SSUI P3 Writeup

P3: FSMs and Interactive Objects

- **due Oct 31st, 11:59PM on Canvas**

Please upload your entire project directory as a `.zip` file and include a link to your github on canvas

In this assignment, you will be modifying files to build FSM-based Interactor objects by filling out code in files specified in `Files to Modify` and create standard interactor objects using said code in `./src/test_cases.ts`. Finally, we ask that you create your own interactor objects -- for interactors deemed "cool" - you may be rewarded with some bonus points. :-)

(note: your interactor does **not** have to be useful)

Preliminaries

Installation Prerequisites

As with P2, we will be using the same webserver-Chrome browser tandem for running code from P3.

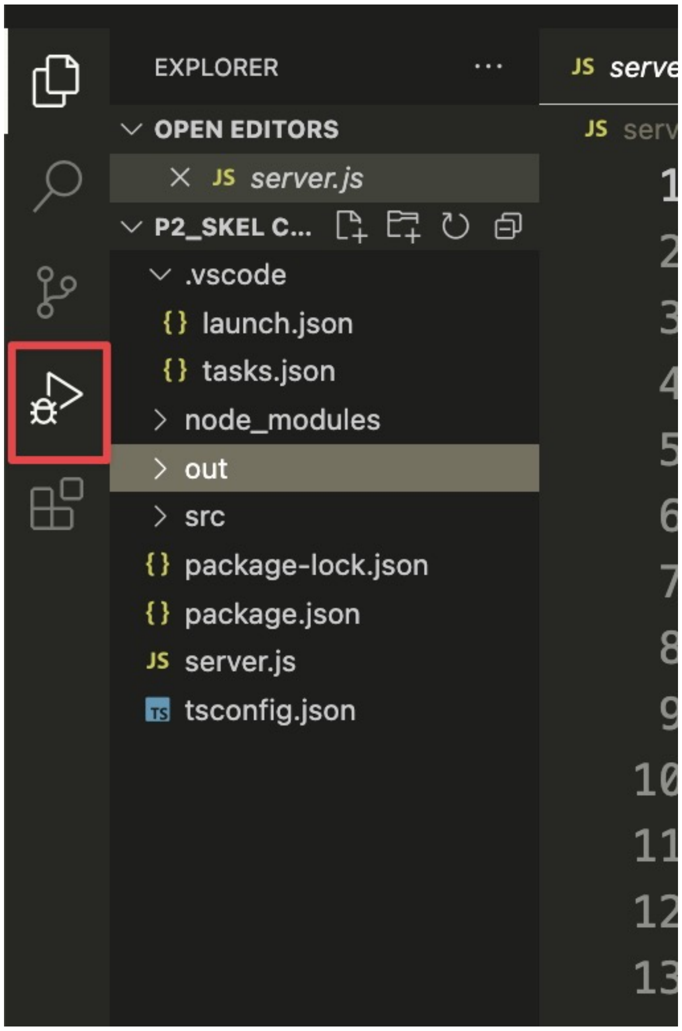
Installing the webserver
`npm install express`

For more details, please consult slides 6-13 from Lecture 5.

