPING
MADE
BEAUTIFUL

invision

TRY IT FREE!



Back-End Web Developer-Information
Technologist

```

5  friends = [
6      {
7          name: "Sarah",
8          age: 47
9      },
10     {
11         name: "Joanna",
12         age: 35
13     },
14     {
15         name: "Kim",
16         age: 39
17     }
18 ];
19
20
21 // Create a short-hand for the buffer output to make the CFXML
22 // body a bit more readable.
23 add = writeOutput;
24
25 // When you use the ColdFusion tags in CFScript, the body of the
26 // becomes an output buffer to which you can write the output-body
27 // of the tag.
28 cfxml( variable = "doc" ) {
29
30     add( "<Friends>" );
31
32     for ( friend in friends ) {
33
34         add( "<Friend age='#friend.age# '>" );
35         add( friend.name );
36         add( "</Friend>" );
37
38     }
39
40     add( "</Friends>" );
41
42 } // END: cfxml.
43
44
45 // Log the XML document to the "remote" service using CFHTTP.
46 cfhttp(
47     result = "apiResponse",
48     method = "post",
49     url = "http://127.0.0.1:8500/cgi.script_name#.logger.cfm",
50     getAsBinary = "yes"
51 ) {
52
53     cfhttpParam(
54         type = "header",
55         name = "X-Sent-By",
56         value = "ColdFusion11"
57     );
58
59     cfhttpParam(
60         type = "body",

```

LOOKING FOR A GREAT JOB?

Michigan State University University
 Advancement Web Information Technologist
 The Back-End Web Developer-Information
 Technologist will play an important role in
 connecting backend data and technologies
 to front-end web interfaces for University
 Advancement, to deliver....

Recent Blog Comments



Mark James

Feb 27, 2014 at 3:28 PM

Encountered "(. Incorrect Select Statement, Expecting a 'FROM', But Encountered '(' Instead, A Select Statement Should Have a 'FROM' Construct.

I found this same issue when trying to use MID() in a cfquery of a query (PDF Solr Collection). I've been going nuts trying to find away around this problem all day. I need part of the date string in
 ... [read](#) »



Dave Gruska

Feb 27, 2014 at 2:45 PM

Lazy Loading Image With AngularJS

Fantastic code, Ben. Thanks - this really speeds up lazy loading significantly within an ng-repeat. One thing I noticed, though, was that it was taking a while before any images showed up above the
 ... [read](#) »



Radoslav Sandov

Feb 27, 2014 at 11:47 AM

Using Underscore.js Templates To Render HTML Partial

because I wanted to use assignmet of external templates like : `<script type="text/template" id="templateid" src="template.js"></script>` `<code& ... read »`



Boyd

Feb 27, 2014 at 11:00 AM

Preventing Cross-Site Request Forgery (CSRF / XSRF) With AngularJS And ColdFusion

Thanks man, I love how I never have to search for too long... ..in this case my search was about 10 seconds on google to find yet another useful post from you. lol... You guys rock! :) There is a jQ
 ... [read](#) »



sergio

Feb 27, 2014 at 10:49 AM

Using A Name Suffix In ColdFusion's CFMail Tag

```

61         value = doc
62     );
63
64 } // END: cfhttp.
65
66
67 // Reset the output buffer. CFContent support finally in CFScript!
68 cfcontent( type = "text/html;charset=utf-8" );
69
70 writeDump( var = doc, label = "Friends" );
71
72 </cfscript>

```

cfscript.cfm hosted with [by GitHub](#)

[view raw](#)

Pretty easy right! And, just about all ColdFusion tags can be used this way; though, I wasn't able to get CFLoop and CFTimer to work properly (not that those are particularly meaningful tags in a CFScript context).

The ColdFusion tag attributes can be provided as a comma-delimited list of key-value pairs; or, as a precomposed attributeCollection set:

```

1 <!-- NOTE: ColdFusion 11 was in BETA at the time of this writing. -->
2 <cfscript>
3
4 // Define all of the attributes first. This allows you to
5 // conditionally build up the attributes to be used.
6 attributes = {
7     name: "asyncCode",
8     action: "run",
9     thisVar: "Hello",
10    thatVar: "World"
11 };
12
13 // Invoke script-based tag with collection.
14 cfthread( attributeCollection = attributes ) {
15
16     thread.echoThis = thisVar;
17     thread.echoTag = thatVar;
18
19 }
20
21 // Join and output.
22 cfthread( name = "asyncCode", action = "join" );
23
24 writeDump( cfthread );
25
26 </cfscript>

```

attributes.cfm hosted with [by GitHub](#)

[view raw](#)

In addition to the generic ColdFusion tag syntax, ColdFusion 11 Beta also provides a convenience function - queryExecute() - specifically for streamlining SQL execution. In previous versions of ColdFusion, we could execute database queries within CFScript by using the Query.cfc component. This always felt a bit junky, which is why I (and many others), still use CFML tags when it comes to data gateway components.

