

TL14 - Introduction to Authoring Extensions for Launch, by Adobe

We'll learn how to write a simple Extension for Launch, by Adobe.

The Extension will contain one Action — it will wait a couple of seconds, then produce a popup alert.

Scaffolding

Extensions are little packages. They follow a format, based on [CommonJS modules](http://wiki.commonjs.org/wiki/Modules/1.1.1) (<http://wiki.commonjs.org/wiki/Modules/1.1.1>), which means there are a couple of configuration files we need to provide.

Thankfully, this is in large part automated.

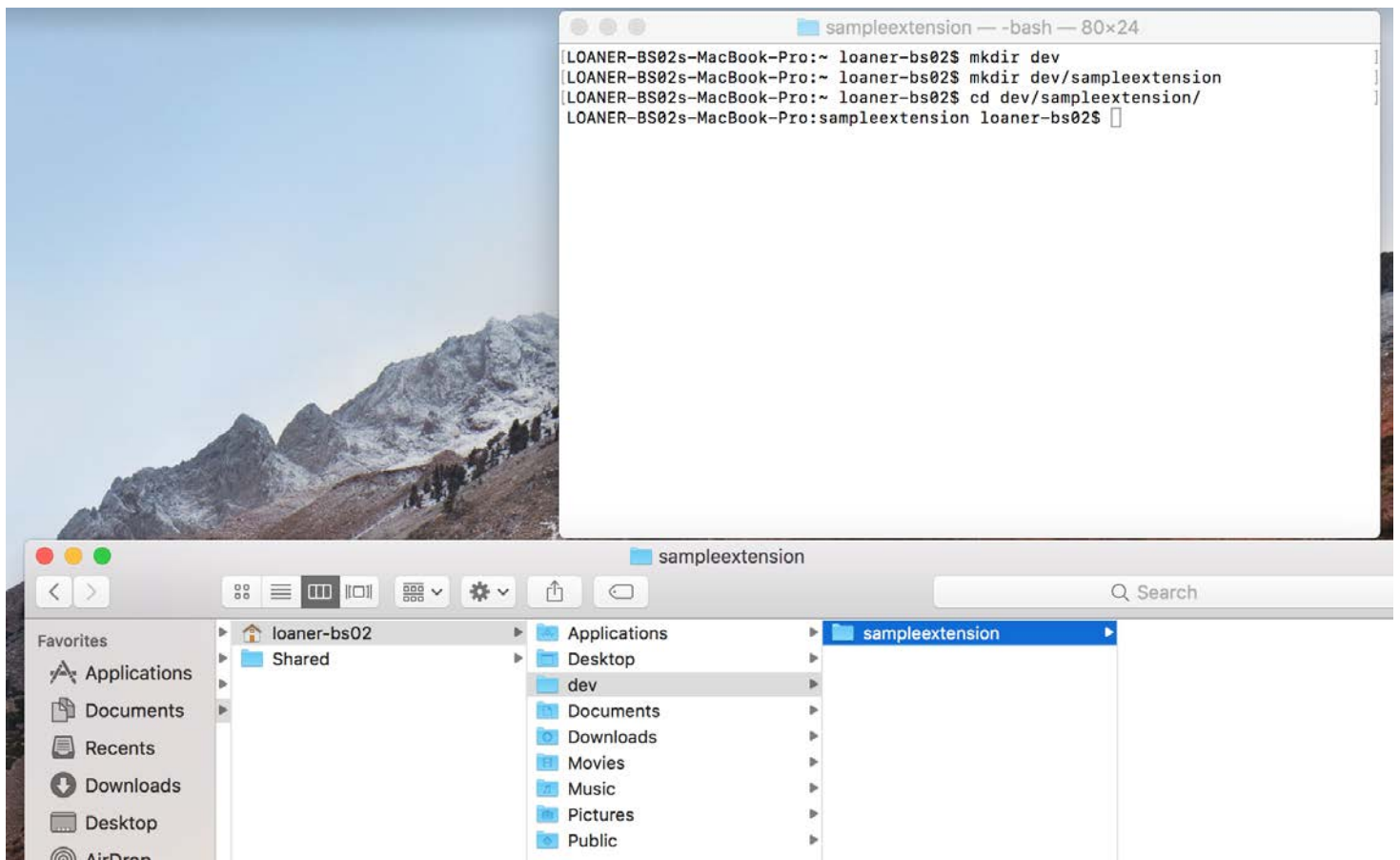
Create a project folder

Create a project folder first.

1. Open a Terminal window
2. Type the following commands:

```
mkdir dev  
mkdir dev/sampleextension  
cd dev/sampleextension
```

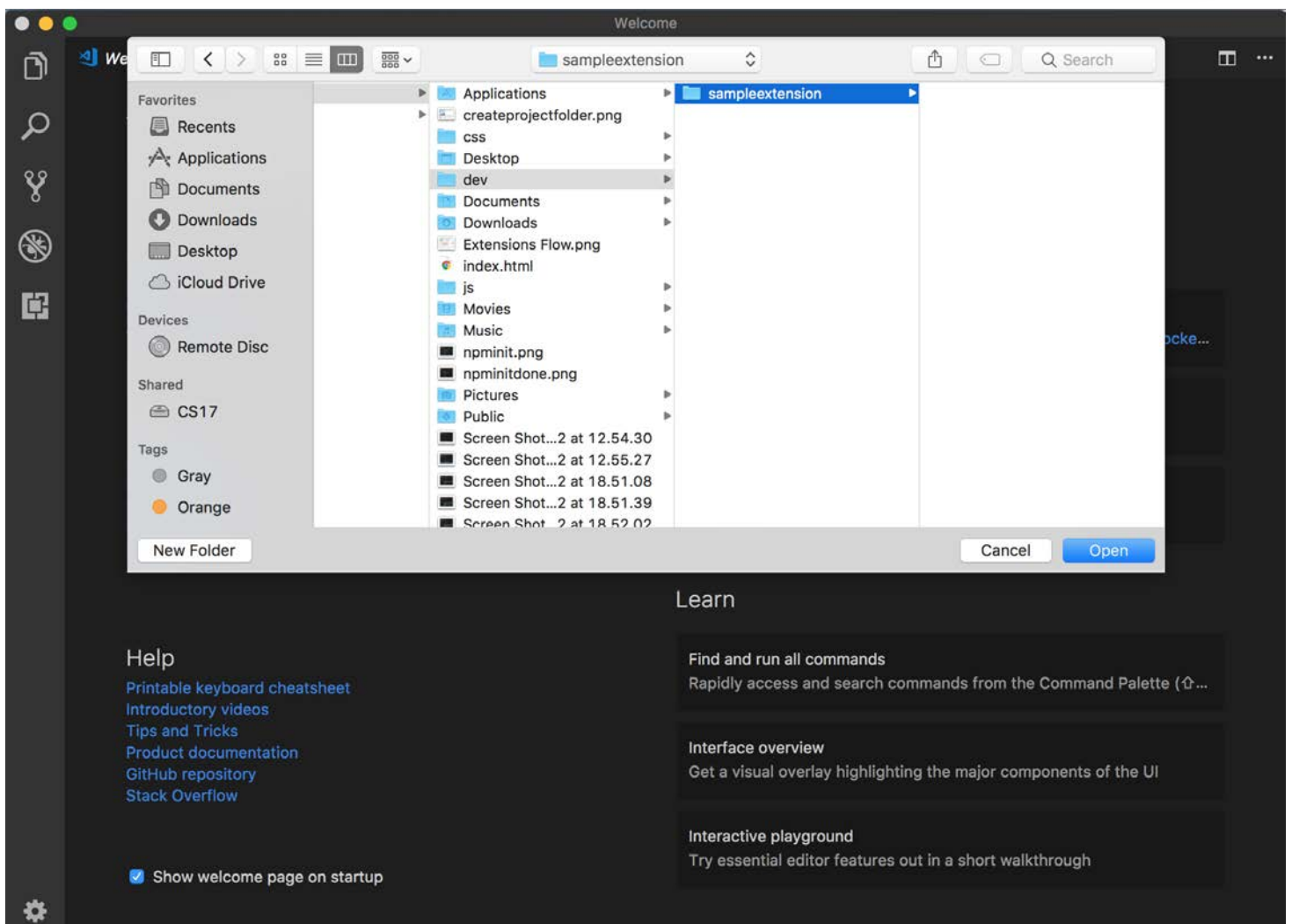
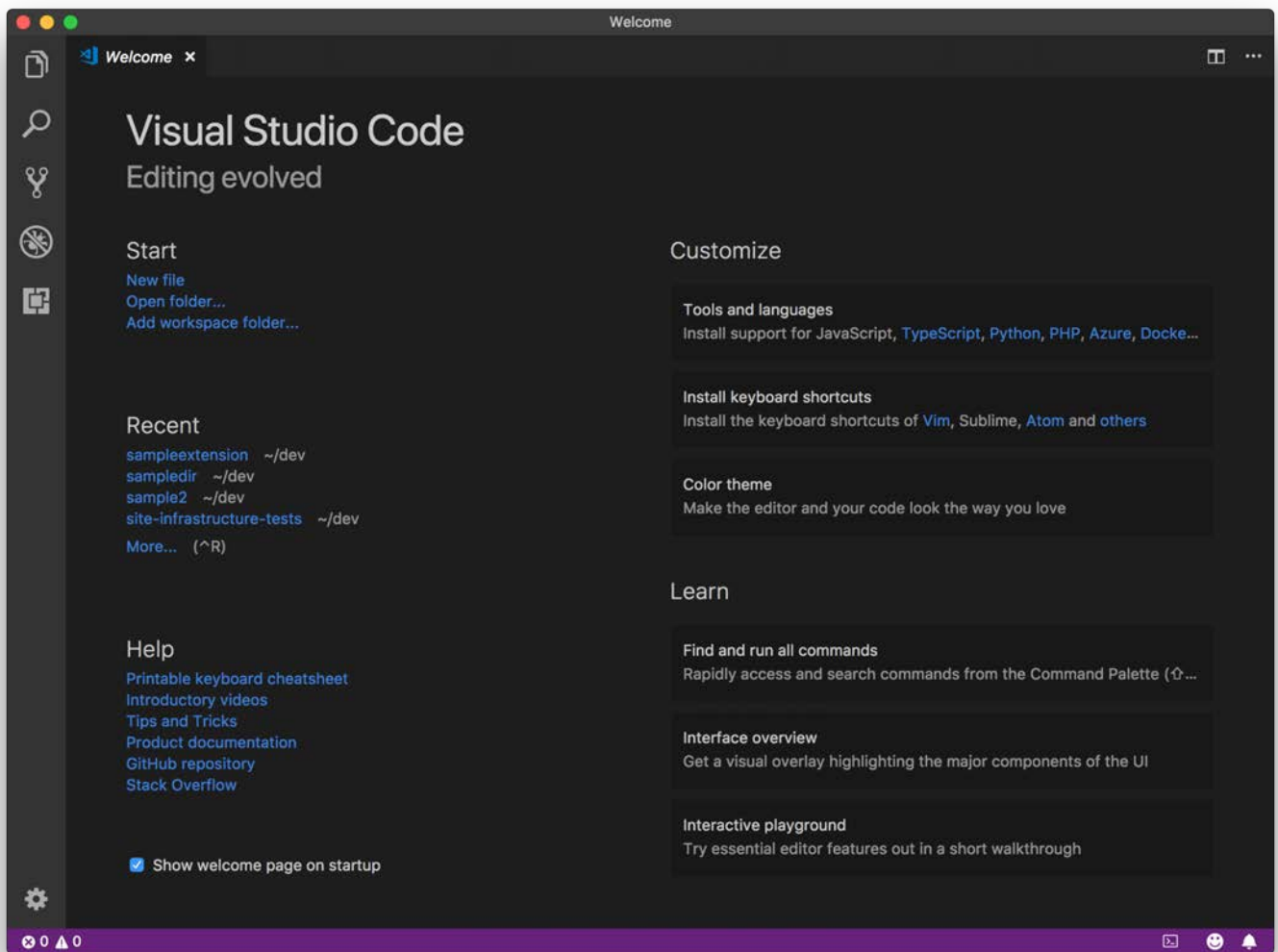
If there are no errors, you should have created two folders, as follows:



