The Emerald Wave Framework

Web Browser

Contents

1. Introduction 3
2. Web Server 3
3. Node.js 4
4. NPM 4
5. Lua Compiler 4
6. Moonshine Virtual Machine 4
7. Bare Frame Web 5
8. Moonshine Distiller 6
9. Index.html 6
10. Ndlink 7
11. BareFrame Initialization Script 7
12. BareFrame Lua Script 7
13. Resource Images 7
14. Canvas 8

1. Introduction

The Emerald Wave Lua Framework consists of a set of classes that allow the programmer to write Lua script to interact with a user on either a TI-nspire handheld calculator or within a web browser. The Emerald Wave Ndlink module provides communication between the EW Framework and the web browser. The Ndlink front end provides an Application Programming Interface based on the TI-nspire Lua API. The Ndlink back end communicates with the browser via the Document Object Model API.

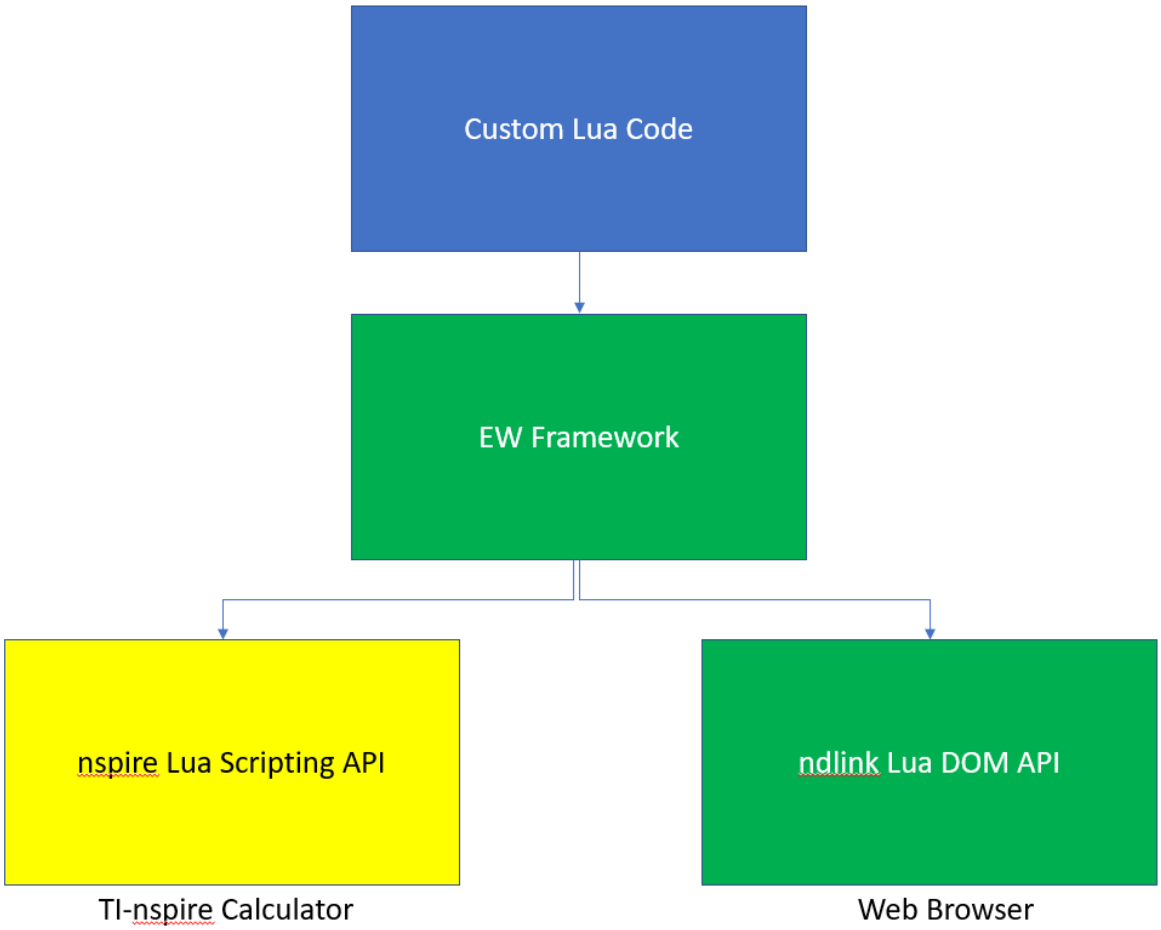

Figure 1 - EW Framework

2. Web Server

To develop and test an EW Framework Lua module running in a browser, the programmer may install a web server on the desktop computer. For example, WAMP is a web server that can be downloaded and installed from this link: (https://sourceforge.net/projects/wampserver/). The remainder of this document will assume a Desktop environment using WAMP as the web server.