The new CFML/CFScript syntax continues to fall short when it comes to the CFQuery tag. This is due to the commingling of the CFQuery's output buffer with the nested CFQueryParam tags. The queryExecute() function tries to bridge this gap by providing a native method that combines SQL, query params, and tag attributes in the same function signature.

Hello, I am new to coldfusion, I have a problem with sending mail within the code I want to put the values ??of a specific mail server, SMTP, USER, PASSWORD, PORT. ETC. Thank you. ... [read](#) »



sergio

Feb 27, 2014 at 10:49 AM

Using A Name Suffix In ColdFusion's CFMail Tag

Hello, I am new to coldfusion, I have a problem with sending mail within the code I want to put the values ??of a specific mail server, SMTP, USER, PASSWORD, PORT. ETC. Thank you. ... [read](#) »



sergio

Feb 27, 2014 at 10:46 AM

Using A Name Suffix In ColdFusion's CFMail Tag

Hello, I am new to coldfusion, I have a problem with sending mail within the code I want to put the values ??of a specific mail server, SMTP, USER, PASSWORD, PORT. ETC. Thank you. ... [read](#) »



Steve Withington

Feb 27, 2014 at 10:30 AM

Using The Timeout Attribute With CFHttp In ColdFusion To Limit Blocking

I don't think I've ever used that attribute before ... glad you posted this. Cheers! ... [read](#) »

ColdFusion Server Monitor

www.fusion-reactor.com/

proactive monitoring on CF 8,9 & 10 download FusionReactor today!



In the following code, I demonstrate the previously available approach, the new CFML/CFScript approach, and then the new queryExecute() approach:

```
1  <!-- NOTE: ColdFusion 11 was in BETA at the time of this writing. -->
2  <cfscript>
3
4      // Previous CFScript implementation of CFQuery.
5      getFriends = new Query(
6          sql = "
7              SELECT
8                  *
9              FROM
10                 friend
11              WHERE
12                 id > :id
13              ORDER BY
14                 id ASC
15          ",
16          datasource = "testing"
17      );
18
19      getFriends.addParam(
20          name = "id",
21          value = 1,
22          cfSqlType = "cf_sql_integer"
23      );
24
25      writeDump( var = getFriends.execute().getResult(), label = "Friends" );
26
27
28      // ----- //
29      // ----- //
30
31
32      // New CFScript implementation of CFQuery.
33      cfquery(
34          name = "friends",
35          datasource = "testing"
36      ) {
37
38          writeOutput( "
39              SELECT
40                  *
41              FROM
42                 friend
43              WHERE
44                 id >
45          " );
46
47          cfqueryparam( value = 1, cfSqlType = "cf_sql_integer" );
48
49          writeOutput( "
50              ORDER BY id ASC
51          " );
52
53      }
54
55      writeDump( var = friends, label = "Friends" );
56
```

```

57
58 // ----- //
59 // ----- //
60
61
62 // New CFScript implementation of CFQuery.
63 // --
64 // NOTE: When passing the SQL as a tag argument, I do not believe
65 // that it is possible to actually use query parameters, which is why
66 // I am hard-coding the "1" in the WHERE clause.
67 cfquery(
68     name = "friends",
69     datasource = "testing",
70     sql = "
71         SELECT
72             *
73         FROM
74             friend
75         WHERE
76             id > 1
77         ORDER BY
78             id ASC
79     "
80 );
81
82 writeDump( var = friends, label = "Friends" );
83
84
85 // ----- //
86 // ----- //
87
88
89 // Alternate new function implementation for CFQuery.
90 // --
91 // NOTE: The last two arguments are optional.
92 friends = queryExecute(
93     "
94         SELECT
95             *
96         FROM
97             friend
98         WHERE
99             id > :id
100         ORDER BY
101             id ASC
102     ",
103     {
104         id: {
105             value: 1,
106             cfSqlType: "cf_sql_integer"
107         }
108     },
109     {
110         datasource: "testing"
111     }
112 );
113
114 writeDump( var = friends, label = "Friends" );
115
116 </cfscript>

```

I have to say, as someone who has always questioned whether or not I could ever give up the CFQuery tag (the tag-based CFQuery tag), the queryExecute() function does seem rather nice. It gets rid of all the ceremony required by the Query.cfc component and still keeps the logic mostly readable. Could this spell the end of all CFML tags for my non-UI (User Interface) related code?

