

# Introduction to ColdFusion Components

Nolan Erck

Bacfug, April 2008

# Overview

- Looking at procedural code in CF.
- Introduction to “Object Oriented” thinking.
- Vocabulary – overview of OO terms.
- Constructors.
- Scopes inside a CFC.
- “Public” vs. “Private”.
- Inheritance.
- Composition.
- Best Practices.

# Who Am I?

- Project Lead for Oasis Tooling.
- Independent Contractor ([southofshasta.com](http://southofshasta.com)).
- 12 years of software development.
- ColdFusion for 8 years (since version 4.5).
- 5 years of C/C++ in the video game industry.
- Clients/projects: [Riffage.com](http://Riffage.com), iConvention, SimPark, SimSafari, Grim Fandango, etc.

# Why CFCs? Why OO?

- Helpful in breaking up really large programs.
- Makes it easier to concentrate on “the task at hand”.
- Encourages code reuse and type safety.
- Lots of upcoming/current technology is OO-based (Mach-ii, Flex).

# Everything is an Object.

- CFCs can be used to “model” a wide variety of things:
  - A car.
  - An animal.
  - A “string utilities” function library.
  - Business rules in your application.
  - And on, and on...

# Overview of Terms.

- Class (CFC, Component)
  - The “type”. Same as “numeric”, “string”, “date”.
- Object
  - Instance of a CFC. The “actual” variable.
- Member
  - A variable inside a CFC.
- Method
  - A UDF inside a CFC.
- Example:

```
<cfset myCar = CreateObject( “component”, “racecar”) />  
<cfset myCar.color = “green” />  
<cfset myCar.drive() />
```

# Your First CFC

# Scopes Inside a CFC

- Variables – anything inside the CFC can see me.
  - This – Global. ANYONE can see (and change) it. Be careful!
  - Var – Local to the current function. Same as in “plain” UDFs.
- \*Be aware of “implied” variables (i.e. cfquery, cfsavecontent, etc).

# Constructors

- Code that runs automatically, as soon as a CFC is created.
- No “real” constructors in ColdFusion.
- Two common alternatives:
  - In-line constructors.
  - The `init()` method.
    - More common than in-line constructors.

# Public vs. Private Methods

- Public – everyone can see (and use) me.
- Private\* – only this CFC (and “extended” CFCs) can see me.

\* Same as “protected” in most other OO languages.

# Inheritance

- When a “parent” CFC provides common functionality for any “child” CFCs.
- The “is a” relationship.
- Use the “extends” keyword in ColdFusion.

# Composition

- The “has a” relationship.
- When one object needs to “dip into” another object.
- Common use: function libraries.

# Best Practices

- Use “init” style constructors.\*
- Always “var” scope your local variables!
- Use `output="false"` with `returnType="string"` instead of displaying output directly.
- Pass in other scopes as arguments:
  - `<cfset objFoo.doSomething( session ) />`
- Use the “hint” attribute – your IDE will thank you.

\* Except when building Mach-ii apps.

# Best Practices (cont.)

In the real world, sometimes you need to break the rules...

- Use your best judgement.
- Make it work first, make it elegant later.
- Your mileage may vary.

# Additional Resources

- [CFCDev mailing list](#).
- [Bacfug mailing list](#).
- [fullasagoog.com](http://fullasagoog.com).
- [coldfusionbloggers.org](http://coldfusionbloggers.org).
- [Macromedia Coding Guidelines](#).

# The End

Thanks!

[nolan.erck@gmail.com](mailto:nolan.erck@gmail.com) if you have any questions.

My blog: [southofshasta.com/blog](http://southofshasta.com/blog)

Shameless plug for my band: [nolanmusic.com](http://nolanmusic.com)