Tips for moving to JavaScript

Common scripting examples

Referring to elements by Element Name or Alias

Take the alias name, FullName and full path name, Page1.Section1.FullName. The below examples demonstrate how these are referenced in LUA and how they can be referenced in JS

LUA

```
FullName.value = "John Smith";
Page1.Section1.FullName.value = "John Smith";
```

JS

Using the val() property

```
$("FullName").val("John Smith");
$("Page1.Section1.FullName").val("John Smith");
```

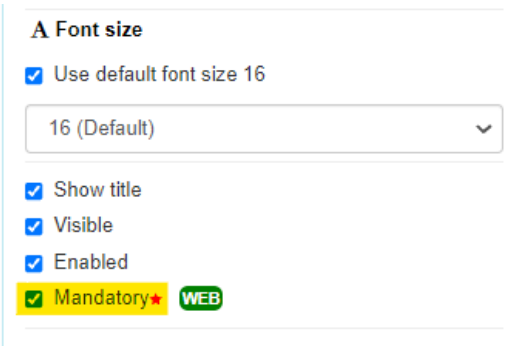
Using the value() property

```
$("FullName").value("John Smith");
$("Page1.Section1.FullName").value("John Smith");
```

Mandatory fields

This can be done via the Form Designer by selecting 'Mandatory' under the fields properties.

Note: this feature can be used on the app and web for both LUA and JS.



LUA

```
if this.value == "" then
   this.valid = false;
   this.message = "Please enter Full Name";
end
```

JS

To do this in scripting, you can write this as below:

```
if (!this.val()){
        this.valid(false, "Please enter Full Name.");
```

```
}
```

Conditional mandatories

LUA

```
if Category.value == "Other" and this.value == "" then
   this.valid = false;
   this.message = "Please describe category";
end
```

JS

```
if ($("Category").val() == "Other" && !this.val()){
          this.valid(false, "Please describe category.");
}
```

Creating a variable

LUA

```
local str = "";
local int = 0;
local bool = true;
```

JS

```
var str = "";
var int = 0;
var bool = true;
```

Concatenation and converting strings to numbers

LUA

```
local x = "Hello";
local y = " world!"
local z = x .. y; -- z now equals 'Hello world!'
```

Converting strings to numbers:

```
local a = tonumber(fieldA.value);
local b = tonumber(fieldA.value);
total = a + b;
```

JS

```
var x = "Hello";
var y = " world!"
var z = x + y; // z now equals 'Hello world!'
OR
var z = x.concat(y); // z now equals 'Hello world!'
```

Converting strings to numbers:

```
var a = $("fieldA").val();
var b = $("fieldA").val();
total = Number(a) + Number(b);
```

Looping

LUA

```
for i = 1,10 do
   if Page1["Item" .. i]["Quantity"].value == "" then
      counter = i;
      break;
   end
end
```

JS

```
for (i = 1; i <= 10; i++){
            if (!$("Page1.Item" + i + ".Quantity").val()){
                        counter = i;
                        break;
            }
}
```

Note: JS can loop through aliases, while LUA requires the fields full path name.

Looping through tables to hide rows

LUA

```
for i = 1,10 do
   Page1["Section1"]["Table1"]["row_" .. i].visible = Page1["Section1"]["Table1"]["cell_1_" .. i].value ~=
"";
end
```

JS

```
for (i = 1; i <= 10; i++) {

   //This line will work on Webforms only
   $("Page1.Section1.Table1").row(i).visible = $("Page1. Section1.Table1.cell_1_" + i + ".Date1").val();

   //The below can be used on the App only
  var dVar = $("Page2.Section2.testTable.cell_1_" + i + ".Date1");
  if (dVar.val()){
     $("Page1.Section1.Table1").showRow(i.toString();
  }else{
     $("Page1.Section1.Table1").hideRow(i.toString());
  }
}
```

If, else if, else

LUA

```
if Account.value == "James" then
   OutputValue.value = "James & Co";
elseif Account.value == "Arthur" then
   OutputValue.value = "Arthur & Sons";
else
   OutputValue.value = "Another Client";
end
```

JS

```
if ($("Account").val() === "James") {
   $("OutputValue").val("James & Co");
}else if ($("Account").val() === "Arthur") {
   $("OutputValue").val("Arthur & Sons");
}else{
   $("OutputValue").val("Another Client");
}
```

Databases

LUA

```
Type1.clear();

local query = "select DISTINCT Product from products order by Product";
local results, error = scriptExecSQL(query);

if results then
 local keyValue = "";

 for i = 1, table.getn(results) do
  local databaseRow = results[i]

  for index, value in pairs(databaseRow) do
   if index == "Product" then
    keyValue = value;
   end
  end
  Type1.add(keyValue);
 end
end
```

JS

```
if (!$("Type1").val()){
 $("Type1").removeOptions();
}

this.executeQuery("select DISTINCT [Product] from products order by [Product]",
 function(error){
  alert(error);
 },
 function(results){
  debugger;
  $.globals.results = results;
  var products = results.rows;

  for (var i = 0; i < products.length; i++){
   var keyValue = results.get.call(products[i], "Product");
    $("Type1").addOption(keyValue);
  }
 }
);
```

Reading Properties in JavaScript

Properties of an element can be read in JavaScript differently to how they are read in LUA.

Typically in LUA a property can be read or set by directly referencing it. Take the title property, this would be accessed by using 'elementName.title'.

In JavaScript, some properties can be referenced in a similar way, however the 'prop' function will need to be used in most cases. The prop function can be used to read or write the following properties:

- **Web only: color (read & write), textColor (read & write)**
- **App only: valid (will return true, until form has been validated), color (write only), textColor (write only), message (after form has been validated only), custom defined properties**
- **All platforms: value, title, enabled, visible**

To read a property's value `var x = $("field").prop("value");` **can be used instead of** `var x = $("field").val();`

Write to property: `$("field").prop("color", "green");`

```
$('Page1.PropertySet.Visible.Text1').prop("visible", true);

$('Page1.PropertySet.TextColor.Label1').prop("textColor", "red");

$('Page1.PropertySet.enabled.SingleSelection1').prop("enabled", false);
```

Best Practices

- **When using number on the app, remember to convert to a string using toString() or enclose the integer in "".**
- **When writing comments, there is a known issue when using single line comments (//), to avoid this please use multi line comments for the time being (/* enter text here */**

```
1   //Single line comment - Description of script
2   $("outputvalue").val("Test value");
3
4   /*
5   Multiple line comments. I don't want any of this
6   description included in my code.
7   */
```

- **There are some differences between scripting in JavaScript for the App and for Webforms, these are all pointed out in the JavaScript Guide.**
- **When creating variables in JavaScript, 'const' can also be used in place of 'var' and 'let' depending on the use case. Const declares immutable constants and let only runs within a function, being discarded afterwards.**