NOTE: The saving grace of the CFScript-based query is the fact that ColdFusion natively allows for multi-line strings. This allows a query statement to be formatted correctly without the ceremony of string concatenation or line-break escaping.

While the new CFScript tag syntax does make just about every construct in the ColdFusion language available, there are a few notable tags that we have all been waiting for!

```
1 <!-- NOTE: ColdFusion 11 was in BETA at the time of this writing. -->
2 <cfscript>
3
4 // Define page settings. CFSettings support finally in CFScript!
5 cfsetting(
6     requestTimeout = 2,
7     showDebugOutput = true
8 );
9
10
11 // ----- //
12 // ----- //
13
14
15 // Define response headers. CFHeader support finally in CFScript!
16 cfheader(
17     name = "X-Served-By",
18     value = "ColdFusion 11 Beta"
19 );
20
21
22 // ----- //
23 // ----- //
24
25
26 // Set new cookies. CFCookie support finally in CFScript!
27 cfcookie(
28     name = "coldFusionLives",
29     value = "You bet your @$$",
30     expires = "never"
31 );
32
33
34 // ----- //
35 // ----- //
36
37
38 // Execute binaries. CFExecute support finally in CFScript!
39 cfexecute(
40     variable = "standardOutput",
41     name = "echo",
42     arguments = "What what!",
43     timeout = 1
44 );
45
46
```

```

47 // ----- //
48 // ----- //
49
50
51 // Build XML documents (via buffer). CFXML support finally in
52 // CFScript. It's not quite as nice looking at E4X; however, it's
53 // a heck of a lot better than xmlParse().
54 cfxml( variable = "doc" ) {
55
56     writeOutput( "<Numbers>" );
57
58     for ( i = 1 ; i < 5 ; i++ ) {
59
60         writeOutput( "<Number>#i#</Number>" );
61
62     }
63
64     writeOutput( "</Numbers>" );
65
66 }
67
68
69 // ----- //
70 // ----- //
71
72
73 // Generating PDFs. CFDocument support finally in CFScript.
74 // ... um, yay, I guess.
75 cfdocument(
76     format = "pdf",
77     filename = expandPath( "./cf11.pdf" ),
78     overwrite = true
79 ) {
80
81     cfdocumentItem( type = "footer" ) {
82
83         writeOutput( "Page #cfdocument.currentPageNumber#" );
84
85     }
86
87     cfdocumentSection() {
88
89         writeOutput( "<h1>ColdFusion 11</h1>" );
90
91     }
92
93     cfdocumentSection() {
94
95         writeOutput( "<h1>Getting Started</h1>" );
96
97     }
98
99 }
100
101
102 // ----- //
103 // ----- //
104
105
106 // Process images. While there are a lot of CFImage methods that

```

```

107 // have been available, I think this is the first time that the
108 // "WriteToBrowser" action is available in CFScript!
109 cfimage(
110     source = "../goose-duck.jpg",
111     action = "writeToBrowser",
112     width = 100
113 );
114
115
116 // ----- //
117 // ----- //
118
119
120 // There are other ways to build a dynamic invocation signature
121 // (such as constructing an argumentCollection object). However,
122 // I just wanted to see if this would work with cfInvoke.
123 // --
124 // NOTE: The ColdFusion 11 Beta documentation states that these
125 // functions are not available; so, don't use them as they may
126 // be removed in the final release.
127 cfinvoke(
128     returnVariable = "result",
129     method = "sum"
130 ) {
131
132     // Dynamically build up the invocation arguments.
133     for ( i = 1 ; i <= 10 ; i++ ) {
134
135         cfinvokeArgument( name = i, value = i );
136
137     }
138
139 }
140
141
142 // I am the method we are invoking with a dynamic number of args.
143 public string function sum() {
144
145     // NOTE: Totally overkill (and inefficient) approach - just
146     // using reduce since it's new in ColdFusion 11.
147     return(
148         arguments.reduce(
149             function( result, item, index, collection ) {
150
151                 return( result + item );
152
153             },
154             0
155         )
156     );
157
158 }
159
160
161 // ----- //
162 // ----- //
163
164
165 // Invoke a ColdFusion custom tag from CFScript! I don't really
166 // use custom tags anymore; but, this is an interesting possibility.

