GM.JS

Written by Chris Anselmo

OVERVIEW

GM.JS is an extension for GameMaker: Studio that allows you to use native JavaScript code when exporting to the HTML5 platform. By using native JavaScript, you can transform your game into a full blown web app. This is ideal when you are hosting your product yourself, rather than creating a game or app to be hosted elsewhere.

❖ *Of course, this extensions works only with the HTML5 export.*

SUPPORT & CONTACT

If you have any comments, feel free to leave one on the **GM.JS** marketplace page. If you have any questions or comments you'd like for me to see directly or privately, I can be reached through my personal e-mail: christopherwk210@gmail.com.

LICENSE

jsExecute(js)

❖ js (string) - Native JavaScript to execute.

This function will allow you to execute native javascript functions. Since GM: S supports multi-line strings, you can easily implement a fair amount of JS while keeping it easy to read.

Example:

```
jsExecute(“
  console.log(‘jsExecute is easy to use.’);

  var elem = document.createElement('h1');
  elem.innerHTML = 'A New Header';
  document.body.appendChild(elem);
“);
```

The example above will log a message to the console, and add a header tag to the bottom of the document body.

❖ *You should avoid creating variables or functions with jsExecute since they aren’t globally available. Instead, use jsAddJs.*

jsAddJs(js)

❖ js (string) - Native JavaScript to execute.

This function is nearly the same as jsExecute, however, this function adds the JS you provide into its own script tag in the head of the document. Keep in mind that each time you call this function, a new script tag is added. JS functions and variables defined here are of global scope, so they can be accessed again by other functions (as long as they are called sequentially).

Example:

```
jsAddJs(“
  function addNumbs(numb1, numb2) {
    return numb1 + numb2;
  }
“);

jsExecute(“
  var myNumb = addNumbs(1,1);
  alert(myNumb);
“);
```

The example above creates a simple function, and then shows a message displaying a sample output of that function.

jsAddCss(css)

❖ css (string) - Native CSS to add to the document.

This function allows you to add a new style tag to the document head using native CSS. Keep in mind that each time you call this function, a new style tag is added.

Example:

```
jsAddCss(“
  body {
    background-color: green;
    padding: 0;
  }
“);
```

The example above adds simple styles to the body of the document.

jsAddLibrary(location)

❖ location (string) - File location or URL of the library you want to add.

This will allow you to reference an external CSS or JS library in your web page. The location you provide should be a local file or URL string, ending in .css or .js. This function will automatically add the appropriate reference in the head tag of the document based on extension (script tag for JS files, and link tag for CSS files). If you provide a string for a file that doesn’t have either of those extensions, (or an invalid file) an error will be logged in the console.

Example:

```
jsAddLibrary(“assets/styles/main.css“);
jsAddLibrary(“https://samplesite.com/libs/script.js“);
```

The example above shows how to add some external libraries. The first line shows how to add a CSS file that is located relatively to the document location. The second line shows how to add a JS file that is located at a specific URL.