You can close the Terminal now, if you want. We'll handle any other command line work within our editor.

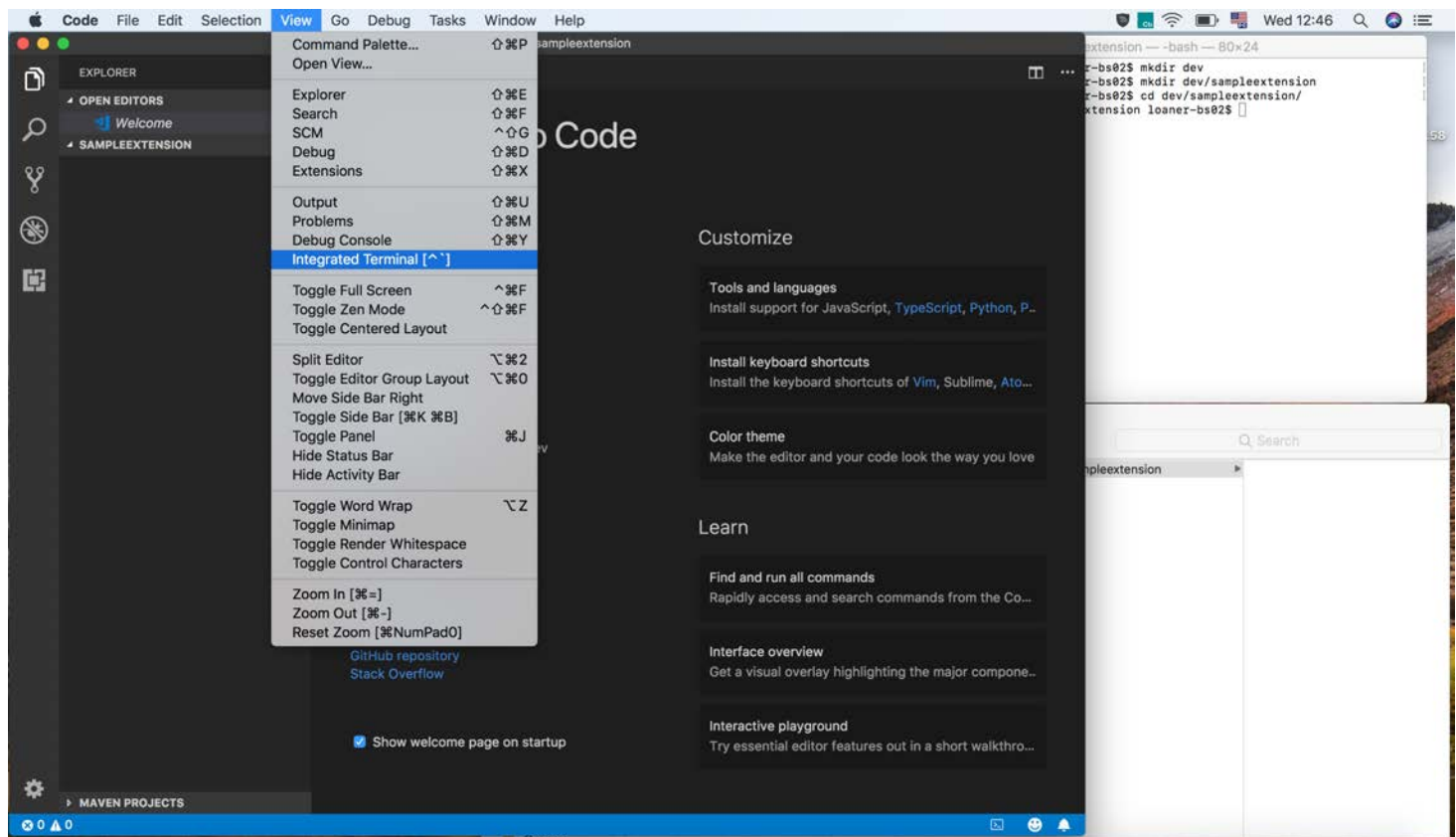
3. Start Visual Studio Code

4. Click "Open Folder..." on the left, choose the folder you made, click "Open"

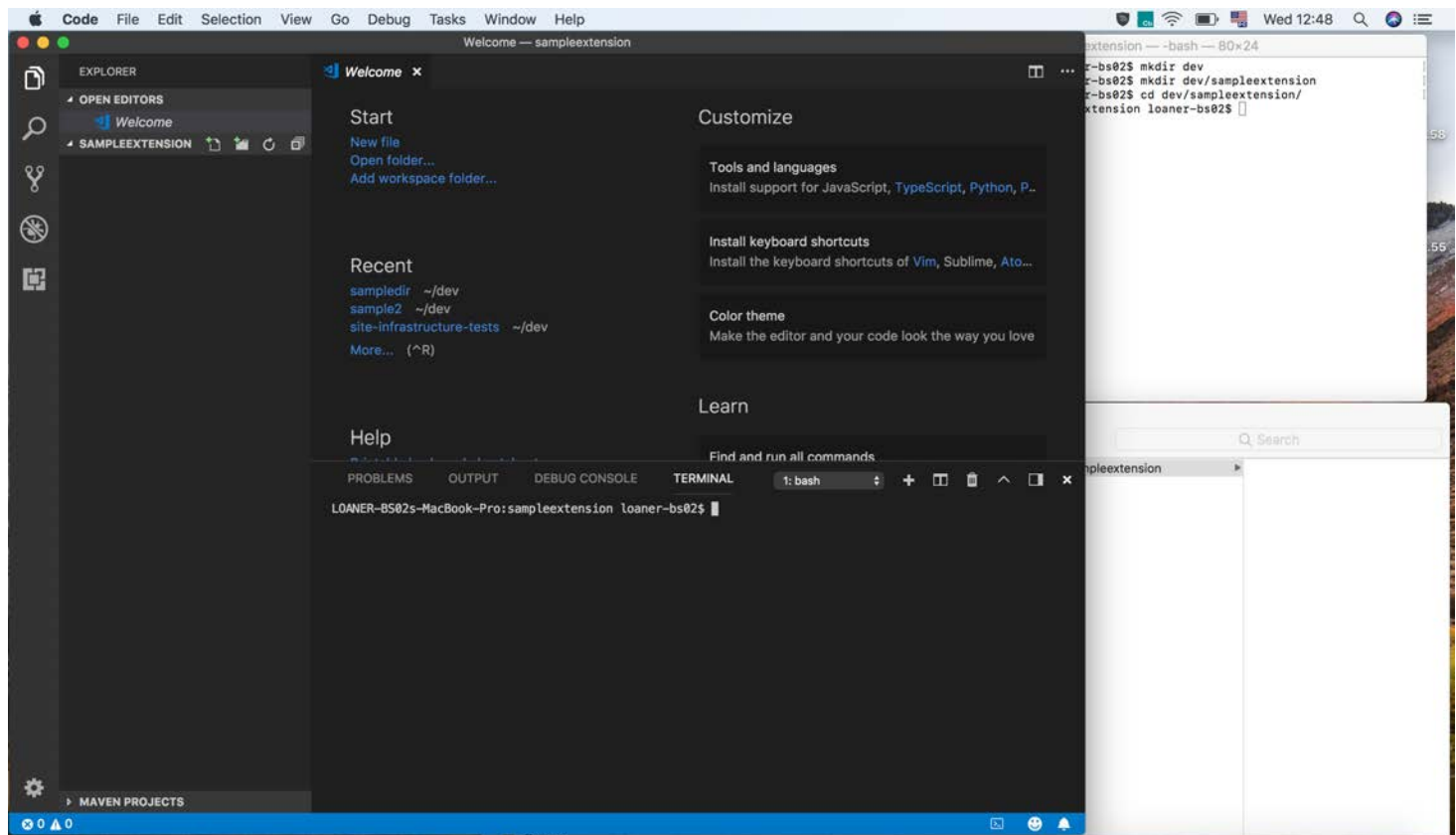


We are now ready to make our Extension.