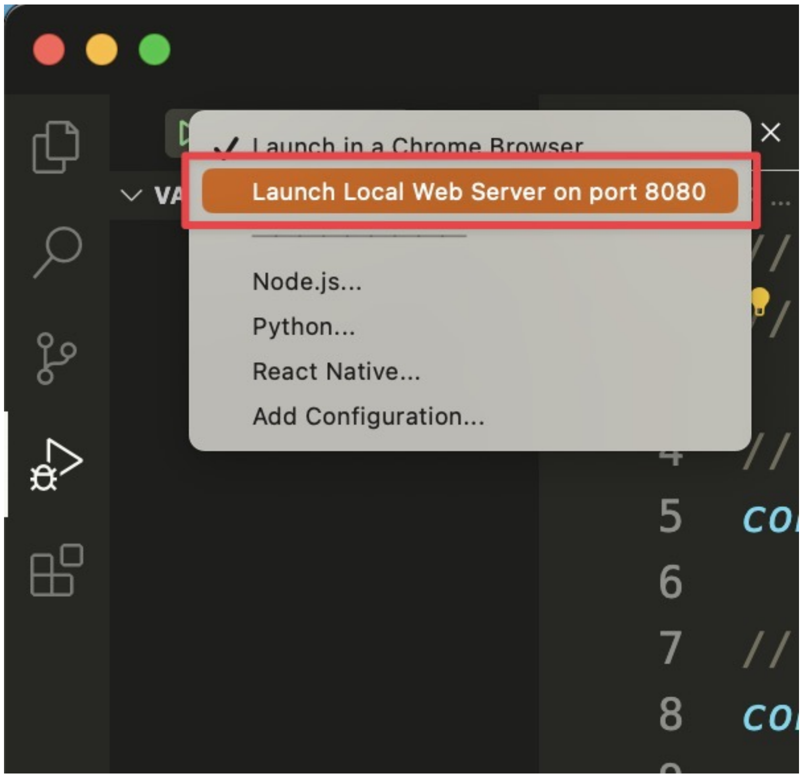
How to run

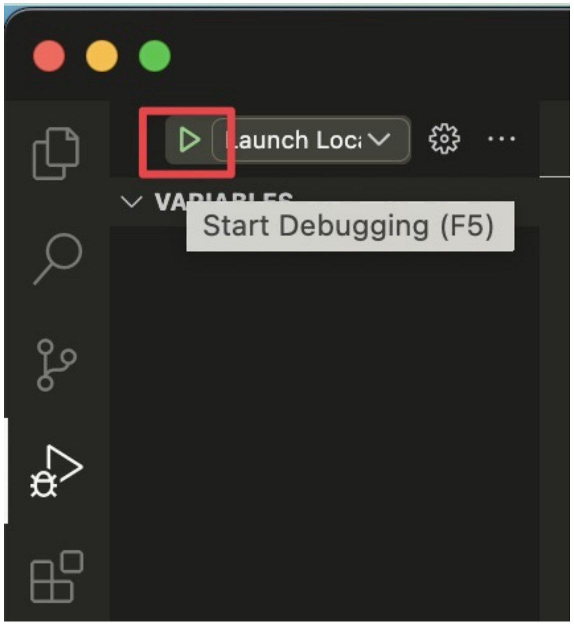
1. Transpile all `.ts` to `.js` and `.js.map` files with `tsc`