To test the WAMP server, open a web browser and enter localhost:80 in the URL bar. The browser then shows the WAMP server home page.

The WAMP server defaults to c:\wamp64\www for all project files. This document assumes that project files are stored in this folder.

3. Node.js

Node.js provides asynchronous, two-way real-time communication between the browser client and the web server. Node.js is a JavaScript runtime environment and is required for the Moonshine Lua VM to run. Node.js can be downloaded and installed from this link: (https://nodejs.org/download/release/v8.13.0/)

4. NPM

NPM is short for Node Package Manager and is used to manage and install software packages. NPM is contained within Node.js, so when Node.js is installed, NPM is also installed.

5. Lua Compiler

The Moonshine distiller (described below) requires a specific version of the Lua compiler. The Lua environment and compiler can be downloaded and installed from this link: (https://github.com/rjpcomputing/luaforwindows/releases/tag/v5.1.5-52)

After the installation of the Lua environment, add a PATH variable by opening Control Panel in Windows and choosing View Advanced System Settings and click the button Environment Variables. Click the New button and enter the path to the file luac.exe which is typically "c:\Program Files (x86)\Lua\Lua 5.1\". This step of adding a PATH variable is needed so that the Moonshine distiller can locate luac.exe.

6. Moonshine Virtual Machine

Moonshine contains a distiller program which converts Lua into bytecode that can be executed by the Moonshine Lua Virtual Machine. The NPM installer contains the Moonshine distiller, so to install the Moonshine distiller on a development machine, open a command prompt and enter the following command:

npm install -g moonshine

The Moonshine distiller has now been installed on the development machine.

7. Bare Frame Web

The Emerald Wave Framework for the web is packaged as a single .zip file called BareFrame.zip, which can be downloaded from this link: (https://lua.emeraldwave.com/downloads/BareFrameWeb.zip). Once the package is downloaded onto the development machine, unzip the file BareFrame.zip into the web server working directory, which is typically c:\wamp64\www. There will now be a folder called c:\wamp64\www\BareFrameWeb which contains all the files necessary to run the EW Framework. The \BareFrameWeb folder contains the following files:

\luavm\moonshine.js

\luavm\DOMAPI.moonshine.js

BareFrame.lua

BareFrame.lua.json

BareFrame_init.lua

BareFrame_init.lua.json

index.html

ndlink.lua

ndlink.lua.json

To test the Framework, open a browser and type “localhost/BareFrameWeb” in the URL bar. If successful, the browser will display, “Hello, World!” in the center and surrounded by white space, as shown below.

Figure 2 - Bare Frame Web

8. Moonshine Distiller

Moonshine contains a distiller program which converts Lua into bytecode that can be executed by the Moonshine virtual machine. Moonshine.js is the Lua bytecode interpreter and DOMAPI.moonshine.js is the extension that communicates with the JavaScript DOMAPI in the web browser.

To distil a Lua script into bytecode, use a command prompt. For example, if Lua script is named BareFrame_init.lua, then enter the following in a command prompt:

moonshine distil BareFrame_init.lua

The output will be BareFrame_init.lua.json which is the Lua bytecode in JSON format.

9. Index.html

By default, the web server opens the file "index.html" at startup. The default index.html provided for Bare Frame Web contains the following lines of HTML markup.

```
<html>
	<head>
		<script src="./luavm/moonshine.js"></script>
		<script src="./luavm/DOMAPI.moonshine.js"></script>
	</head>
	<body>
		<script>
			new shine.VM(shine.DOMAPI).load('./BareFrame_init.lua.json')
		</script>
	</body>
</html>
```

In the <head> section are the two lines that are required to load the Moonshine environment into the browser. In the example above, moonshine.js is the Moonshine Lua VM which is located in the /luavm folder below the current folder. DOMAPI.moonshine.js is the DOMAPI extension which allows the EW Framework to communicate with the Javascript Document Object Module within a browser.

In the <body><script> section is the line which instantiates a new Moonshine virtual machine, while loading the first distilled Lua file. In this example, the name of the distilled Lua file is BareFrame_init.lua.json.

10. Ndlink

The Ndlink module is named ndlink.lua and is written in Lua and provides communication between the EW Framework and the web browser's Document Object Model. The distilled version of this file that runs in the web browser is named ndlink.lua.json.

