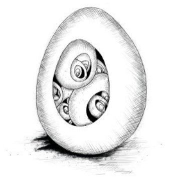
Project : A Programming Language

creating the egg language interpreter



Getting started:

This is a project based on chapter 12 of Eloquent Javascript. To get started create a project called egg, with a file called **`egg.js`** and and **`index.html`** that includes / runs it.

Read through the project at: https://eloquentjavascript.net/12_language.html and paste the different pieces of sample code into your egg.js and make sure they work (starting from the 3rd piece of code there are console.log() lines along with their expected output).

Exercises:

The book provides some of its own exercises, but before we get to those lets first do some cleanup.

- The code currently pollutes the global namespace. Create a namespace (object) called **`egg`**, and move all the global functions and objects into it, refactoring the code as needed. Be sure to check that everything still works.
- Install jsdoc with: **`npm install -g jsdoc`** and write in code documentation comments for each function. See appendix A for a guide on writing jsdoc comments. Be sure to document each function's parameters, return, possible exceptions thrown. Once you've written jsdoc you can generate a documentation site with the command: **`jsdoc egg.js`**
- Write mocha tests for each of the functions. In other words add the following to your index.html to setup mocha and chai. Then also create a **`test.js`** file with your tests and include it in your index.html after where you've included egg.js

```html
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/mocha/3.2.0/mocha.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/mocha/3.2.0/mocha.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/chai/3.5.0/chai.js"></script>
<script>
    // initialize mocha
    mocha.setup('bdd');
    // chai has a lot of stuff, let's make assert global
    let assert = chai.assert;
</script>
```

- Do the exercises (arrays, closure, comments, fixing scope) listed at the bottom of the project description: https://eloquentjavascript.net/12_language.html Be sure to also add jsdoc and mocha tests for the new features. Note that during the last exercise (fixing scope) many of your tests will break as you add the new feature (set).
- On the **`index.html`** file of your jsdoc site add a "try-it-now" feature similar to what you find on w3schools. In other words: create a text area of the left where people can write egg code, a text area on the right into output (the result of print) is placed, and a button on top to run the code. Hint: on your index.html file you will probably want give **`egg.topScope.print`** a different implementation in order to get your output into right textarea.

Appendix A: JSDoc

JSDoc creates a website with documentation based on special comments (it looks for comments that start with /**) in the code. An example comment and function are listed below.

```
/**
 * General description about the function goes here. Something like:
 * Adds the two provided numbers
 *
 * @param {number} a description of this parameter here
 * @param {number} b description of this parameter here
 * @returns {number} description of the return here
 * @throws {SyntaxError} when one or both parameters are not a number
 */
function add(a, b) {
     if (typeof(a) !== "number" || typeof(b) !== "number") {
          throw SyntaxError("wrong number of arguments for add");
     }
     return a + b;
}
```

Note the use of @param, @returns, and @throws to describe what the parameters, return, and optional errors thrown are. For each of these their type is included in curly braces { }.

Sometimes a parameter can be multiple types. In that case you can use a type union, example: {(number|boolean)}. For more on types, such as nullable, non-nullable, variable number of parameters, optional parameters, callbacks, see: https://jsdoc.app/tags-type.html

Appendix A continues on the next page.

Documenting Namespaces in JSDoc

It's important to properly document the existance of namespaces, JSDoc will create a separate html page for each namespace. An example of the syntax is listed below.

```
/**
 * The mystuff namespace that contains all my stuff
 *
 * @namespace mystuff
 */
let mystuff = Object.create(null);


/**
 * JSDoc should automatically realize that this function is in mystuff
 */
mystuff.myFunc = function () { }


/**
 * If it doesn’t, or you want to be extra clear, you can use:
 * @memberof myStuff
 * @method myFunc2
 */
mystuff.myFunc2 = function () { }
```

Nested Namespaces

You can also create nested namespaces. A separate page will also be created for each nested namespace.

```
/**
 * @namespace myStuff.extraStuff
 */
myStuff.extraStuff = Object.create(null);

/**
 * @memberof myStuff.extraStuff
 * @method extraThing
 */
myStuff.extraStuff.extraThing = function() {}
```