

## Introduction

The **Panel Builder** scripting engine now supports JavaScript syntax for writing functional code and executing commands.

A sample script using JavaScript syntax could look like the following:

```
myVariable = '127.0.0.1';
myFunction(myVariable);
```

All standard JavaScript syntax can be used and all defined variables are assigned to the global scope as long as you do not prefix them with the var declaration (i.e., `var k = 0;`). The var declaration in JavaScript is used to attach a variable to a local scope and that variable will no longer be available after that function or script ends/returns. A number of helpful functions are also included in the global JavaScript scope (see the following table):

Option	Description
<b>Global Objects</b>	
<b>project</b>	Object that represents the current <b>Panel Builder</b> project and its panels.
<b>panel</b>	Object that represents the currently visible panel.
<b>self</b>	Object representing the button or widget that is calling the script.
<b>buttons</b>	Array holding all of the buttons on the currently visible panel.
<b>widgets</b>	Array holding all of the widgets on the currently visible panel.
<b>Server</b>	An object containing information about the server like <code>Server.name</code> and <code>Server.ip</code> . Optionally server warnings, errors, and notices in the console can be turned off by setting <code>Server.errors = false</code> , etc.
<b>Client</b>	An object containing information about the client.
<b>\$/jQuery</b>	A jQuery object.
<b>Global Functions</b>	
<b>noop</b>	An empty function that does nothing. Can be used in the place of a callback.
<b>sleep</b>	Pauses execution for specified amount of milliseconds.
<b>setTimeout</b>	Allows execution of function after specified amount of milliseconds.
<b>setInterval</b>	Allows execution of function in an interval of specified amount of milliseconds.
<b>asciiToHex</b>	Function used to convert an ASCII string to a HEX string.
<b>hexToAscii</b>	Function used to convert a HEX string to an ASCII string.
<b>get</b>	Pass this function the ID of a button, widget, or panel and it returns the object.
<b>require</b>	Function that is passed the string name of the filepath of a Module.
<b>gotopanel</b>	Function that moves to a new panel.
<b>tcpclient</b>	Function to send TCP message to a client.
<b>udpclient</b>	Function to send UDP message to a client.
<b>telnetclient</b>	Function to send a telnet message to a client.
<b>int</b>	Function to convert a string or number to an integer.
<b>float</b>	Function to convert a string or number to a float value.
<b>string</b>	Function to convert an object or number to a string.

Option	Description
<b>time</b>	Function to return UNIX time in seconds.
<b>millitime</b>	Function to return UNIX time in milliseconds.
<b>openLink</b>	Function to open a browser tab or window to a network or internet address.
<b>setServerVar</b>	Function to save a variable to the server.
<b>getServerVar</b>	Function to retrieve a variable from the server.
<b>setProjectVar</b>	Function to save a variable to the project's directory on the server.
<b>getProjectVar</b>	Function to retrieve a variable from the project's directory on the server.
<b>tweencss</b>	Function used to animate CSS transformations and other values.
<b>createElement</b>	Function that allows creation of DOM nodes (e.g., <code>document.createElement()</code> ).
<b>nCmd</b>	Function used to send a TCP message to another machine.
<b>nCmdAsync</b>	Function used to send a TCP message to another machine asynchronously.
<b>ajaxCmd</b>	Function used to send scripting messages through AJAX.
<b>wsCmd</b>	Function used to send scripting messages through the websocket.

## Modules

Modules are pre-written functions that can be called in the JavaScript areas provided they are linked to first by a require call in the project setup script (as shown below):

```
require('js/modules/SVSi/N_Series.js');
```

There are helpful drop-down menu options in the script editor for adding require statements to your setup script which will list all the available modules by their manufacturer and fill in the necessary file link for you.

Here is a simple example of what would be in the N\_Series Module JavaScript file that would be included if the above require function call was added to your project setup script:

```
modules.SVSi.N_Series = function (ip, port, onError){

    var obj = {};

    obj.ip = ip;
    obj.port = !undef(port) ? port : '50001';

    obj.getStatus = function(async){

        if (async)
            nCmdAsync('tcpclient '+obj.ip+' '+obj.port+' '+asciiToHex('getStatus'),
            function(data){
                obj.status = data;
            });
        else obj.status =
            nCmd('tcpclient '+obj.ip+' '+obj.port+' '+asciiToHex('getStatus'));
    }
    obj.setStream = function(stream){
        nCmd('tcpclient '+obj._ip+' '+asciiToHex('set:'+stream));
    }
    obj.print = function(){
        console.log(obj);
    }
    return obj;
}
```

This function takes an IP address of a machine and a port number and sets up a JavaScript object for interacting with that machine. As shown below, after the JavaScript file is included by the require function, the object could then be created and used in a script on a button or in the project setup script:

```
myN_SeriesObj = SVSi.N_Series('192.0.0.1', '50010');  
myN_SeriesObj.switch(251);
```

Normally you would declare the object in the project setup script and then only call its functions in a button's script.