Web Workers

Overview

In this lab, you'll implement a multi-threaded Web application that calculates prime numbers. The user will be able to find all the prime numbers in a specified range. The user will be able to kick off several "searches" at once – each goes into a different thread (and the user will be able to cancel a "search" if they want to).

Roadmap

There are 3 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Understanding the starter project
2. Performing multithreading
3. Additional suggestions (if time permits)

Server

HTML files can be opened from the file system, but for security reasons browsers will limit the possibilities. It is better to open a server in the project folders. Here are two ways to do that.

Visual Studio Code

1. Open Visual Studio Code
2. Choose menu "File | Open…" and open the root folder of the course material. After opening you should see a folder called ".vscode" at the top of the tree in the Explorer on the left.
3. Choose menu "Tasks | Run Task…" and choose task "npm: install".
4. Select/Open a file in the folder that you want to be the root of your server. This will generally be the homepage of your app.
5. Choose menu "Tasks | Run Build Task…" or choose the shortcut.
6. To close the server again, choose menu "Tasks | Terminate Task…".

Command line

1. Globally install "live-server" with command "npm install -g live-server" (see: https://www.npmjs.com/package/live-server)
2. Open the command line (Windows) or terminal (macOS) in the folder that you want to be the root of your server. This will generally be the homepage of your app.
3. Run "live-server" to open the server. Choose Ctrl+c to close the server again.

Exercise 1: Understanding the starter project

Start Visual Studio and open the *start* project for this lab as follows:

- *Folder:*
  `\start`

Open `index.html`. This is the home page (and only page) in the Web application. It has the following UI elements, plus a bit of on-load initialization code:

- A text box, where the user can enter a number
- A button named *Find Primes*
- A button named *Cancel*
- A text area named *Result*

The project also has a helper script file named `utilities.js`, to determine if a given number is prime.

Exercise 2: Performing multithreading

Handle the click event on the *Find Primes* button as follows:

- Kick-off a Web Worker to calculate all the prime numbers between 1 and the number specified in the textbox.
- The Web Worker should loop through the number range, and add prime numbers to an array like this:

  ```
  var primes = [];   // Empty array initially
  ...
  primes[count++] = aPrimeNumer;
  ```

- The Web Worker should post this array back to the main page when done.
- The main page should retrieve the data, and display in the text area.

Run the application and test that it works correctly.

Exercise 3 (if time permits): Additional suggestions

- Allow the user to cancel the current "Find Primes" operation.
- Allow the user to kick off several "Find Primes" operations simultaneously (how will you display the results...?).