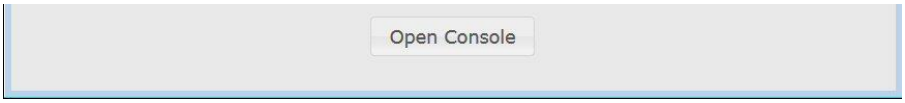
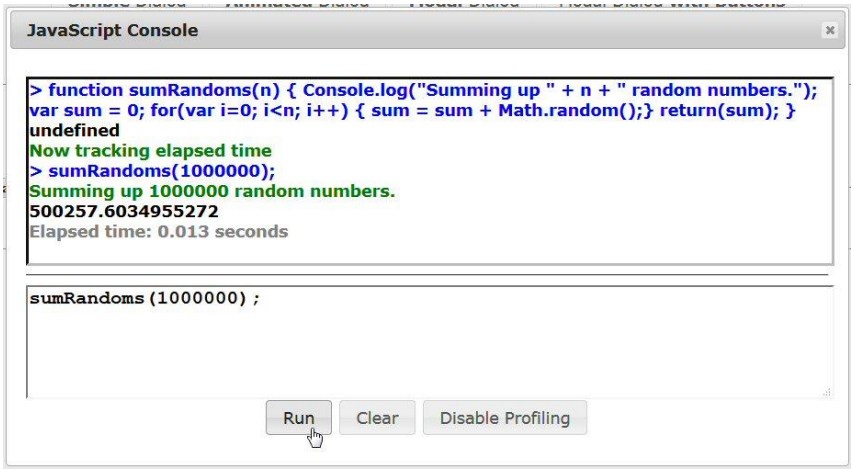
Coding Challenge Overview: Portable JavaScript Console

The goal of this project is to create an easy way for the user to pop up a dialog box with an interactive JavaScript console, similar to the console in Firebug or the Google Chrome Developer tools, but that works identically in any browser. Simply loading the appropriate JavaScript file in a page that uses jQuery UI should result in a button being automatically inserted at the bottom of the page, as below.



Pressing the button should result in a jQuery UI Dialog box that lets the user interactively evaluate JavaScript and jQuery expressions and statements, as below.

There should be an option that lets the user turn on performance profiling, where the elapsed time for each entered command is shown. There should also be a logging function that prints simple Strings to the console output window.



The code should assume that jQuery and jQuery UI have already been loaded in the page, but make no other assumptions about the JavaScript or HTML in the page. It should work identically in any browser that supports jQuery and jQuery UI.

Coding Challenge Requirements: Portable JavaScript Console

1. **Do not use HTML in the body.** The entire process should be accomplished simply by loading a specified JavaScript file in any otherwise normal page that uses jQuery UI. For the console itself, there should be no no markup of any sort in the body, and nothing other than a single <script> tag in the head. You may, however, assume that jQuery and jQuery UI have already been loaded.
2. **Avoid excessive HTML inside JavaScript strings.** Instead, use Ajax to load the button and other elements from a file that is inside the folder containing your JavaScript file. You may assume that you know the location of the folder containing your code, but even better would be to define the location in a variable that can be overriden.
3. **Use jQuery UI styles.** All buttons, dialogs, and other elements used in the console should use the current jQuery UI theme.
4. **Create style sheet links dynamically.** Do not require a <link...> element in the head for the styles specific to the console. Instead, insert the <link> dynamically. Hint: use $("head").append.
5. **Use the global context.** That is, you should be able to enter "var x = 5;" in the console, then later enter "x" and get back 5. Hint: use window.eval instead of eval (and Google for "eval global scope" for why this is even an issue). Note that $.globalEval in jQuery does not work for this problem, since it does not return a value.
6. **Handle JavaScript errors.** If the end user enters illegal JavaScript, your console output window should show the error message. Hint: research try/catch blocks in JavaScript.

```
> Math.random()
0.08269640119232557
> Math.randoom()
TypeError: Math.randoom is not a function
```

7. **Prevent name conflicts.** Define all your functions and variables in a namespace.
8. **Handle < and > in the output.** Do not fail with JavaScript commands that contain < or >. Hint: see string.replace.

```
> var x = 5;
undefined
> if (x < 10) { Console.log("yes"); }
yes
undefined
```

9. **Scroll appropriately.** If the user enters many commands in the input window, be sure the output window shows the latest one. Hint: research the scrollTop() function and the scrollHeight property.
10. **Show times in seconds.** If elapsed time profile is enabled, display the elapsed time in seconds. Hint: research Date.now() or [better!] performance.now().
11. **Maximize the width of the dialog.** Size it relative to the current browser width (e.g., 90%). Hint: research how to determine the browser width.
12. **Think of interesting new features and try them out.** Be creative.