ColdFusion is a scripting language for web development. Read more here.

CFML

***C**old**F**usion **M**arkup **L**anguage*
ColdFusion started as a tag-based language. Almost all functionality is available using tags.

```
<em>HTML tags have been provided for output readability</em>

<!--- Comments start with "<!---" and end with "--->" --->
<!---
    Comments can
    also
    span
    multiple lines
--->

<!--- CFML tags have a similar format to HTML tags. --->
<h1>Simple Variables</h1>
<!--- Variable Declaration: Variables are loosely typed, similar to javascript --->
<p>Set <b>myVariable</b> to "myValue"</p>
<cfset myVariable = "myValue" />
<p>Set <b>myNumber</b> to 3.14</p>
<cfset myNumber = 3.14 />

<!--- Displaying simple data --->
<!--- Use <cfoutput> for simple values such as strings, numbers, and expressions --->
<p>Display <b>myVariable</b>: <cfoutput>#myVariable#</cfoutput></p><!--- myValue --->
<p>Display <b>myNumber</b>: <cfoutput>#myNumber#</cfoutput></p><!--- 3.14 --->

<hr />

<h1>Complex Variables</h1>
<!--- Declaring complex variables --->
<!--- Declaring an array of 1 dimension: literal or bracket notation --->
<p>Set <b>myArray1</b> to an array of 1 dimension using literal or bracket notation</p>
<cfset myArray1 = [] />
<!--- Declaring an array of 1 dimension: function notation --->
<p>Set <b>myArray2</b> to an array of 1 dimension using function notation</p>
<cfset myArray2 = ArrayNew(1) />

<!--- Outputting complex variables --->
<p>Contents of <b>myArray1</b></p>
<cfdump var="#myArray1#" /> <!--- An empty array object --->
<p>Contents of <b>myArray2</b></p>
<cfdump var="#myArray2#" /> <!--- An empty array object --->

<!--- Operators --->
<!--- Arithmetic --->
<h1>Operators</h1>
<h2>Arithmetic</h2>
<p>1 + 1 = <cfoutput>#1 + 1#</cfoutput></p>
<p>10 - 7 = <cfoutput>#10 - 7#<br /></cfoutput></p>
```

```
<p>15 * 10 = <cfoutput>#15 * 10#<br /></cfoutput></p>
<p>100 / 5 = <cfoutput>#100 / 5#<br /></cfoutput></p>
<p>120 % 5 = <cfoutput>#120 % 5#<br /></cfoutput></p>
<p>120 mod 5 = <cfoutput>#120 mod 5#<br /></cfoutput></p>

<hr />

<!--- Comparison --->
<h2>Comparison</h2>
<h3>Standard Notation</h3>
<p>Is 1 eq 1? <cfoutput>#1 eq 1#</cfoutput></p>
<p>Is 15 neq 1? <cfoutput>#15 neq 1#</cfoutput></p>
<p>Is 10 gt 8? <cfoutput>#10 gt 8#</cfoutput></p>
<p>Is 1 lt 2? <cfoutput>#1 lt 2#</cfoutput></p>
<p>Is 10 gte 5? <cfoutput>#10 gte 5#</cfoutput></p>
<p>Is 1 lte 5? <cfoutput>#1 lte 5#</cfoutput></p>

<h3>Alternative Notation</h3>
<p>Is 1 == 1? <cfoutput>#1 eq 1#</cfoutput></p>
<p>Is 15 != 1? <cfoutput>#15 neq 1#</cfoutput></p>
<p>Is 10 > 8? <cfoutput>#10 gt 8#</cfoutput></p>
<p>Is 1 < 2? <cfoutput>#1 lt 2#</cfoutput></p>
<p>Is 10 >= 5? <cfoutput>#10 gte 5#</cfoutput></p>
<p>Is 1 <= 5? <cfoutput>#1 lte 5#</cfoutput></p>

<hr />

<!--- Control Structures --->
<h1>Control Structures</h1>

<cfset myCondition = "Test" />

<p>Condition to test for: "<cfoutput>#myCondition#</cfoutput>"</p>

<cfif myCondition eq "Test">
    <cfoutput>#myCondition#. We're testing.</cfoutput>
<cfelseif myCondition eq "Production">
    <cfoutput>#myCondition#. Proceed Carefully!!!</cfoutput>
<cfelse>
    myCondition is unknown
</cfif>

<hr />

<!--- Loops --->
<h1>Loops</h1>
<h2>For Loop</h2>
<cfloop from="0" to="10" index="i">
    <p>Index equals <cfoutput>#i#</cfoutput></p>
</cfloop>

<h2>For Each Loop (Complex Variables)</h2>

<p>Set <b>myArray3</b> to [5, 15, 99, 45, 100]</p>
```

```
<cfset myArray3 = [5, 15, 99, 45, 100] />

<cfloop array="#myArray3#" index="i">
    <p>Index equals <cfoutput>#i#</cfoutput></p>
</cfloop>

<p>Set <b>myArray4</b> to ["Alpha", "Bravo", "Charlie", "Delta", "Echo"]</p>

<cfset myArray4 = ["Alpha", "Bravo", "Charlie", "Delta", "Echo"] />

<cfloop array="#myArray4#" index="s">
    <p>Index equals <cfoutput>#s#</cfoutput></p>
</cfloop>

<h2>Switch Statement</h2>

<p>Set <b>myArray5</b> to [5, 15, 99, 45, 100]</p>

<cfset myArray5 = [5, 15, 99, 45, 100] />

<cfloop array="#myArray5#" index="i">
    <cfswitch expression="#i#">
        <cfcase value="5,15,45" delimiters=",">
            <p><cfoutput>#i#</cfoutput> is a multiple of 5.</p>
        </cfcase>
        <cfcase value="99">
            <p><cfoutput>#i#</cfoutput> is ninety-nine.</p>
        </cfcase>
        <cfdefaultcase>
            <p><cfoutput>#i#</cfoutput> is not 5, 15, 45, or 99.</p>
        </cfdefaultcase>
