

The Scoop on Scoping

Josh Adams

Adobe Systems Incorporated
Sr. SE, ColdFusion Specialist

joadams@adobe.com



Adobe

The Big Stuff...Up Front!

- **The Golden Rule of Scoping:** scope all references to all variables all the time (where ColdFusion allows it)
 - A very basic example: `<cfset Variables.myVariable = 12>` instead of `<cfset myVariable = 12>`

About Variables

- A variable is a name for a space in a computer's memory where data is stored
 - A variable has a name (a.k.a. an identifier) and it contains data (a.k.a. a value)
- In ColdFusion, variable values can be set and reset an unlimited number of times
 - Example:
`<cfset Variables.myVar = 1>`
`<cfset Variables.myVar = 2>`
- Variables can be passed as attributes to CFML tags, can be passed as parameters (a.k.a. arguments) of CFML functions, can be used in CFML expressions, etc.
- Note: ColdFusion has internal constants but does not have user-assignable constants
 - To emulate the behavior of a user-assignable constant, use a variable and assign a value to it only a single time

The logo consists of the letters 'C' and 'F' in a bold, white, sans-serif font, positioned on a blue square background with a vertical gradient from light blue at the top to dark blue at the bottom. The square is reflected on a dark surface below it.

CF

Demo

Variables

About Variable Scopes

- Variables have **scope**
 - Scope is the indication of where the variable is available and how long it persists
 - ColdFusion has a number of variable scopes (more in a moment)
 - ColdFusion allows variable names to be used one time in each scope
 - So you cannot have two variables of the same name in a single scope but you can have two variables of the same name when each variable is in a different scope
 - Example:
`<cfset Variables.myVar = 1>`
`<cfset Request.myVar = 100>`

Variable Scopes

- ColdFusion has the following scopes:

| | |
|--------------|----------------|
| Application | Arguments |
| Attributes | Caller |
| CGI | Client |
| Cookie | Flash |
| Form | function local |
| Request | Server |
| Session | This |
| ThisTag | Thread |
| thread local | URL |
| Variables | |

- For full details see [ColdFusion Developer's Guide: Scope Types](#)

Referencing Variables

- You can directly specify a variable's scope when referencing the variable
 - Use a period to separate the variable's scope name from the variable's name
 - Example: `Variables.myVariable`
 - Note that this is not the only way in which periods can be used in variable references; I recommend reading [ColdFusion Developer's Guide: Understanding Variables and Periods](#)
- If you do not directly specify a scope...
 - ...when using the `<cfset>` tag, the Variables scope is assumed (except if a variable of that name already exists in an applicable scope that cannot be referenced by name)
 - ...when not using the `<cfset>` tag, ColdFusion will search for the variable as follows:

| | |
|---|------------|
| 1. function local (UDFs & CFC methods only) | 7. CFFile |
| 2. thread local (inside threads only) | 8. URL |
| 3. Arguments | 9. Form |
| 4. Variables | 10. Cookie |
| 5. Thread | 11. Client |
| 6. CGI | |

The logo consists of the letters 'C' and 'F' in a bold, white, sans-serif font, positioned on a blue square background with a subtle gradient. The 'C' is on the left and the 'F' is on the right, both centered vertically.

Demo

Referencing Variables

The Golden Rule of Scoping

- **The Golden Rule of Scoping:** scope all references to all variables all the time (where ColdFusion allows it)
 - ColdFusion allows referencing all scopes by name except the function local and thread local scopes
- Because ColdFusion must search for variables when you do not specify the scope, you can improve performance by specifying the scope for all variables
- Also, scoping your variables makes your code clearer and easier to read
- Finally, scoping all variables can prevent unintended behavior of your code caused by a variable in one scope being used in place of a variable of the same name in another scope
 - An often overlooked area in which this is true is security: because of scope searching, attackers who add variables into the URL or Form scopes can inject data into your application in ways you may not have considered
- Remember, ColdFusion variables can be created in a multitude of ways; in all cases, you should scope your variables

Directly Referencing Variable Scopes: Best Practices Examples

```
<cfset Variables.myVar = "some value">
```

```
<cfset Variables.myObj = CreateObject("component",  
  "myCFC")>
```

```
<cfif Form.myField EQ "this value">  
  <cfoutput>#Form.myField#</cfoutput>  
</cfif>
```

```
<cfparam name="Variables.aVariable" default="hello">
```

```
<cfloop index="Variables.myLoopVar" ...></cfloop>
```

```
<cfquery name="Variables.myQ" datasoure="d"></cfquery>
```

```
<cfset Request.myStruct = StructNew(<)>
```

```
<cfset Request.myStruct.myKey = 12>
```

The logo consists of the letters 'C' and 'F' in a white, sans-serif font, positioned on a blue square background with a vertical gradient from light blue at the top to dark blue at the bottom. The square is reflected on a dark surface below it.