5. From the "View" menu, select "Integrated Terminal"



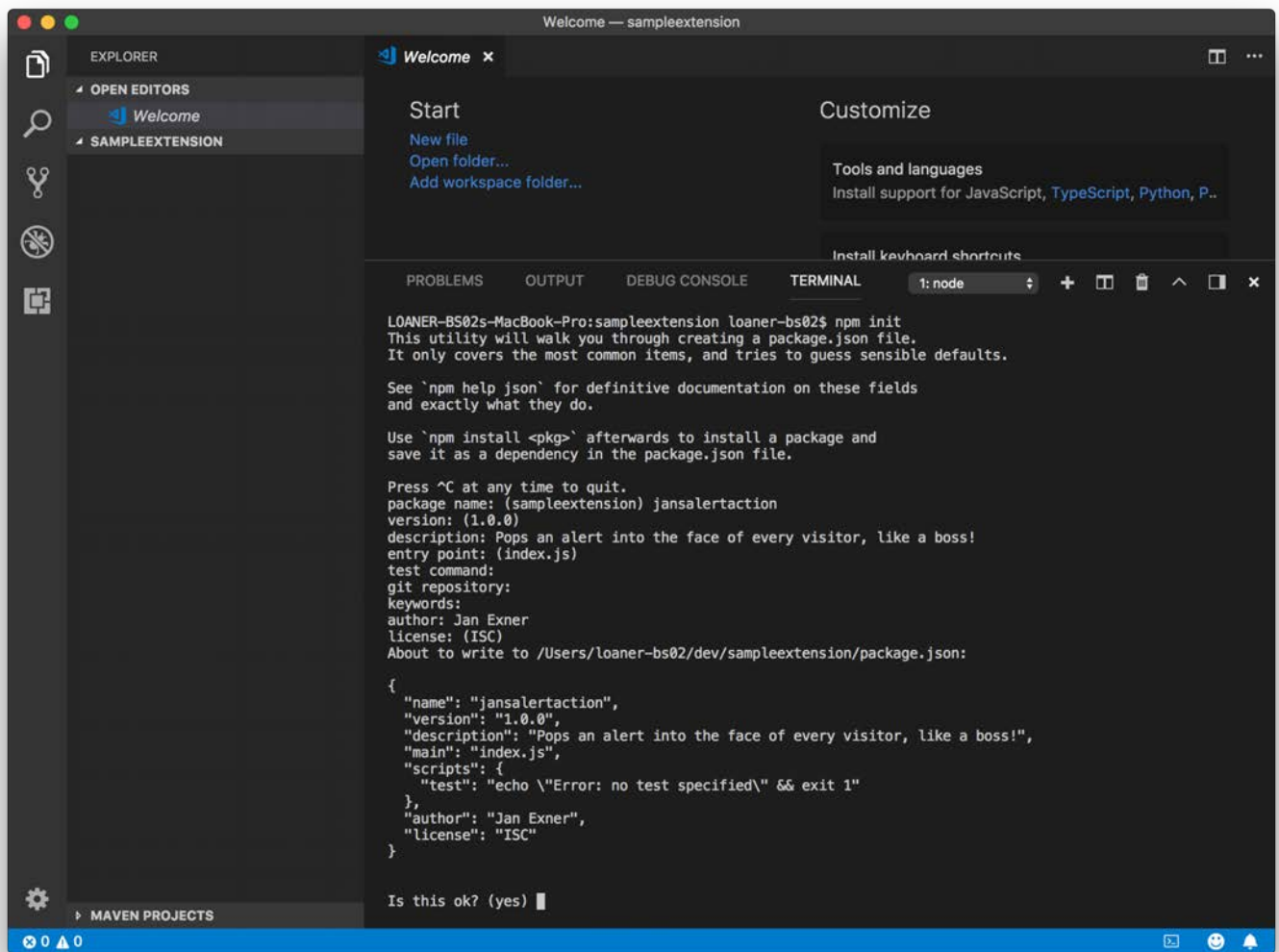
As a result, you will have a command line on the bottom right of Visual Studio Code



The next step is the semi-automatic creation of the extension project structure.

