Beginning FrameMaker Scripting — Functions

This webinar will provide principles for developing successful automation scripts. We will show you how to use functions to simplify and organize your code.

Objectives

After completing this unit, you should be able to:

- Understand ExtendScript functions and how to use them.

Topics

In this unit, you will learn about the following topics:

- Function: A block of code identified by a name
- Functions are called from elsewhere in your code
- Functions can return a value
- Returning multiple values from a function
- Function variables
- Functions with a non-fixed number of parameters
- Rick Quatro bio

Function: A block of code identified by a name

```
function highlightCell (cell, doc) {

  // Function code goes inside curly braces.

}
```

From left-to-right:

- “function” keyword
- name for the function
- zero or more “parameters” or “arguments” inside of parenthesis
- open and close curly braces enclose the function’s code

Functions are called from elsewhere in your code

```
highlightCell (cell, doc);

function highlightCell (cell, doc) {

  // Function code goes inside curly braces.

}
```

The parameter names in the function call don't have to match the names in the calling code, but the order of the parameters has to match.

```
highlightCell (myCell, app.ActiveDoc);
```

Functions can return a value

```
alert (getArea (5, 5));

function getArea (height, width) {

  // Calculate the area.
  return (height * width);

}
```

The return statement immediately exits the function; any code after the return statement will not be executed.

Any function without the `return` statement returns `undefined`.

Returning multiple values from a function

```
var rectValues = getDimensions (5, 5);
alert (rectValues.area);
alert (rectValues.perimeter);

function getDimensions (height, width) {

  // Initialize an object for the area and perimeter.
  var dimensions = {area: 0, perimeter: 0};

  // Calculate the area and perimeter.
  dimensions.area = (height * width);
  dimensions.perimeter = (height * 2) + (width * 2);

  // Return the object.
  return (dimensions);

}
```

Function variables

Variables declared inside of a function with the var keyword are not visible outside of the function.

```
var rectValues = getDimensions (5, 5);
alert (dimensions); // Causes error; unknown variable

function getDimensions (height, width) {

  // Initialize an object for the area and perimeter.
  var dimensions = {area: 0, perimeter: 0};
  // Calculate the area and perimeter.
  dimensions.area = (height * width);
  dimensions.perimeter = (height * 2) + (width * 2);
  // Return the object.
  return (dimensions);

}
```

Function variables

Changes made to a “scalar” variable inside a function aren’t visible outside the function.

```
var counter = 1;
alert ("Before function call: " + counter); // 1
incrementCounter (counter);
alert ("After function call: " + counter); // 1

function incrmentCounter(counter) {

  counter += 1;
  alert ("Inside function: " + counter); // 2

}
```

Function variables

Changes made to an object variable inside a function **are** visible outside the function.

```
var counts = {counter:1}; // object
alert ("Before function call: " + counts.counter); // 1
incrementCounter (counts);
alert ("After function call: " + counts.counter); // 2

function incrmentCounter(counts) {

  counts.counter += 1;
  alert ("Inside function: " + counts.counter); // 2

}
```

Function variables

```
var counts = [1]; // array
alert ("Before function call: " + counts[0]); // 1
incrementCounter (counts);
alert ("After function call: " + counts[0]); // 2

function incrmentCounter(counts) {

  counts[0] += 1;
  alert ("Inside function: " + counts[0]); // 2

}
```

Functions with a non-fixed number of parameters

You can have a function process a varying number of parameters instead of a fixed number of parameters. You use the built-in `arguments` object of a function.

```
alert (getAverage (100, 92));
alert (getAverage (85, 91, 75, 82));
alert (getAverage (76, 84, 81, 79, 95, 88));

function getAverage () {

  var total = 0.0;
  for (var i = 0; i < arguments.length; i += 1) {
    total += arguments[i];
  }

  return total / arguments.length;
}
```

Rick Quatro bio

Rick Quatro is president of Carmen Publishing Inc, a western New York company providing FrameMaker solutions and support. He has been using FrameMaker since 1993 and FrameScript since its initial release in 1998. Rick's focus is on FrameMaker automation with FrameScript and ExtendScript. He has written thousands of scripts for clients worldwide, including Adobe, Eastman Kodak, Cisco, Harris, and Lockheed Martin. He is a well-known contributor to the FrameMaker online community.

Rick lives near Rochester, New York with his wife, Sherry. They are proud parents of eight sons, ages 25 to 10.

You can contact Rick by emailing rick@frameexpert.com or by calling 585-283-5045. Rick's website is www.frameexpert.com and his blog is at http://frameautomation.com.