CF

Demo

The Golden Rule of Scoping & Violations of it

Scopes in CFCs

- The local scope for a CFC instance, as for a page, is the Variables scope
 - Data in the Variables scope of a CFC instance is only available to code in that CFC instance
- CFC instances also can make use of the This scope which, like the Variables scope, is associated with only a single CFC instance
 - Data in the This scope of a CFC instance is available to code outside the CFC by referencing the instance name as prefix
 - Example `<cfset>` in CFC: `<cfset This.aVar = "Josh">`
 - Example reference in code outside CFC: `<cfoutput>#Variables.myObj.aVar#</cfoutput>`

The logo consists of the letters 'C' and 'F' in a white, sans-serif font, positioned on a blue square background. The 'C' is on the left and the 'F' is on the right, both centered vertically. The blue square has a subtle gradient and is reflected on the dark background below it.

Demo

Scopes in CFCs

Function Local Scope

- Function local scope variables must be created using the var keyword
 - When using the `<cffunction>` tag to create a function, variables must be created at the very top of the function, immediately after any `<cfargument>` tags
 - Example:

```
<cffunction name="myFunction">  
  <cfargument name="anArgument" required="no">  
  <cfset var myFuncVar = 0>  
  <cfset var myLoop = 0>  
  
  <cfloop index="myLoop" from="1" to="10">  
  </cfloop>  
</cffunction>
```
- Commonly in UDFs and CFC methods developers fail to specify the function local scope for variables that should have been placed into that scope
 - This can lead to unintended behavior of your code

The logo consists of the letters 'C' and 'F' in a bold, white, sans-serif font, positioned on a blue square background with a vertical gradient from light blue at the top to dark blue at the bottom. The square is reflected on a dark surface below it.

CF

Demo

Function Local Scope

Something Crazy

- ColdFusion **does** allow you to use the name of a scope as the name of a variable
 - Example: `Variables.Variables`
 - I cannot think of any good reason to ever do this, but if you do it, you'll certainly want to (and may in fact actually have to in order to avoid errors) always scope your references of such variables

The letters 'C' and 'F' are displayed in a white, bold, sans-serif font. They are positioned on a blue square background that has a subtle gradient from light blue at the top to a darker blue at the bottom. The 'C' is on the left and the 'F' is on the right, both centered vertically within the square.

Demo

Something Crazy

A Note on Scoping a User-defined Function (UDF)

- UDF names are essentially ColdFusion variables and, like ColdFusion variables, UDFs exist in a scope
 - When you define a UDF, ColdFusion puts it in the Variables scope
 - You can assign a UDF to a scope just like you assign a variable to a scope: by assigning the function to a name in the desired scope
 - Example: `<cfset Request.myFunction = Variables.myFunction>`

The logo consists of the letters 'C' and 'F' in a bold, white, sans-serif font, centered within a blue square that has a subtle gradient from light to dark blue.

Demo

UDF Scoping

Some Best Practices

- **Scope Referencing**
 - Follow **The Golden Rule of Scoping**: scope all references to all variables all the time (where ColdFusion allows it)
- **Scope usage**
 - Inside of CFCs, use the Variables scope instead of the This scope
 - Use getters to retrieve and setters to set data inside CFCs; do not expose it via the This scope
 - Inside UDFs and CFC methods, use the function local scope except when you have an identified need for a variable whose scope is the entire CFC instance (in that case, use the Variables scope)
 - Use the Application scope for application-specific information
 - When doing this, do not set unchanging variables multiple times—instead test for the existence of such variables and set them only when they do not exist
 - Example: `<cfparam name="Application.myDSN" value="BigDSN">`

Resources

- [ColdFusion Developer's Guide](#)
- [ColdFusion Developer's Guide \(in PDF format\)](#)

Josh Adams

Adobe Systems Incorporated
Sr. SE, ColdFusion Specialist
(678) 701-7051

joadams@adobe.com

<http://blog.joshuaadams.com>

Adobe ColdFusion Sales Team

cfsales@adobe.com



Adobe