Section 1: JavaScript and TypeScript

Sep 29, 2016

Install node.js

Node.js is an installation of JavaScript that runs in your terminal, not in your browser. The independence from the browser simplifies development of non-web programs. Node.js is also suitable for writing server programs because its VM is very efficient. Read more at node.js.

Install node.js on your laptop from this page. We will be using version 4.6.

This installation includes also the package manager npm which gives you access to code developed by thousands of programmers.

Exercise To learn something relevant to this course, use npm to find JavaScript packages related to DSLs (domain specific languages):

```
$ npm search DSL
```

This command may take a while if you are running npm search for the first time, so execute this command from a separate terminal window.

Install git

Git is a version control software that we will use to check out the code for today's section and the starter kits for the homeworks. If your machine does not have git (test it by executing git --version from the command line), install it using these instructions.

Exercise Pull the code for this section from Julie's repository.

```
$ git clone https://github.com/jn80842/cse401section1.git
```

This will clone a JavaScript code snippet into working directory. This code will be in the directory cse401section1. Change to that directory and open the file section1.js with your favorite editor. We assume here that you'll use Sublime.

Executing a JavaScript program from an editor

In order to run your JavaScript file from the Sublime editor, you can use this wikihow to set up a new "build environment" in Sublime. **Follow method 2!**

You can also start a REPL at the command line by typing node, or even open Chrome and use its console by right-clicking and choosing 'inspect'.

Exercise Set up the build environment per this wikihow. Next, use console.log(expression) to print values to the console. For example, print the result of the map operation:

```
console.log(a.map(times2));
```

What output do you see when you print a function value, e.g., times2?

A quick review of JavaScript

We are now ready to go over JavaScript's first-class functions and prototypes, as well as a simple call chaining example. We will do this by going over the file section1.js line by line.

Exercise Follow the comments in the file. Add print statements to observe the values of individual expressions. Make sure you understand each line of code before you proceed to the next one.

A quick review of TypeScript

TypeScript is a strict superset of JavaScript that offers optional static typing and some other useful features. Now that you have node and npm installed, you can install it easily:

```
$ npm install -g typescript
```

TypeScript compiles to JavaScript with the tsc command. See https://www.typescriptlang.org for documentation and code samples.

Exercise Add type annotations to the section1.ts file in the repo above.

An introduction to Homework 1

In homework 1, we will be implementing a small, readable regular expression DSL. Today, we'll set up our repositories and take a quick look at the code. Our assignments will be hosted on Bitbucket (bitbucket.org). If you don't have one, create an account, then fork our repository at https://bitbucket.org/csep590c_sp16_overlord/cse401_au16_overlord. Make sure your

repository is private! Clone your repository by clicking the actions button (the … icon on the left), copying the `git clone [repo]` command and pasting it at your command line.

Once your code is set up, navigate to the hw1/engexp folder and run `npm install` to install all dependencies, then `npm test` to run the test suite. The tests are written in the Chai.js framework (another DSL!); see the documentation at http://chaijs.com/.

Exercise Add one failing and one passing test. Commit your changes and push, then check the bitbucket site to make sure your commit appears.

Aside from writing tests, you may want to experiment with or debug your code directly using the node REPL. At the command line, type `node`, then copy line 4 of the test/engexp.js file and paste it in the REPL to load the EngExp code (make sure the path is correct!).

```
$ node
> var engexp_1 = require("../src/engexp");
```

Node also has a debugger similar to gdb if you prefer (https://nodejs.org/api/debugger.html).