JavaScript tool API

'eval' Method

eval(String script)

This method will evaluate a JavaScript code from a string and return the result.

```
$js.eval('script')
```

eval(String script, Object bindingObject)

This method will evaluate a JavaScript code with a single binding object. The code will be evaluated from a string. The "importer" name will be used as a binding name for this object.

```
#foreach ($class in $Class)
 $js.eval('importer.getName()', $class)
#end
```

eval(String script, String bindingName, Object bindingObject)

This method will evaluate a JavaScript code with a single binding object and specified binding name. The code will be evaluated from a string. The binding name will be used as the name for this object.

```
#foreach ($class in $Class)
 $js.eval('cls.getName()', 'cls', $class)
#end
```

eval(String script, Map bindingMap)

This method will evaluate a JavaScript code with a set of binding arguments (a name and an object). The code will be evaluated from a string. The binding map consists of key-value pairs for this binding name and binding object.

```
#set ($dict = $map.createHashMap())
#set ($void = $dict.put("first", "foo"))
#set ($void = $dict.put("last", "bar"))
$js.eval("first + ' ' + last", $dict)
```

'execute' Method

execute(String filename)

This method will execute a JavaScript file. The filename parameter is a file path to the JavaScript file.

```
$js.execute('filename.js')
```

execute(String filename, Object bindingObject)

This method will execute a JavaScript file with a single binding object. The filename parameter is a file path to the JavaScript file. The "importer" name will be used as the binding name for this object.

```
#foreach ($class in $Class)
 $js.execute('filename.js', $class)
#end
```

```
// filename.js
importer.getName();
```

execute(String filename, String bindingName, Object bindingObject)

This method will execute a JavaScript file with a single binding object and specified binding name. The filename parameter is a file path to the JavaScript file. The binding name will be used as the name for this object.

```
#foreach ($class in $Class)
 $js.execute('filename.js', 'cls', $class)
#end
```

```
// filename.js
cls.getName();
```

execute(String filename, Map bindingMap)

This method will execute a JavaScript file with a set of binding arguments (a name and an object). The filename parameter is a file path to the JavaScript file. The binding map consists of key-value pairs for this binding name and binding object.

```
#set ($map = $map.createHashMap())
#set ($void = $map.put("first", "foo"))
#set ($void = $map.put("last", "bar"))
$js.execute('filename.js', $map)
```

```
// filename.js
first + ' ' + last;
```

- **Absolute Path**: If a 'filename' is provided with an absolute path, JavaScript tool will read the JavaScript file from an absolute location such as $js.execute('*c:/mycode /readclass.js*').
- **Relative Path**: If a 'filename' is provided with a relative path, JavaScript tool will read the template from a relative location. This relative location starts from the current directory in which the template is located such as $js.execute('readclass.js').

'call' Method

call(String function)

This method will provide a short and reusable method to call a JavaScript function. This function must be defined before calling this method. This function can be defined by 'eval' or 'execute'.

```
$js.execute('javascript.js')
$js.call('calc(1, 3)')
```

```
// javascript.js
function calc(var1, var2)
{
 return var1 + var2;
}
```

call(String function, Object bindingObject)

This method will provide a short and reusable method to call a JavaScript function with a single binding object. The "importer" name will be used as the binding name for this object. This function must be defined before calling this method. This function can be defined by 'eval' or 'execute'.

```
$js.execute('javascript.js')
$js.call('calc(1, 3)', 10)
```

```
// javascript.js
function calc(var1, var2)
{
 var f = Number(importer ? importer : 0);
 return f + var1 + var2;
}
```

call(String function, String bindingName, Object bindingObject)

This method will provide a short and reusable method to call a JavaScript function with a single binding object and specified binding name. This binding name will be used as the name for this object. This function must be defined before calling this method. This function can be defined by 'eval' or 'execute'.

```
$js.execute('javascript.js')
$js.call('calc(1, 3)', 'factor', 10)
```

```
// javascript.js
function calc(var1, var2)
{
 var f = Number(factor ? factor : 0);
 return f + var1 + var2;
}
```

call(String function, Map bindingMap)

This method will provide a short and reusable method to call a JavaScript function with a set of binding arguments (a name and an object). The binding map consists of key-value pairs for this binding name and binding object.

```
#set ($map = $map.createHashMap())
#set ($void = $map.put("first", "foo"))
#set ($void = $map.put("last", "bar"))
$js.call('hello()', $map)
```

```
// filename.js
function hello()
{
 return 'hello ' + first + ' ' + last;
}
```