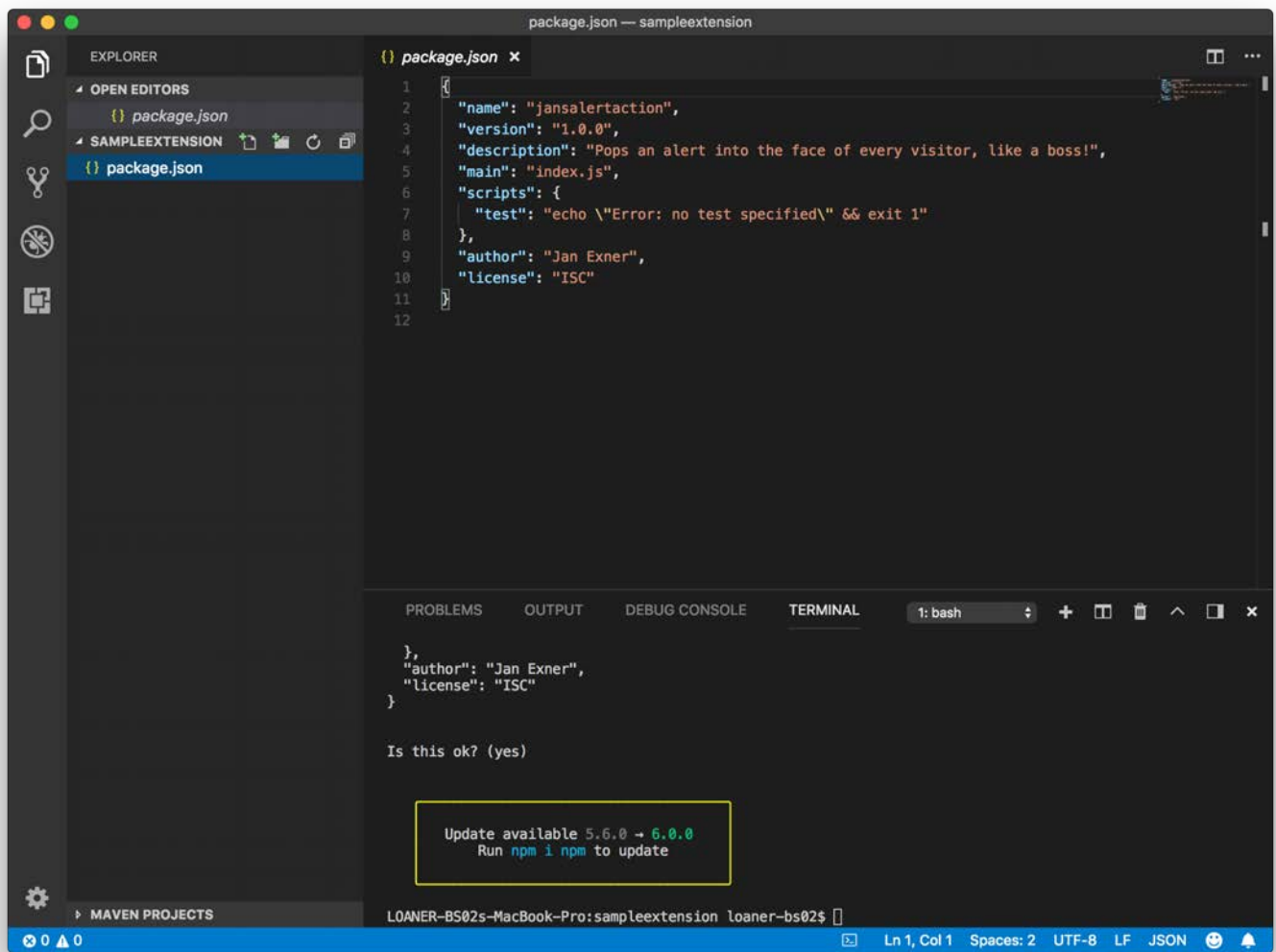
6. Type `npm init` into the terminal

7. Provide reasonable answers for the prompts (feel free to follow the screenshot below)



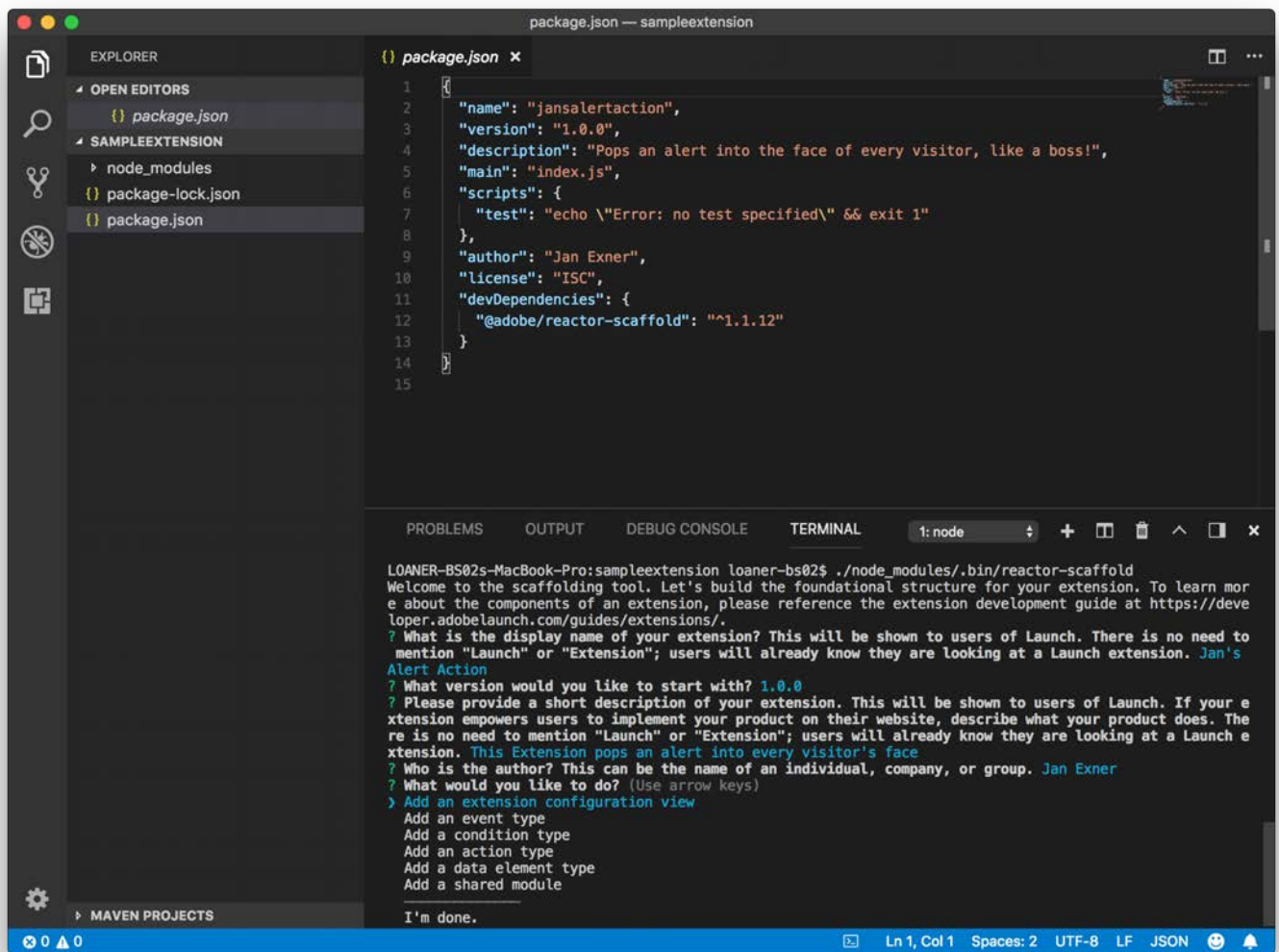
8. At the final prompt, press

We now have a basic structure, note that there is now a file in your folder called "package.json".



Onwards...

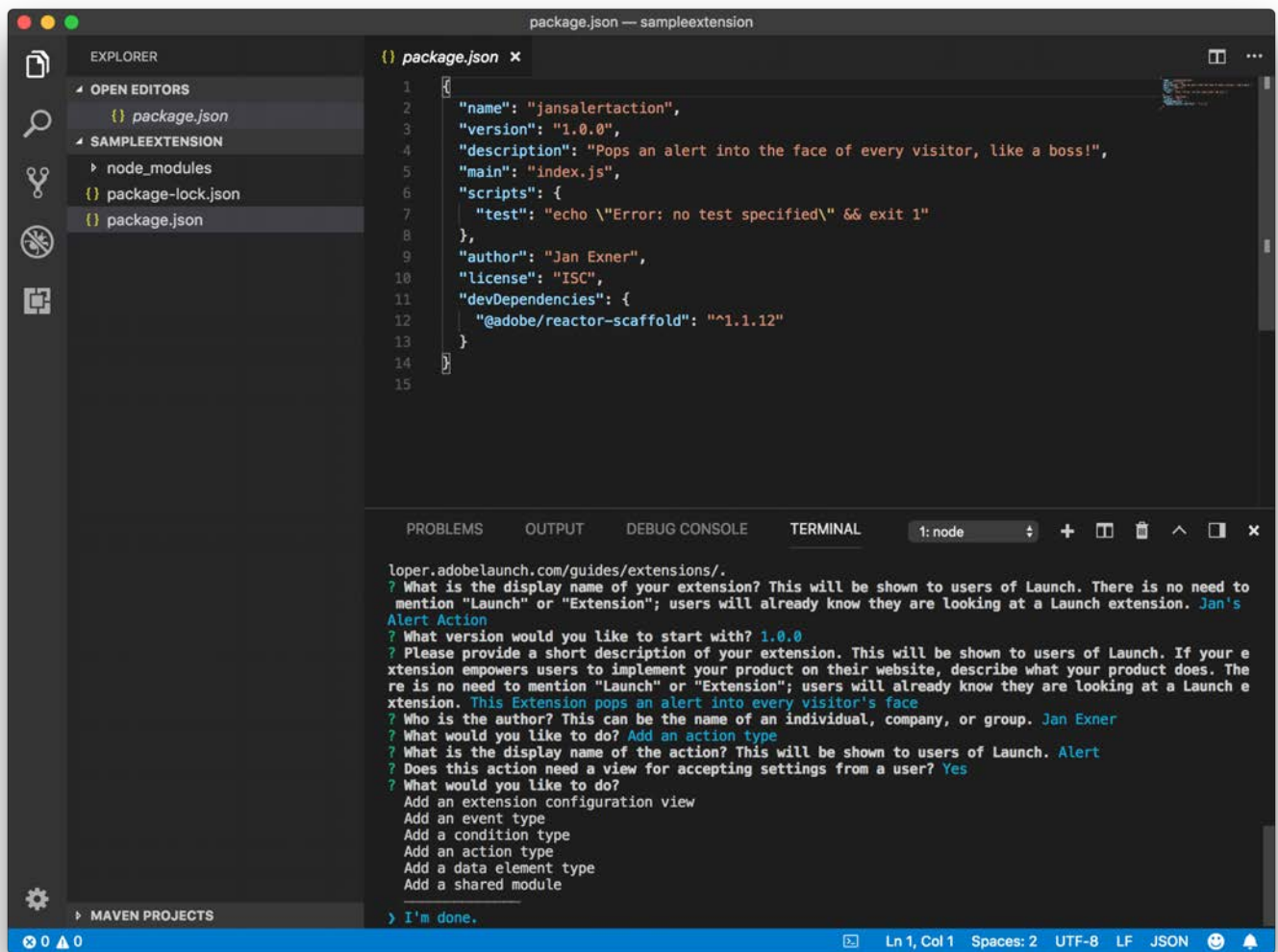
9. Type `npm install @adobe/reactor-scaffold --save-dev` into the command line
10. Type `node_modules/.bin/reactor-scaffold` into the command line
11. Provide reasonable answer at the prompts (feel free to follow the screenshot below)



The scaffolding tool now presents you with a sort of menu. You can navigate the menu with the up and down arrow keys, and select an option using

