CRC function in FileMaker using JavaScript

Recently a client asked how to leverage JavaScript in FileMaker to do a specific calculation. For this blog post we replace the client's function with a CRC function to show you how to do it.

Up until a few weeks ago we would have pointed to loading JavaScript in a web viewer and using WebView.RunJavaScript to run the JavaScript. But now we would point to WebView.Evaluate and just run it. See the example script in our documentation for this CRC function.

With MBS FileMaker Plugin in version 10.0 we got our own JavaScript functions using the DukTape engine. We have an example in the documentation to run the CRC function right in a Let command and the JS.Evaluate command. But instead of initializing it each time in a Let statement, you may prefer to split this into three scripts, so you do the initialization only once.

The following script initializes the JavaScript environment once and uses a call to JS.Evaluate to pass the JavaScript functions:

```
If [ Length($$js) = 0 ]
        Set Variable [ $$js ; Value: MBS( "JS.New" ) ]
        Set Variable [ $r ; Value: MBS( "JS.Evaluate"; $$js;
"function makeCRCTable(){
    var c;
    var crcTable = [];
    for(var n =0; n < 256; n++){
        c = n;
        for(var k =0; k < 8; k++){
            c = ((c&1) ? (0xEDB88320 ^ (c >>> 1)) : (c >>> 1));
        }
        crcTable[n] = c;
    }
    return crcTable;
}

function crc32(str) {
    var crcTable = crcTable || (crcTable = makeCRCTable());
    var crc = 0 ^ (-1);

    for (var i = 0; i < str.length; i++ ) {
        crc = (crc >>> 8) ^ crcTable[(crc ^ str.charCodeAt(i)) & 0xFF];
    }

    return (crc ^ (-1)) >>> 0;
```

```
};") ]
End If
```

As you see we store the JavaScript environment in a global $$js variable, so we can refer to it everywhere in this file. Instead of letting JS.Evaluate define the functions, we could alternatively use JS.AddFunction if you prefer:

```
If [ Length($$js) = 0 ]
	Set Variable [ $$js ; Value: MBS( "JS.New" ) ]
	Set Variable [ $r ; Value: MBS( "JS.AddFunction"; $$js; "makeCRCTable";
"function makeCRCTable(){
	var c;
	var crcTable = [];
	for(var n =0; n < 256; n++){
		c = n;
		for(var k =0; k < 8; k++){
			c = ((c&1) ? (0xEDB88320 ^ (c >>> 1)) : (c >>> 1));
		}
		crcTable[n] = c;
	}
	return crcTable;
}") ]
	Set Variable [ $r ; Value: MBS( "JS.AddFunction"; $$js; "crc32"; "function
crc32(str) {
	var crcTable = crcTable || (crcTable = makeCRCTable());
	var crc = 0 ^ (-1);

	for (var i = 0; i < str.length; i++ ) {
		crc = (crc >>> 8) ^ crcTable[(crc ^ str.charCodeAt(i)) & 0xFF];
	}

	return (crc ^ (-1)) >>> 0;
}") ]
End If
```

Next we have a script to process data and use JS.CallFunction to run our function. This allows us to pass the argument as JSON data and avoid building JavaScript on the fly where wrong escaping could cause a JavaScript injection by an user:

```
If [ Length($$js) > 0 ]
	Set Variable [ $p ; Value: MBS( "JSON.CreateString"; test::Input) ]
	Set Field [ test::Output ; MBS( "JS.CallFunction"; $$js; "crc32"; $p ) ]
End If
```

When the solution closes you can cleanup the JavaScript environment:

```
If [ Length($$js) > 0 ]
	Set Variable [ $r ; Value: MBS( "JS.Free"; $$js ) ]
	Set Variable [ $$js ; Value: "" ]
End If
```

If you have questions, please do not hesitate to contact us.