    </cfswitch>
</cfloop>

<hr />

<h1>Converting types</h1>

<style>
    table.table th, table.table td {
        border: 1px solid #000000;
        padding: 2px;
    }

    table.table th {
        background-color: #CCCCCC;
    }
</style>

<table class="table" cellspacing="0">
    <thead>
        <tr>
            <th>Value</th>
```

```
        <th>As Boolean</th>
        <th>As number</th>
        <th>As date-time</th>
        <th>As string</th>
    </tr>
</thead>
<tbody>
    <tr>
        <th>"Yes"</th>
        <td>TRUE</td>
        <td>1</td>
        <td>Error</td>
        <td>"Yes"</td>
    </tr>
    <tr>
        <th>"No"</th>
        <td>FALSE</td>
        <td>0</td>
        <td>Error</td>
        <td>"No"</td>
    </tr>
    <tr>
        <th>TRUE</th>
        <td>TRUE</td>
        <td>1</td>
        <td>Error</td>
        <td>"Yes"</td>
    </tr>
    <tr>
        <th>FALSE</th>
        <td>FALSE</td>
        <td>0</td>
        <td>Error</td>
        <td>"No"</td>
    </tr>
    <tr>
        <th>Number</th>
        <td>True if Number is not 0; False otherwise.</td>
        <td>Number</td>
        <td>See &#34;Date-time values&#34; earlier in this chapter.</td>
        <td>String representation of the number (for example, &#34;8&#34;).</td>
    </tr>
    <tr>
        <th>String</th>
        <td>If "Yes", True <br>If "No", False <br>If it can be converted to 0, False <br>If it can
        <td>If it represents a number (for example, &#34;1,000&#34; or &#34;12.36E-12&#34;), it is
        <td>If it represents a date-time (see next column), it is converted to the numeric value of
        <td>String</td>
    </tr>
    <tr>
        <th>Date</th>
        <td>Error</td>
        <td>The numeric value of the date-time object.</td>
        <td>Date</td>
```

```
            <td>An ODBC timestamp.</td>
        </tr>
    </tbody>
</table>

<hr />

<h1>Components</h1>

<em>Code for reference (Functions must return something to support IE)</em>

<cfcomponent>
    <cfset this.hello = "Hello" />
    <cfset this.world = "world" />

    <cffunction name="sayHello">
        <cfreturn this.hello & ", " & this.world & "!" />
    </cffunction>

    <cffunction name="setHello">
        <cfargument name="newHello" type="string" required="true" />

        <cfset this.hello = arguments.newHello />

        <cfreturn true />
    </cffunction>

    <cffunction name="setWorld">
        <cfargument name="newWorld" type="string" required="true" />

        <cfset this.world = arguments.newWorld />

        <cfreturn true />
    </cffunction>

    <cffunction name="getHello">
        <cfreturn this.hello />
    </cffunction>

    <cffunction name="getWorld">
        <cfreturn this.world />
    </cffunction>
</cfcomponent>

<cfset this.hello = "Hello" />
<cfset this.world = "world" />

<cffunction name="sayHello">
    <cfreturn this.hello & ", " & this.world & "!" />
</cffunction>

<cffunction name="setHello">
    <cfargument name="newHello" type="string" required="true" />
```

```
    <cfset this.hello = arguments.newHello />

    <cfreturn true />
</cffunction>

<cffunction name="setWorld">
    <cfargument name="newWorld" type="string" required="true" />

    <cfset this.world = arguments.newWorld />

    <cfreturn true />
</cffunction>

<cffunction name="getHello">
    <cfreturn this.hello />
</cffunction>

<cffunction name="getWorld">
    <cfreturn this.world />
</cffunction>


<b>sayHello()</b>
<cfoutput><p>#sayHello()#</p></cfoutput>
<b>getHello()</b>
<cfoutput><p>#getHello()#</p></cfoutput>
<b>getWorld()</b>
<cfoutput><p>#getWorld()#</p></cfoutput>
<b>setHello("Hola")</b>
<cfoutput><p>#setHello("Hola")#</p></cfoutput>
<b>setWorld("mundo")</b>
<cfoutput><p>#setWorld("mundo")#</p></cfoutput>
<b>sayHello()</b>
<cfoutput><p>#sayHello()#</p></cfoutput>
<b>getHello()</b>
<cfoutput><p>#getHello()#</p></cfoutput>
<b>getWorld()</b>
<cfoutput><p>#getWorld()#</p></cfoutput>
```

CFScript

C*old****F****usion* ***S****cript*
In recent years, the ColdFusion language has added script syntax to mirror tag functionality. When using an up-to-date CF server, almost all functionality is available using scrypt syntax.

Further Reading

The links provided here below are just to get an understanding of the topic, feel free to Google and find specific examples.

1. Coldfusion Reference From Adobe
2. Open Source Documentation