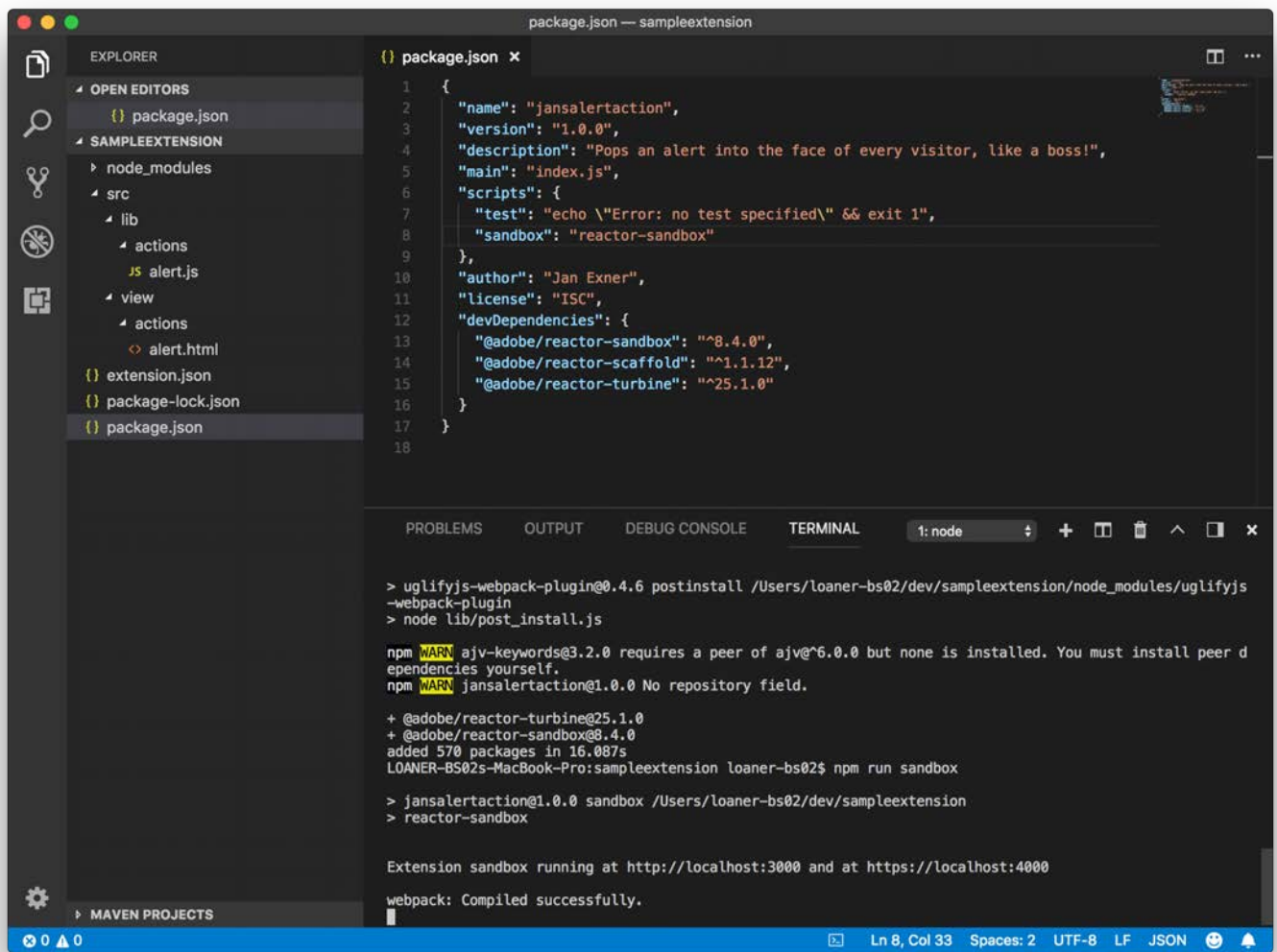
12. Move down to "Add an action type" and press

13. Provide reasonable answers at the prompts. "Alert" is a good display name and make sure you say "y" to the second question!



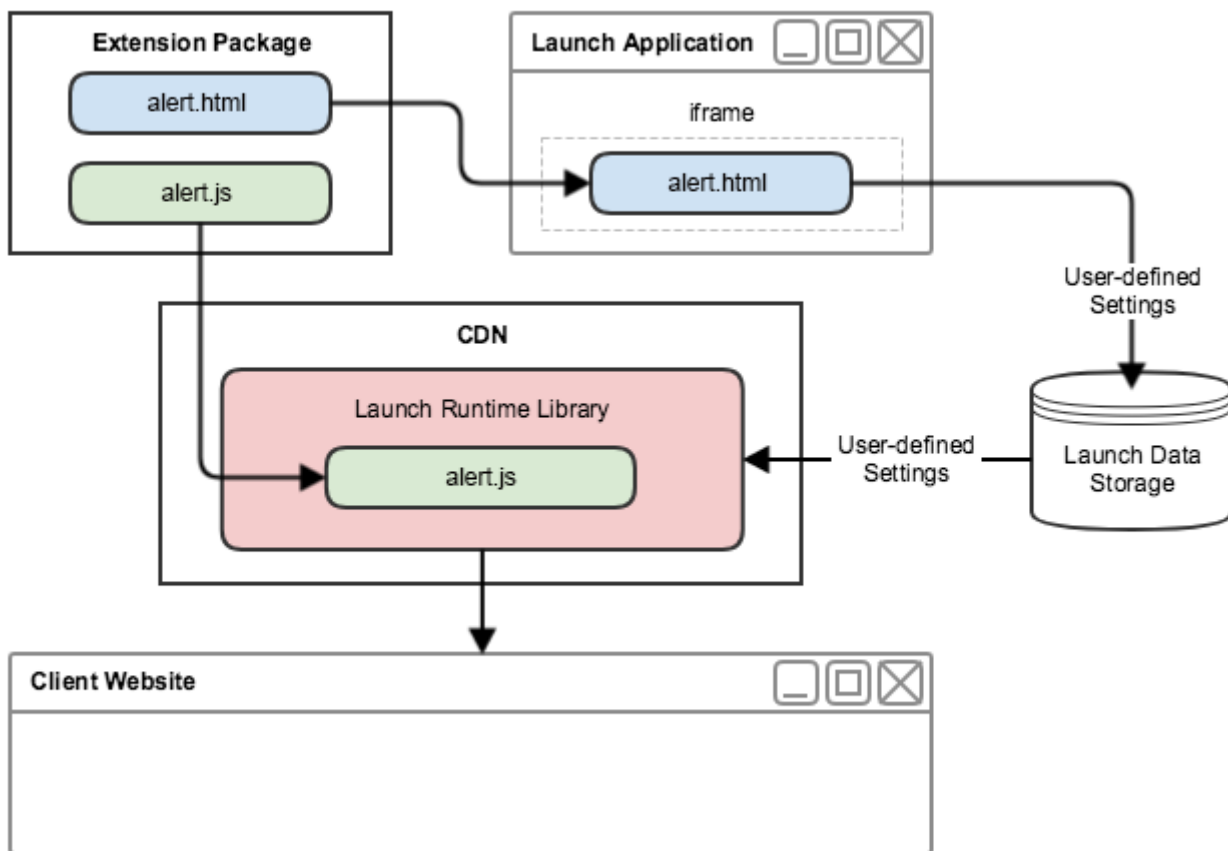
14. Move down to the "I'm done" line and press

The scaffolding tool will create a couple of files and folders. Most notably, it will create two files named after your Action's name. In my case, those are "alert.html" and "alert.js"



Context

For context, this is how Launch, by Adobe uses Extensions.



The "alert.html" file is used by Launch to enable users to provide configuration. The "alert.js" file is what will be delivered into the web site that uses Launch.

Coding

We will provide code for both, the HTML and Javascript, files.

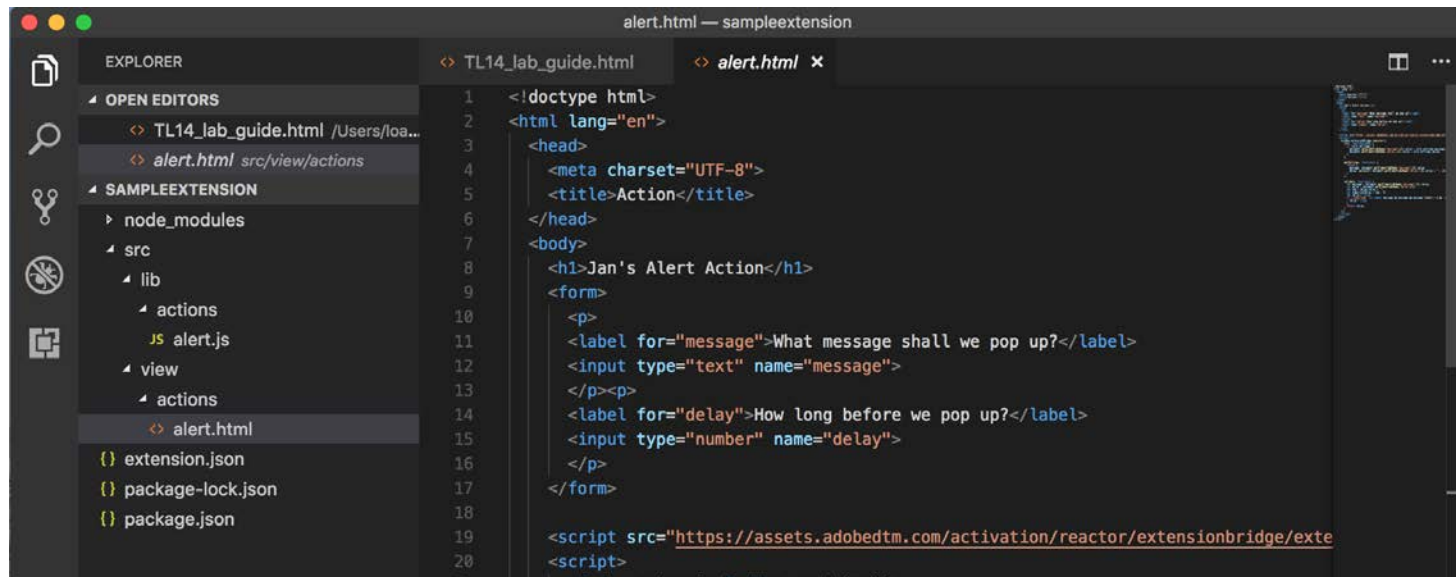
alert.html

When we open the "alert.html" file, it looks pretty empty. We will provide some HTML to create a form. Users of our plugin should be able to input the text for our alert, plus a delay in seconds.

