Using Scripts in Derivative/Filter Processor

There are some differences in how JavaScript should be used in the JavaScript tab of Derivative and Filter Processor.

The intent of the Derivative JavaScript tab is to perform combined calculations. For example:

```
var temp = expensiveOperation();

var Result1 = temp*2;

var Result2 = fn(temp);
```

In this way, the *expensiveCalculation* has to be done only once.

You can then write the Derivatives in the table as follows:

```
NewField1 : Integer = Result1

NewField2 : Integer = Result2
```

It is not intended for defining functions, though in JavaScript you can do that anywhere. There is no return from this JavaScript – the values are accessed directly, hence the JavaScript scope must be available to the table so it can "see" Result1 and Result2. The internal name of this JavaScript is "PreProcessor" - which gives you an idea of its intent.

In the case of Filter, the JavaScript tab does expect a return value – a boolean determining whether to pass the record or not. As only one result is required (unlike Derivative), we evaluate the script and expect a return. We do not need to make the scope of the JavaScript persist back to the filter table. In this case the JavaScript is working as a direct action and not as a pre-processor.