2. Navigate to the Debug and Run panel in VSCode

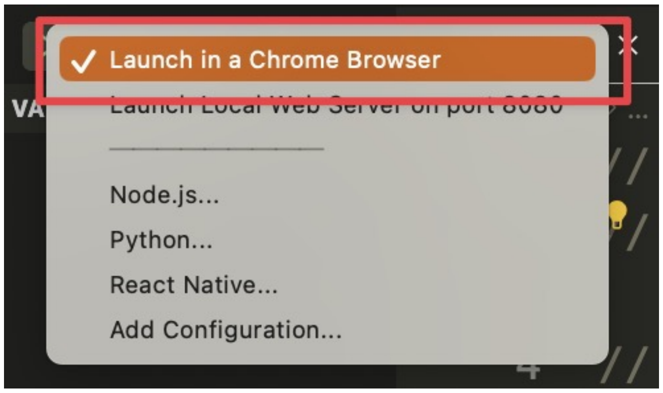


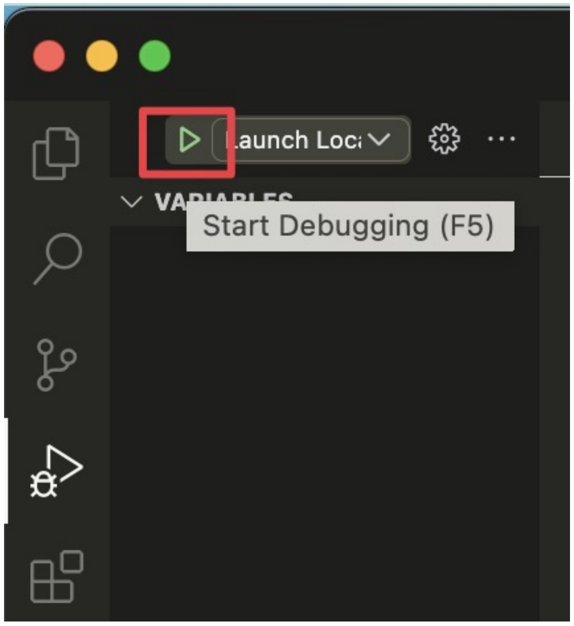
3. Activate the web server in VSCode


Launch in a Chrome Browser
Launch Local Web Server on port 8080
Node.js...
Python...
React Native...
Add Configuration...



4. Activate the Project in Chrome


✓ Launch in a Chrome Browser
Launch Local Web Server on port 8080
Node.js...
Python...
React Native...
Add Configuration...



which should launch a Chrome browser window running test cases from `src/test_cases.ts`

Examples of Expected Behavior

If you have run the project correctly in its default state (e.g. no code has been implemented) - you should a blank page (with a blank square), with output in the browser's console.

When run, the default behavior is to produces a dump on the console of the tree created in `test_cases.ts`, but otherwise produces no output.

NOTE: If you are having issues with launching the server, try and line 19 in `.vscode/launch.json` to `"program": "${workspaceFolder}/server.js"` (rather than what was there previously `"program": "${workspaceFolder}\\server.js"`)

Classes and Descriptions

Files to modify

all files you are expected to complete and modify will be in the `./src` directory.
Note: Please take time to read the code in all of the files, not just those below!
- `./src/Action.ts`
- `./src/EventSpec.ts`
- `./src/FSM.ts`
- `./src/FSMInteractor.ts`
- `./src/Region.ts`
- `./src/Root.ts`
- `./src/Transition.ts`

as before, each method or subsection of code which expects your input/code will be marked by a `//=== YOUR CODE HERE ===`

All testing of the system can be done by creating instances of the above object classes in `test_cases.ts` -- please look over the given examples to get an idea of how to create your own test cases.

A brief explanation of each class to implement is provided here, though the provided code has much more detailed documentation.

Action

Class specifying what occurs when an FSM transition is activated.

EventSpec

Defines aspects of user events (essentially defines user event handlers).

FSM

Base class defining logic for the finite state machine underpinning all of the `FSMInteractor` Objects.

FSMInteractor

Base class for all Interactor Objects.

Region

Defines areas of the screen in Interactors associate with.

Root

Much like TopObject in P2, this class acts as the root of the tree containing all `FSMInteractor` objects. Thus it manages changes, and keeps track of damaged areas.

Transition

Represents (a single) transition between states in an FSM.

Standard Objects to Create

We leave the appearance of each standard interactor up to your preference/implementation.

Checkbox

A standard checkbox. ☑

Radio Button(s)

These are traditionally empty circles with a text label next to them, where a grouping of several radio buttons will only allow one of the group to be selected at any given time.

Rotary Dial

A circular dial, with an indicator on some part of the circle/dial, with labels around the outer edge of the dial. (Think of a dial to set the heat on your range/oven.)

Button

A standard button, exhibiting behavior as defined in Lecture 7 Slides, pg 22-26.

Your Own Interactor

The design, functionality, implementation of your own Interactor is up to you, so long as they are an instance of `FSMInteractor`. We encourage you to have fun!

Evaluation

P3 will be evaluated on the basis of technical correctness, primarily with your examples of standard interactor objects (e.g. the standard interactor objects should behave as expected). About 10-15% will be reserved for completion of your own interactor, with the possibility of extra points on "coolness".

As before, some portion (around 10%) of the grade will be used to evaluate overall code hygiene -- so please comment your code!