15. Click onto the "alert.html" file to open it in an editor
16. Copy the code below into the file, replacing the "Action Template" line

```
<h1>Jan's Alert Action</h1>
<form>
  <p>
    <label for="message">What message shall we pop up?</label>
    <input type="text" name="message">
  </p>
  <p>
    <label for="delay">How long before we pop up?</label>
    <input type="number" name="delay">
  </p>
</form>
```

This is the result



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Action</title>
6 </head>
7 <body>
8   <h1>Jan's Alert Action</h1>
9   <form>
10     <p>
11       <label for="message">What message shall we pop up?</label>
12       <input type="text" name="message">
13     </p><p>
14       <label for="delay">How long before we pop up?</label>
15       <input type="number" name="delay">
16     </p>
17   </form>
18
19   <script src="https://assets.adobedtm.com/activation/reactor/extensionbridge/exte
20   <script>
```

Underneath, you can see three Javascript functions. Each Extension has to provide those functions, they are how Launch interacts with your configuration page.

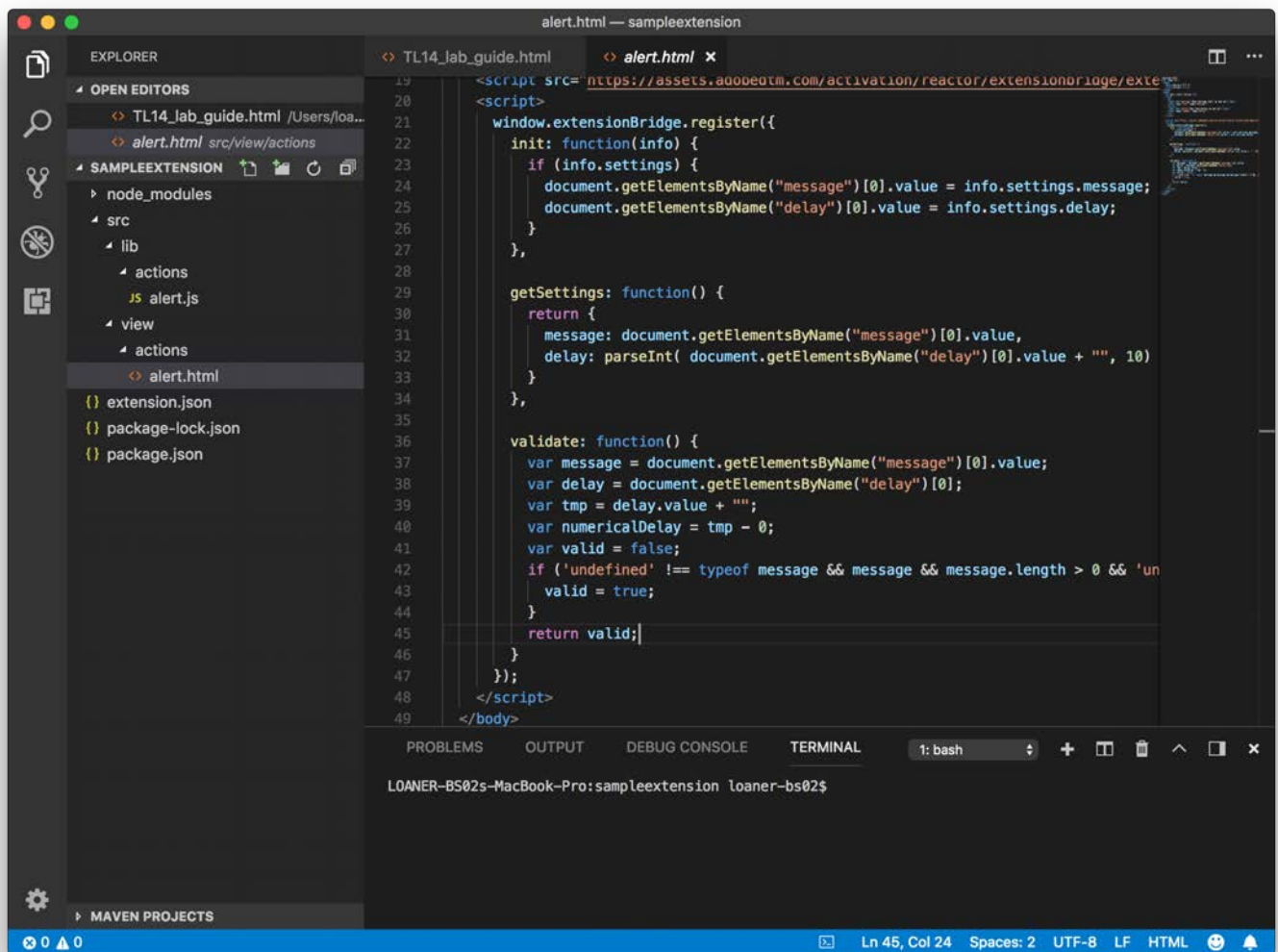
17. Copy the code below, replacing the whole `<script>` block