```



```

167     cf_myTag( variable = "customTagOutput" ) {
168
169         writeOutput( "This is my tag content!" );
170
171     }
172
173 </cfscript>

```

new-tags.cfm hosted with [by GitHub](#)

[view raw](#)

I will probably never jump through the necessary hoops required to use CFDocument in CFScript. I only have it in this demo as a means to further illustrate using nested CFScript tags with output buffers. And, CFXml, while not the prettiest, is still better than what we had before.

While ColdFusion 11 has made [just about] every CFML tag available in CFScript, this update does leave us with a number of redundant constructs. We now have several tags that can be defined in multiple ways. The following is a non-exhaustive demonstration of both the existing and the new approaches to select tags:

```

1  <!-- NOTE: ColdFusion 11 was in BETA at the time of this writing. -->
2  <cfscript>
3
4      // Previous CFScript implementation of CFThread.
5      thread
6          action = "run",
7          name = "AsyncCode"
8      {
9
10         thread.ben = "Jamin";
11
12     }
13
14
15     // NOTE: This difference applies to other body-style tags like:
16     // - CFLock.
17     // - CFTransaction.
18     // - CFSaveContent.
19
20
21     // New CFScript implementation of CFThread.
22     cfthread(
23         action = "run",
24         name = "AsyncCode2"
25     ) {
26
27         thread.word = "to your mother!";
28
29     }
30
31
32     // Join thread using existing syntax.
33     thread name = "AsyncCode" action = "join";
34
35     // Join thread using new syntax.
36     cfthread( name = "AsyncCode2", action = "join" );
37
38
39     // ----- //
40     // ----- //
41
42

```

```

43 // Previous CFScript implementation of CFHttp.
44 apiRequest = new Http(
45     result = "apiResponse",
46     method = "post",
47     url = "http =//127.0.0.1:8500#cgi.script_name#.logger.cfm",
48     getAsBinary = "yes"
49 );
50
51 apiRequest.addParam(
52     type = "header",
53     name = "X-Sent-By",
54     value = "ColdFusion11"
55 );
56
57 apiResponse = apiRequest.send();
58
59
60 // New CFScript implementation of CFHttp.
61 cfhttp(
62     result = "apiResponse",
63     method = "post",
64     url = "http =//127.0.0.1:8500#cgi.script_name#.logger.cfm",
65     getAsBinary = "yes"
66 ) {
67
68     cfhttpParam(
69         type = "header",
70         name = "X-Sent-By",
71         value = "ColdFusion11"
72     );
73
74 } // END: cfhttp.
75
76
77 // ----- //
78 // ----- //
79
80
81 // Previous CFScript implementation of CFParam that is totally
82 // broken and you should only use this syntax if you don't like
83 // your fellow programmers. No, for real, don't use this syntax.
84 // It's not cool. And people will make fun of you.
85 param url.groove = "sauce";
86
87 // Previous CFScript implementation of CFParam.
88 param name = "url.ben" type = "string" default = "jamin";
89
90 // New CFScript implementation of CFParam.
91 cfparam( name = "url.foo", type = "string", default = "bar" );
92
93
94 // ----- //
95 // ----- //
96
97
98 // Previous CFScript implementation of CFInclude.
99 include "./code.cfm";
100
101 // New CFScript implementation of CFInclude.
102 cfinclude( template = "./code.cfm" );

```

```
103
104
105 // ----- //
106 // ----- //
107
108
109 // Previous CFScript implementation of CFThrow.
110 throw( type = "Fail", message = "Ooops." );
111
112 // New CFScript implementation of CFThrow.
113 cfthrow( type = "Fail", message = "Ooops." );
114
115
116 // ----- //
117 // ----- //
118
119
120 // Previous CFScript implementation of CFExit.
121 exit "exitTemplate";
122
123 // New CFScript implementation of CFExit.
124 cfexit( method = "exitTemplate" );
125
126
127 // ----- //
128 // ----- //
129
130
131 // Previous CFScript implementation of CFAbort.
132 abort "Something went wrong";
133
134 // New CFScript implementation of CFAbort.
135 cfabort( showError = "Something went wrong" );
136
137 </cfscript>
```

updated-tags.cfm hosted with [by GitHub](#) [view raw](#)

To be honest, I am not sure which syntax I like better. I am very familiar with using the existing syntax for tags like CFTransaction and CFLock in CFScript. Plus, the color-coding (in SublimeText 2) is much better for the existing syntax, at least for the moment. That said, I do love consistency; so, if I were to start using some of the new tags, which required the new syntax, I'd probably switch over all tags to the new syntax for the sake of consistent formatting.

