Tips for our JavaScript functions

Here a few tips we recently learnt for our JavaScript engine based on Duktape in the MBS Plugins for FileMaker:

1. Use performance.now() to get time within JavaScript and measure how long something takes. Here an example where we measure how many milli seconds it takes to run a loop one million times:

   ```
   function testFunction() {
      for (var i = 0; i < 1e6; i++) {}
   }

   var t1 = performance.now();
   testFunction();
   var t2 = performance.now();

   Print('test took: ' + (t2 - t1) + ' milliseconds');
   ```
2. All global functions and properties in JavaScript are attached to the global object. You can access them explicitly via globalThis keyword.
3. List global properties via JavaScript: `Var list = Object.keys(globalThis);`
   The list will include all global names including functions like Print registered by MBS Plugin.
4. Check Duktape.version for the version number. If we update the engine, you can check the version number and compare. Currently the value is 20500 for version 2.5.
5. Use unicode escapes in strings if needed to mask special characters. While the engine handles UTF-8 fine, you can escape some characters with \u followed with a hex 4 digit number, e.g.

   ```
   n = '\u0041';
   ```

   Which will assign "A" to variable n.
6. Use built-in functions to encode text as hex or base64 via Duktape.enc: `Duktape.enc('base64', 'Hello')`
   This returns "SGVsbG8=".
7. Use Duktape.dec and TextDecoder class to decode base64/hex encoded text to UTF-8 string: `var result = new TextDecoder().decode(Duktape.dec('base64', 'Zm9v'));`
   Result is now a variable with text "foo".
8. Access call stack with Duktape.act function. Query Duktape.act(i) with various values of i. Pass -1 for the act() call, -2 for the current function and -3 for the caller of current function. The

returned object contains properties for lineNumber and function. When you query function.name, you get the name of the function.

9. Add console object if needed. You can add a console object and include a log function, which sends arguments to a Print function:
   ```
   console = { log: function()
   { Print(Array.prototype.join.call(arguments, ' ')); } };
   ```
10. Provide environment some JavaScript may need. If some JavaScript expects a global window object to store something, you can simply create a dummy one: `window = {};`
    This may not provide functionality, but allows at least code to check for this to work.
11. Prefer functions over evaluating code. Code in functions is parsed and uses registers for the virtual machine, while code evaluated may use named variables, so if you can put your code in a function and just evaluate the function call, things may execute a bit faster.
12. Be aware all numbers in JavaScript are doubles. Evaluate "9999999999999999" and you get "10000000000000000" back. Comparing numbers may not give the result you expect as "0.1+0.2==0.3" evaluates to false.

If you have questions, please do not hesitate to contact us. See also JS.Evaluate and related JavaScript functions in MBS FileMaker Plugin.