```

window.extensionBridge.register({
  init: function(info) {
    if (info.settings) {
      document.getElementsByName("message")[0].value = info.settings.m
      document.getElementsByName("delay")[0].value = info.settings.del
    }
  },
  getSettings: function()
  {
    return {
      message: document.getElementsByName("message")[0].value,
      delay: parseInt( document.getElementsByName("delay")[0].value +
    }
  },
  validate: function() {
    var message = document.getElementsByName("message")[0].value;
    var delay = document.getElementsByName("delay")[0];
    var tmp = delay.value + "";
    var numericalDelay = tmp - 0;
    var valid = false;
    if ('undefined' !== typeof message && message && message.length > 0
      valid=true;
    }
    return valid;
  }
});

```

This is the result

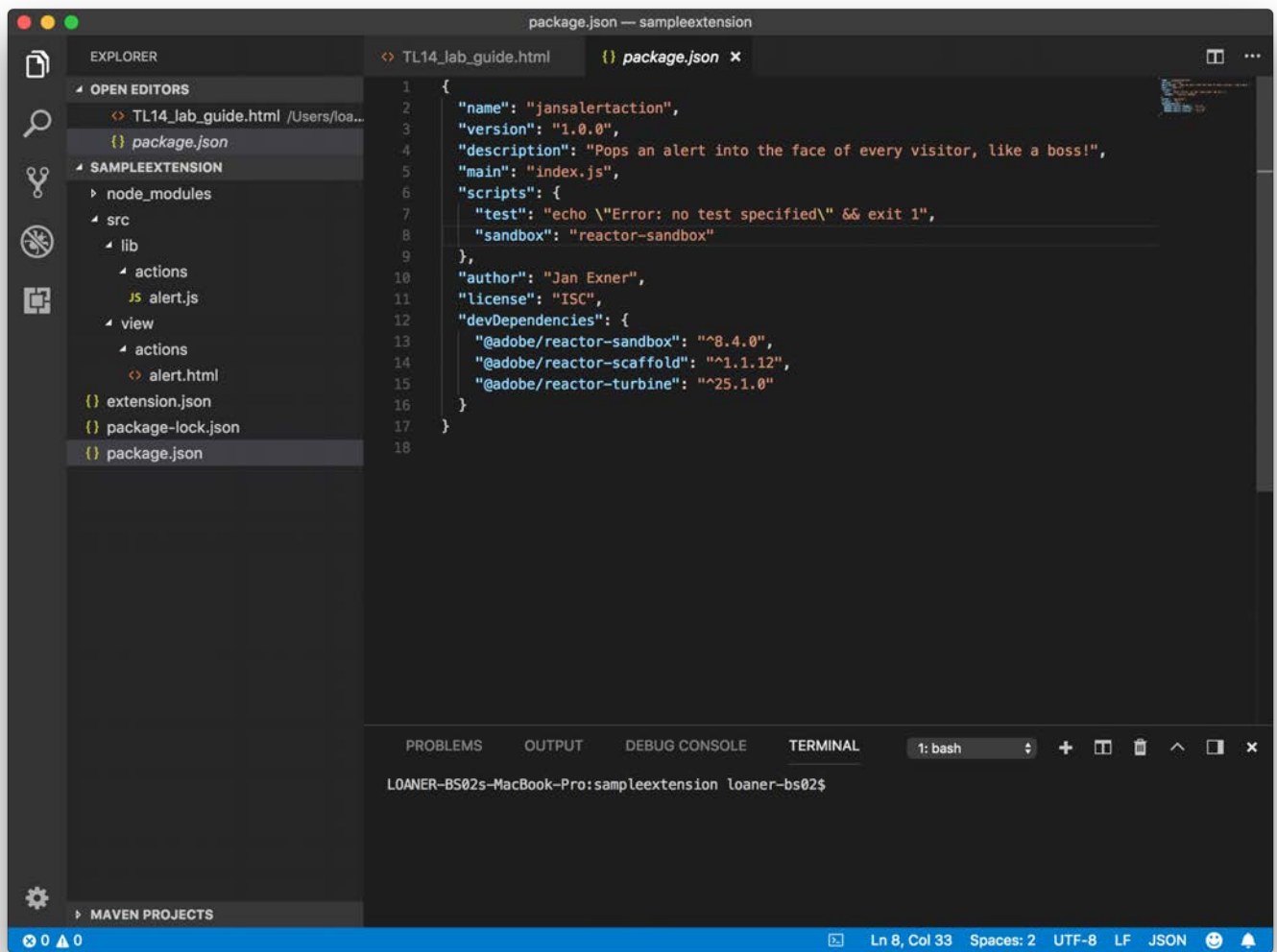


Debugging - the Sandbox

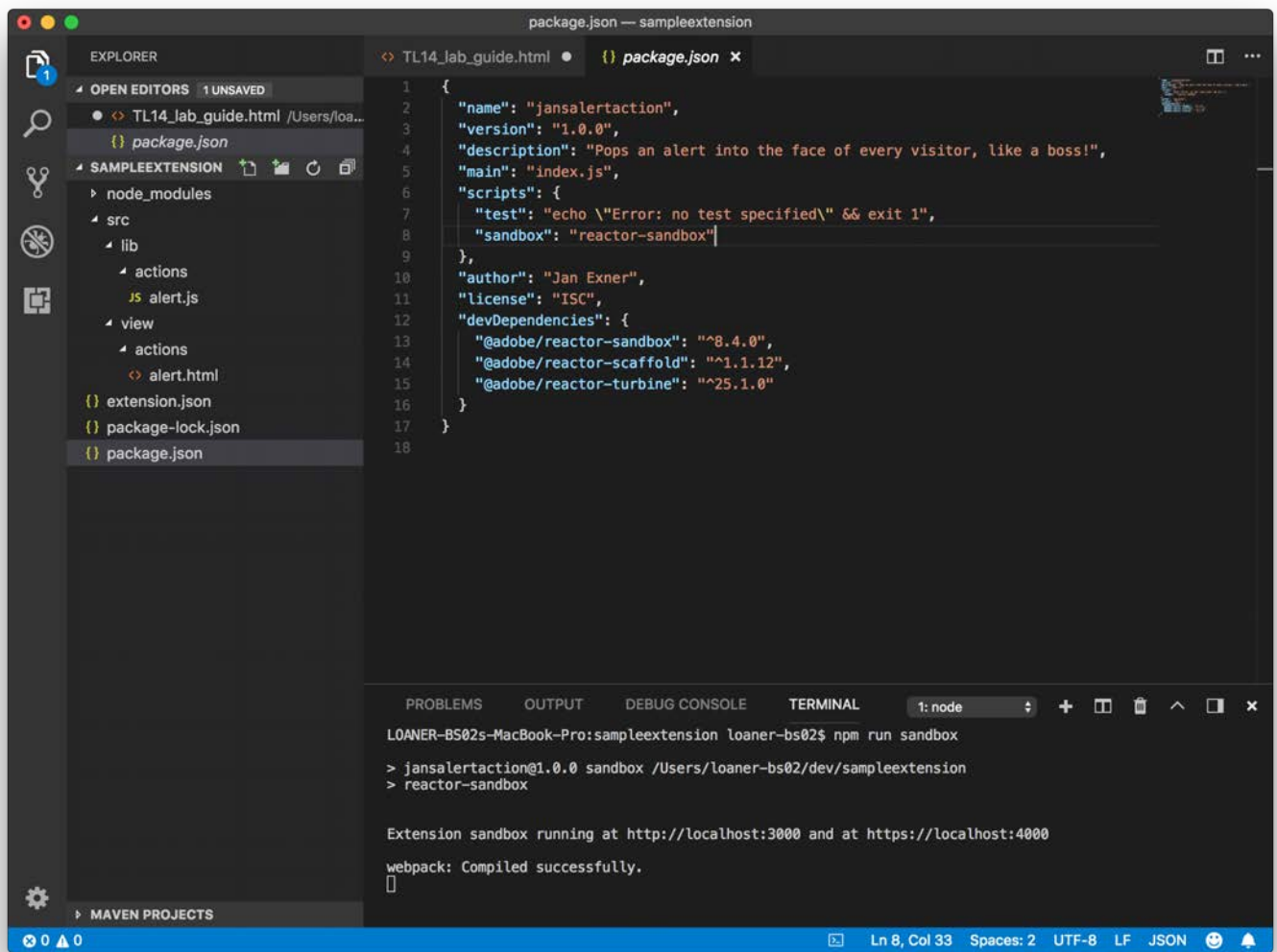
We have to test our code, and there is a tool for that, the "Reactor Sandbox".

18. Type `npm install @adobe/reactor-turbine @adobe/reactor-sandbox --save-dev` into your command line
19. Open the "package.json" file by clicking it.
20. Find the "scripts" block in the file, note it contains one line
21. At the end of the line, type `,`, followed by `enter`
22. paste this into the new line: `"sandbox": "reactor-sandbox"`
23. Type `node_modules/.bin/reactor-sandbox init` into the command line

npm will install some packages, and your "package.json" file should look like this:

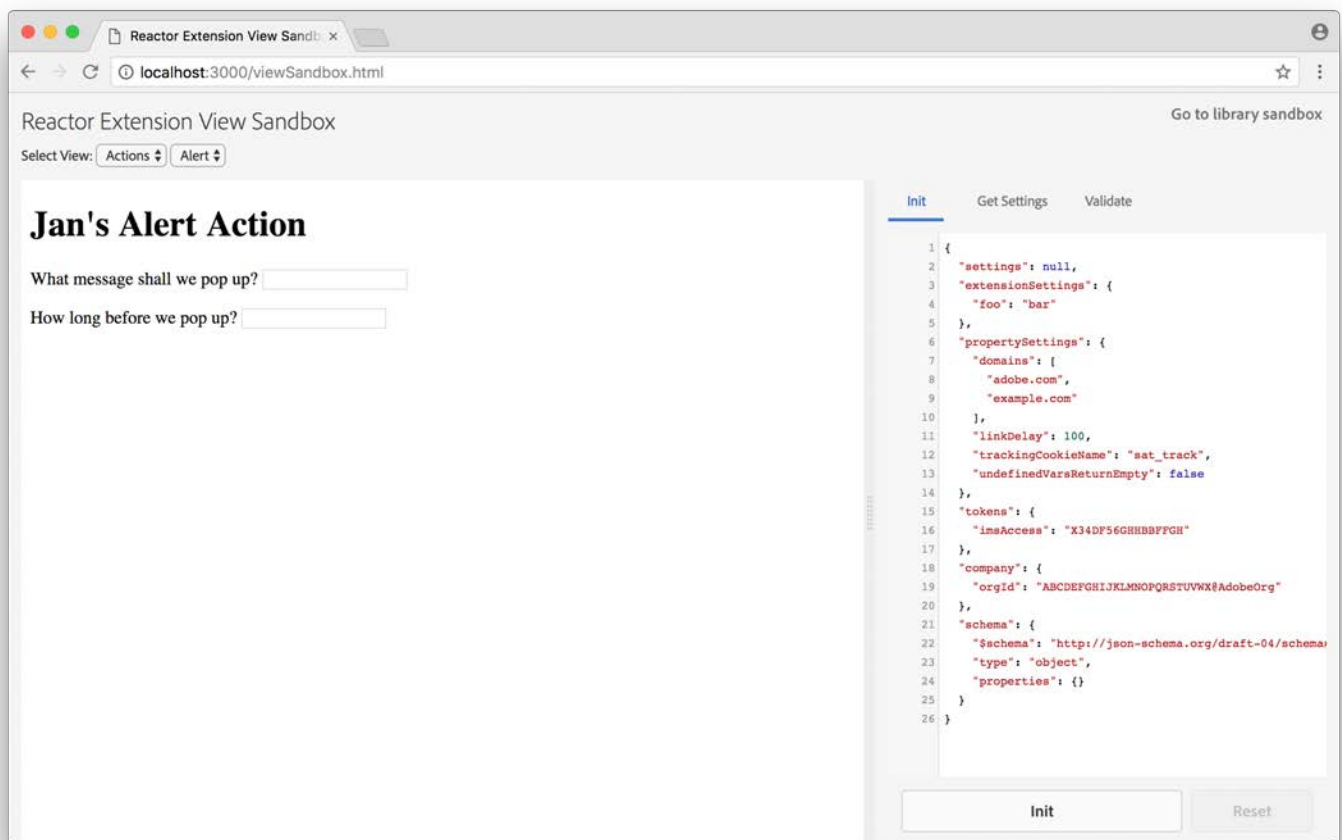


24. Type `npm run sandbox` into the command line.



We just launched a Launch-specific test environment on our machines, which we can see using a browser.

- 25. Open a new tab in your browser
- 26. Type `localhost:3000` into the URL



Moving on to the other file.

Coding, part 2

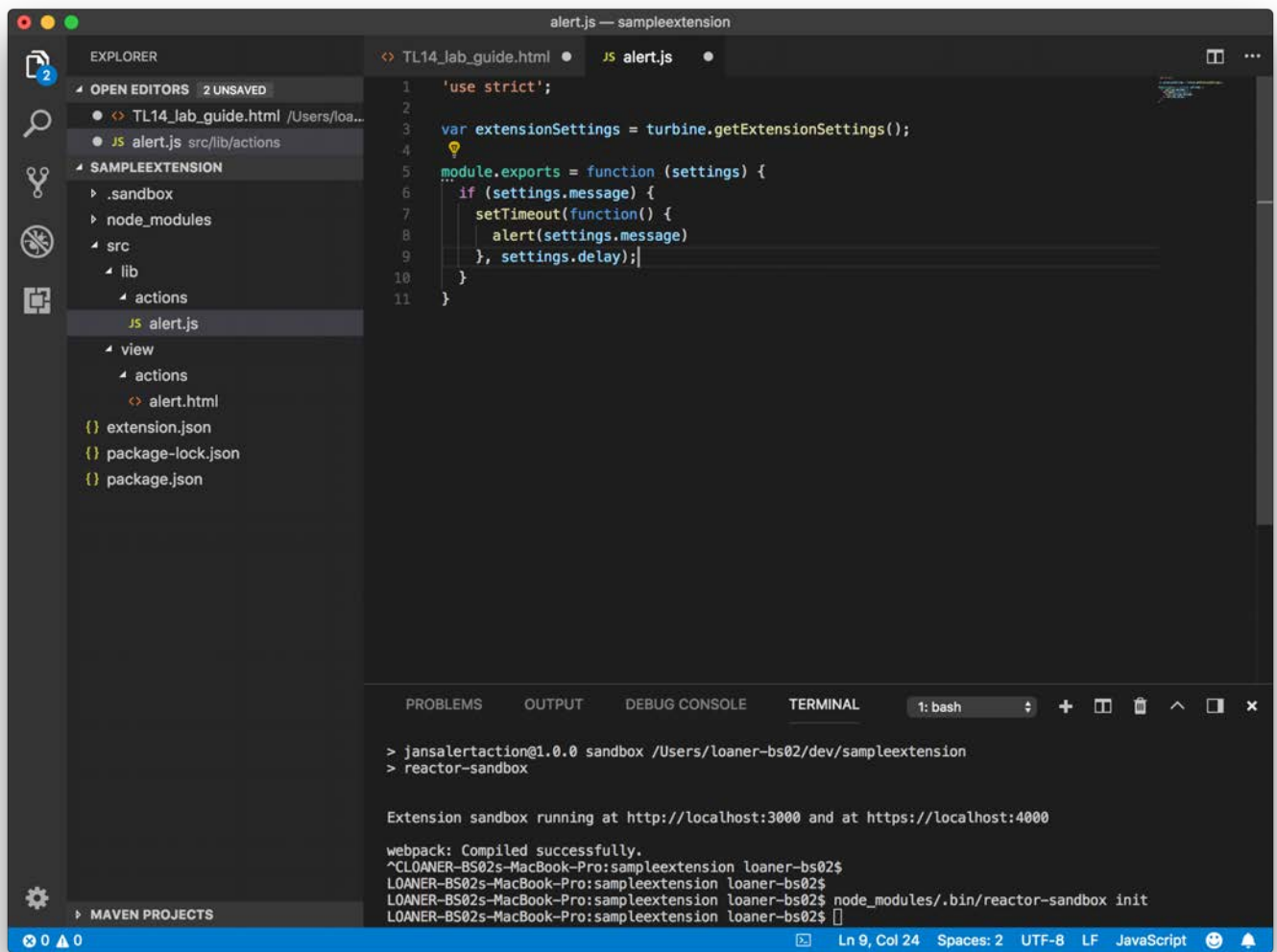
alert.js

The "alert.js" file contains the code that Launch will deliver into the site. This code has to actually make our Action happen.

27. Copy the following code into the "alert.js" file