 Like

 7 people like this. [Sign Up](#) to see what your friends like.

You Might Also Be Interested In:

- [The Elvis Operator vs. IsNull\(\) In ColdFusion 11 Beta](#)

Looking For a New Job? [View All Jobs](#) | [Post A Job](#) »

- **Back-End Web Developer-Information Technologist** at Michigan State University
- **ColdFusion Developer** at Nonfat Media
- **Mid-to-Senior Level Web Application Developer** at SiteVision, Inc.

- **Junior-to-Mid Level Web Application Developer** at Peavey Electronics
- **Full time on site Senior Coldfusion Developer needed** at Interactive Sites

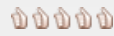
25% of job board revenue is donated to Kiva. *Loans that change lives* - [Find out more](#) »

Reader Comments



Ben Nadel

Feb 24, 2014 at 7:39 AM // reply »



11,694 Comments

80,148 Points

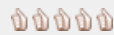
@All,

I should note that both struct pairs and attribute pairs can be defined using either "=" or ":". However, I have chosen to use "=" for attributes and ":" and for structs. Using the colon with the attribute pairs doesn't sit well with me - it sets my spider sense off.



Ben Nadel

Feb 24, 2014 at 8:12 AM // reply »



11,694 Comments

80,148 Points

@All,

Also, one thing I forgot to mention is that the CFML tags do not / cannot return values. So, don't try to do something like this:

```
response = cfhttp() { ... }
```

It won't compile. In order to get "return" values, you have to use the normal "result", "variable", and "returnVariable" attributes that you would have in normal CFML markup.



Cfsplender

Feb 24, 2014 at 8:34 AM // reply »

1 Comments

9 Points

Thanks a lot ben, Really awesome updates from adobe .. waiting for it!!



Jean

Feb 24, 2014 at 11:04 AM // reply »



14 Comments

40 Points

It is a shame Adobe did not follow Railo's lead for the basic syntax (no cf in front of the script version of a tag).



Kevin Boudloche

Feb 24, 2014 at 12:27 PM // reply »



18 Comments

197 Points

I like the cf leading the script version of the tag, it makes it easy to differentiate a tag-based cfscrip use vs a regular function. I also like the added parentheses to wrap the attributes. I hope railo follows suit so that I can use that syntax.



Aaron Martone

Feb 24, 2014 at 1:26 PM // reply »




60 Comments

635 Points

Ben, you have too much fun with your code. I can tell, merely by your examples, that you're having lots of fun dissecting through CF11.

Keep up the examples; this singular post has been THE BEST reference for cfscrip syntax that I could find online.



Gert Franz

Feb 25, 2014 at 2:43 AM // reply »

2 Comments

21 Points


@Kevin: we originally implemented the tag based notation in cfscrip as the CFML advisory committee proposed it 5 years ago. So in the latest 4.2 update, we extended the syntax to match ACF's version of it. So the following works in Railo:

```
cfloop(from="1", to="10", index="i") {}  
or  
cfloop(from:"1" to:"10" index:"i") {}  
or  
loop from="1" to="10" index="i" {}
```

If possible, we will implement the following syntax as well shortly:


```
response = cfhttp (...)
```

Gert



John Allen

Feb 26, 2014 at 2:17 PM // reply »

 13 Comments

124 Points

@Franz you all totally rule. Railo rulz

 Like 7 people like this. [Sign Up](#) to see what your friends like.

Post A Comment

Comment Etiquette: Please do not post spam. Please keep the comments on-topic. Please **do not post unrelated questions** or **large chunks of code**. And, above all, please be nice to each other - we're trying to have a good conversation here.

Author Name:

Author Email:

Author Website:

Comment:

Supported HTML tags for formatting:
bold italic <code>code</code>

Remember my information

Subscribe to comments

Send me a copy of this comment