11. BareFrame Initialization Script

The Lua script that is written for the TI-Nspire should run unmodified in a web browser. However, there is an external Lua script that is needed to start the process. The initialization Lua script in our example is named BareFrame_init.lua and contains the following three lines.

```
require 'ndlink'

require 'BareFrame'

ndlink:start()
```

Ndlink is the distilled Lua module for communicating between the programmer's Lua script and the web browser. BareFrame is the distilled module containing the programmer's Lua script. 'require' is a Lua keyword.

ndlink:start() initiates the Nspire API event loop just as if the Lua module were running on a TI-Nspire calculator. ndlink:start() will initiate and then send the first event to the Lua script, which is on.construction(), which is then handled inside the programmer's Lua script.

BareFrame_init.lua.json is the distilled version of the Lua script that will be executed by the Moonshine virtual machine.

12. BareFrame Lua Script

The Lua script in this example is named BareFrame.lua. This file contains the same Lua code that is used to run a program on a TI-Nspire. BareFrame.lua contains the entire Emerald Wave Framework with zero custom code. BareFrame.lua.json is the distilled version of the Lua script that will be executed by the Moonshine virtual machine.

13. Resource Images

In the TI-Nspire emulator environment, images are loaded into the Resources tab. For the web version, the images are stored in a folder named 'images' which is located directly below the runtime folder. To provide the ndlink module with the names of these files and their display sizes, the Lua initialize script requires additional information. The code snippet below shows how to load the image files for the BasicFrameWeb project.

```
require 'ndlink'

_R = {}

_R.IMG = {}

_R.IMG.hourglass = "images/hourglass.png"

_R.IMG.clipboardicon = "images/clipboardicon.png"

_R.IMG.yellowarrow = "images/yellow_arrow.png"

_R.IMG.images = {  _R.IMG.hourglass, _R.IMG.clipboardicon, _R.IMG.yellowarrow }

_R.IMG.width = { [_R.IMG.hourglass] = 640, [_R.IMG.clipboardicon] = 200, [_R.IMG.yellowarrow] = 225 }

_R.IMG.height = { [_R.IMG.hourglass] = 1280, [_R.IMG.clipboardicon] = 200, [_R.IMG.yellowarrow] = 299 }

_R.IMG.name = { [_R.IMG.hourglass] = "hourglass", [_R.IMG.clipboardicon] = "clipboardicon", [_R.IMG.yellowarrow] = "yellowarrow" }

require 'BasicFrame'

ndlink:start()
```

The variable _R is the TI-Nspire table that contains the resources. The subtable that contains the images is called _R.IMG.

In this example, there are three images: Hourglass, ClipboardIcon, and YellowArrow.

_R.IMG.hourglass sets the name of the resource as it is used in the EW Framework Lua script and sets the location of the image. The folder 'images' is a required location and cannot be changed.

_R.IMG.images = { _R.IMG.hourglass, _R.IMG.clipboardicon, _R.IMG.yellowarrow } creates a table used by ndlink to know which images are available for the Lua module.

_R.IMG.width is a table of the pixel widths for each image.

_R.IMG.height is a table of pixel heights for each image.

_R.IMG.name is a table with the string names for each image which can be used by the programmer.

14. Canvas

In a web environment, the EW Framework uses the HTML element <canvas> for visual output. There must be at least one defined canvas. The default <canvas> name is 'EWFrameCanvas'. The ndlink module creates the canvas.

Canvas ID

The ndlink module will create the canvas ID during ndlink:start(). If the programmer wishes to asynchronously use the canvas ID before calling ndlink:start(), the programmer must define the canvas ID in HTML. For example: <canvas id="MyCanvas”></canvas>

If the programmer has created a canvas ID, then this ID must be passed into ndlink:start(). For example: ndlink:start(“MyCanvas”). The programmer may also simply pass the default name. For example: ndlink:start(“EWFrameCanvas”).

ndlink:start creates a variable called ‘canvasId’.

Width and Height

The default width and height are set at 95% of the window. There are two methods to override the default width and height.

Method 1: The width and the height may be passed as a parameter of ndlink:start(). For example, ndlink:start(“EWFrameCanvas”, 1000, 1000).

Method 2: The width and height may be set using Javascript in the default HTML file. For example:

```
document.getElementById( canvasId ).width = 1000;

document.getElementById( canvasId ).height = 1000;
```

Event Listeners

The programmer may add an event listener to the default HTML file to be able to execute code when an event such as a mouse click or when a window resize occurs. For example, to listen to a resize event, the programmer will use the following Javascript line:

```
var canvasEventListener = document.getElementById( canvasId ).addEventListener( “resize”,
function(e){ … });
```