INTRODUCTION TO COLDFUSION

Objectives

• Understand how to set and output variable data using `cfset` and `cfoutput`
• Understand how to comment ColdFusion code
• Understand how to include an external document using `cfinclude`
• Understand relational and arithmetic operators in ColdFusion
• Understand how to write conditional code using `cfif`, `cfelse`, `cfelseif`, and `cfswitch`
ColdFusion Resources

Probably the best books for learning ColdFusion are the ones that come with the product.

Good news! You can download them for free:


*ColdFusion Markup Language (CFML) Reference*—definition of all tags and built-in functions.

*ColdFusion Markup Language (CFML) Quick Reference*—pamphlet-style book giving examples of tags and usage.

Dreamweaver also has built-in reference: Shift-F1 and select Adobe CFML Reference or Adobe CF Function Reference.

---

Things to know

We'll be moving to a new server:
http://coldfusioncs.etsu.edu/

ColdFusion pages must end in .cfm or .cfml extension.

ColdFusion Markup Language (CFML) is tag based:

- All tag names start with `cf`.
- Like HTML, *most* tags have both a start and end tag.

Since ColdFusion *produces* HTML output, but is not itself HTML, this has no impact on XHTML compliance.

There are approximately 100 CFML tags.

There are approximately 270 built-in functions.

CFML also has a "language within a language" CFScript, which is more JavaScript-like.
More Things to Know

ColdFusion is case-insensitive, however you should be consistent in your casing:

- Tag names in all lower case.
  (For example, `<cfabort>`)  
- Variable names in consistent case style.

Like PHP, server only processes elements inside ColdFusion tags. ColdFusion/XHTML mixed to create output.

- If it's not a part of a ColdFusion tag or contained between tags, it's plain XHTML.

Commenting ColdFusion Code

Comment tags just like HTML comments, but with 3 dashes instead of two

<!--- This is a ColdFusion comment. --->
Variables in ColdFusion

Variables types are not declared explicitly.

Naming rules:

- Begin with a letter and consist of letters, numbers, Unicode characters, and underscores only.

<cfset> used to create and initialize variable:

<cfset variable_name = expression>

String values placed in quotes (single or double).

Variables created with <cfset> are in the variables namespace, and can be more fully identified with (optional) variables prefix:

variables.variable_name

Output

<cfoutput>...</cfoutput> produces output.

Variable value substitution occurs by placing variable name inside #s.

<cfoutput>varX is #varX#</cfoutput>
<cfoutput>varX is #variables.varX#</cfoutput>

HTML can be part of output:

<cfoutput><li>varX is #varX#</li></cfoutput>

If it is not a CF tag or contained in #s, it is not changed by the CF processor.

This is also common:

<li>varX is <cfoutput>#varX#</cfoutput></li>
Example Program

```
<cfset firstName = "John">
<cfset lastName = "Adams">
<cfset fullName = firstName " " lastName>
<cfset varOne = 12>
<cfset varTwo = "97">
<cfset sum = variables.varOne + variables.varTwo>
<cfoutput>Hello #variables.fullName#</cfoutput>
<br/>
<cfoutput>#varOne# + #varTwo# = #sum#</cfoutput>
```

The above illustrates explicit and implicit variables namespace declarations. Pick a style and use it consistently in your coding.

http://coldfusioncs.etsu.edu/pittares/demo/demo1.cfm

Be careful to achieve correct output

```
<cfset school = "ETSU">
<p>Here's a cfoutput tag without the #'s:
<cfoutput>school</cfoutput></p>

<p>Here's a # without the cfoutput tag:
#school#</p>

<p>Here's both together:
<cfoutput>#school#</cfoutput></p>
```

http://coldfusioncs.etsu.edu/pittares/demo/demo1a.cfm
Working with special characters

Double quote, single quote, and pound sign have special meaning in CF. To include them in a string, double the character.

Escaping single and double quotes is context-sensitive. Don't need to escape single quotes inside double quotes and vice versa.

```
<cfset s1 = "We said, ""Happy Birthday!"">
<cfset s2 = 'She said, "Thank you"'>
<cfoutput> #s1#<br />#s2#</cfoutput>
```

http://coldfusioncs.etsu.edu/pittares/demo/demo1b.cfm

"Including" variable definitions

`cfinclude` allows server-side inclusion of external files.