```
'use strict';
var extensionSettings = turbine.getExtensionSettings();
module.exports = function (settings) {
  if (settings.message) {
    setTimeout(function() {
      alert(settings.message)
    }, settings.delay);
  }
}
```

The file should look like this



The Javascript can be debugged using the Sandbox, too. There is some setup to do, though.

- 28. Open the "container.js" file in the ".sandbox" folder
- 29. Replace the content with the code below

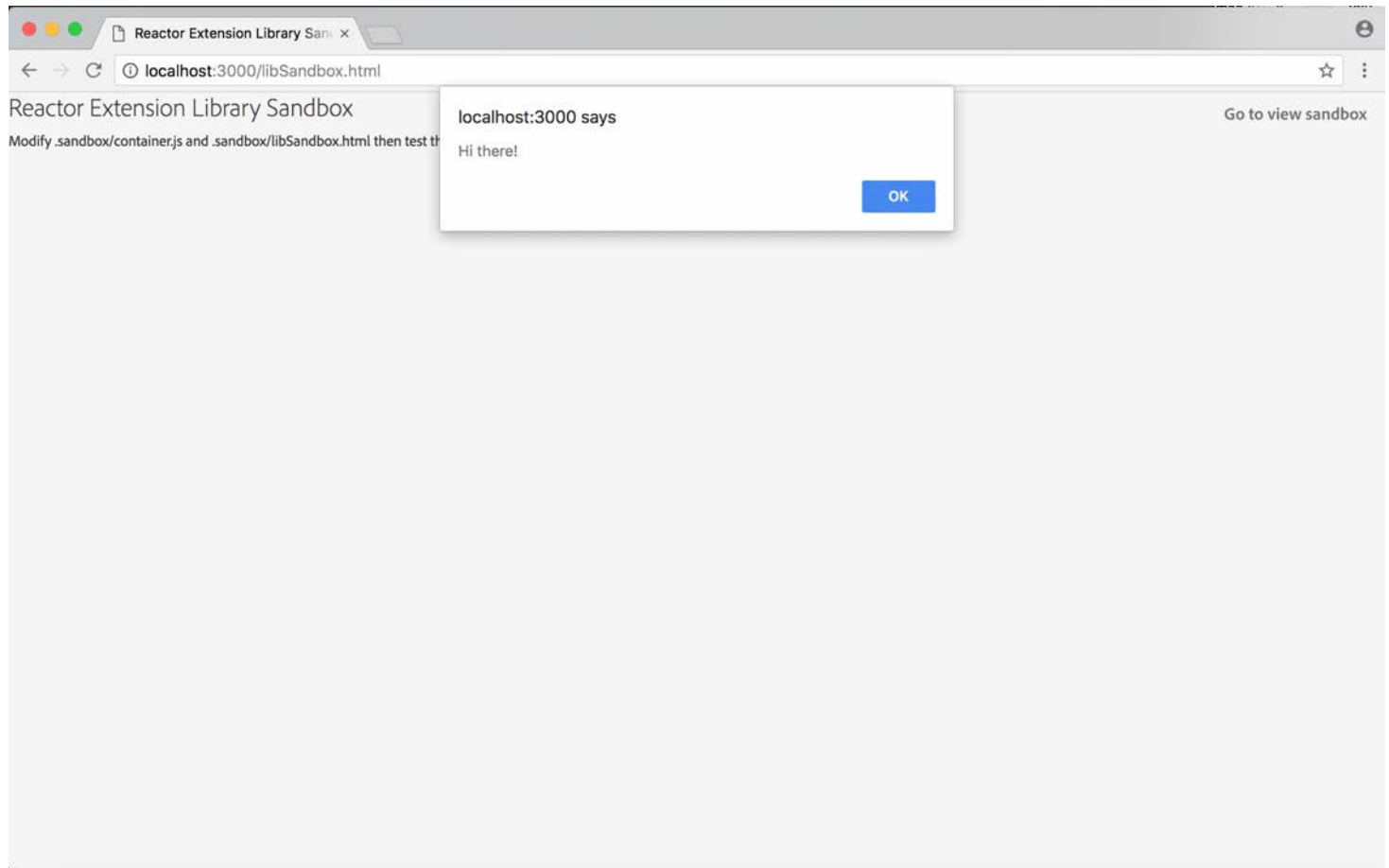
```

'use strict';
module.exports = {
  rules: [
    {
      name: 'Example Rule',
      events: [
        {
          // This is a simple event type provided by the sandbox which triggers
          // as soon as the DTM library is loaded. This event type is provided for
          // convenience in case your extension does not have event types
          modulePath: 'sandbox/pageTop.js',
          settings: {}
        }
      ],
      actions: [
        {
          modulePath: 'jans-alert-action/src/lib/actions/alert.js',
          settings: {
            delay: 10000,
            message: "Hi there!"
          }
        }
      ]
    }
  ],
  extensions: {
    // Set up an extension configuration you would like to test. The top level key is the
    // name of your extension (as defined in your extension.json).
    'jans-alert-action': {
      displayName: "Jan's Alert Action"
    }
  },
  property: {
    name: 'Sandbox property',
    settings: {
      domains: [ 'adobe.com', 'example.com' ],
      linkDelay: 100,
      euCookieName: 'sat_track',
      undefinedVarsReturnEmpty: false
    }
  },
  buildInfo: {
    turbineVersion: "25.1.0",
    turbineBuildDate: "2018-04-12T18:10:34Z",
    buildDate: "2018-03-30T16:27:10Z",
    environment: "development"
  }
}

```

```
};
```

This "container.js" file simulates a Launch Property. It contains one Rule, triggered at Page Top (or Library Load), and one Action (our Action). You can now in the Sandbox click over to the Library Sandbox, and you should see an alert after 10 seconds.



Further Reading

If you want to know more, here are a couple of resources

- [Launch Extension Guides \(https://developer.adobelaunch.com/guides/extensions/\)](https://developer.adobelaunch.com/guides/extensions/)
- [Launch-by-Adobe \(https://medium.com/launch-by-adobe\)](https://medium.com/launch-by-adobe) blog (technical)
- [Miniseries on Launch Extensions \(https://webanalyticsfordevelopers.com/2018/01/09/launch-extensions/\)](https://webanalyticsfordevelopers.com/2018/01/09/launch-extensions/) on [webanalyticsfordevelopers.com \(https://webanalyticsfordevelopers.com\)](https://webanalyticsfordevelopers.com)

When you are done with your Extension, and you are considering making it public, speak to Jeff Chasin.