It is common to place frequently used site variables in a single file consisting only of `cfset` tags.

The `cfinclude` tag has a single attribute--`template`--which specifies the file to be included.

```
<cfinclude template="CommonVariables.cfm">
Included file can be a ColdFusion file or contain other content.
Outputting function return values/expressions

Function return values can be written to output by placing function call inside #s.

Example: `Now()` returns current date and time.

```cfoutput>
Today's date: #Now()#
</cfoutput>```

[http://coldfusioncs.etsu.edu/pittares/demo/demo2.cfm](http://coldfusioncs.etsu.edu/pittares/demo/demo2.cfm)

Example: `DateFormat()` takes in a date code and a mask, and returns a formatted date string.

```cfoutput>
Today's date: #DateFormat(Now(), "mmmm d, yyyy")#
</cfoutput>```

[http://coldfusioncs.etsu.edu/pittares/demo/demo2a.cfm](http://coldfusioncs.etsu.edu/pittares/demo/demo2a.cfm)

When any expression is placed inside #s, the **result of the expression** printed.

Arithmetic and Boolean Operators in Expressions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+,-,*,/</code></td>
<td>as traditional</td>
</tr>
<tr>
<td>MOD</td>
<td>modulus</td>
</tr>
<tr>
<td>\</td>
<td>integer division</td>
</tr>
<tr>
<td>^</td>
<td>exponentiation</td>
</tr>
<tr>
<td>&amp;</td>
<td>string concatenation</td>
</tr>
<tr>
<td>NOT, AND, OR</td>
<td>as traditional</td>
</tr>
<tr>
<td>XOR</td>
<td>exclusive or (true if only 1 side of relation is true)</td>
</tr>
<tr>
<td>EQV</td>
<td>Equivalent (true if both sides of relation are true or both sides of relation are false)</td>
</tr>
<tr>
<td>IMP</td>
<td>Implication (x IMP y is false only if x is true and y is false. True in all other cases.)</td>
</tr>
</tbody>
</table>

[http://coldfusioncs.etsu.edu/pittares/demo/operators.cfm](http://coldfusioncs.etsu.edu/pittares/demo/operators.cfm)
Relational Operators in Expressions

Since characters '<' and '>' are used for tags, ColdFusion requires text-based operators:

<table>
<thead>
<tr>
<th>Non-supported operator</th>
<th>ColdFusion operator(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>IS, EQUAL, EQ</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT, LESS THAN</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LTE, LE, LESS THAN OR EQUAL TO</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT, GREATER THAN</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GRE, GREATER THAN OR EQUAL TO</td>
</tr>
<tr>
<td>&lt;&gt;, !=</td>
<td>IS NOT, NEQ, NOT EQUAL</td>
</tr>
<tr>
<td></td>
<td>CONTAINS</td>
</tr>
<tr>
<td></td>
<td>DOES NOT CONTAIN</td>
</tr>
</tbody>
</table>

http://coldfusioncs.etsu.edu/pittares/demo/operators2.cfm

Notable operator rules

String comparison is case insensitive.
Except for CONTAINS and DOES NOT CONTAIN, CF will attempt to do numeric comparison if conversion can be made. If it cannot, string comparison is used.
Conditional Execution

\[<\text{cfif expression}>, <\text{cfelse}>, \text{and } <\text{cfelseif expression}>\]

\[<\text{cfif expression1}>\]
- Executed if expression1 is true.
- Executed if expression1 is true.
\[</\text{cfif}>\]

\[<\text{cfif expression2}>\]
- Executed if expression2 is true.
- Executed if expression2 is true.
\[<\text{cfelse}>\]
- Executed if expression2 is false.
\[</\text{cfif}>\]

\[<\text{cfif expr3}>\]
- Executed if expr3 is true.
\[<\text{cfelseif expr4}>\]
- Executed if expr3 is false, expr4 is true.
\[<\text{cfelse}>\]
- Executed if expr3 and expr4 are false.
\[</\text{cfif}>\]

Notice that there are no \[</\text{cfelse}>, \text{or } <\text{cfelseif}>\] tags.