Skill AP
APSSDC

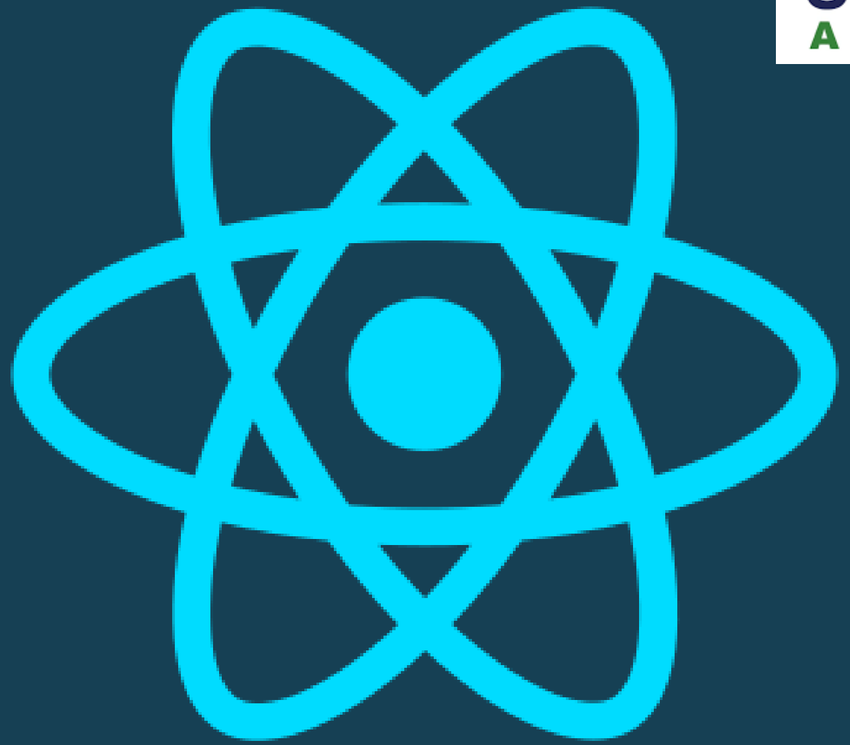
HOW TO CREATE REACT JS PROJECT IN VSCODE

DAY 4 - WEB DEVELOPMENT MASTERCLASS

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

REACT JS PROJECTS

React is a popular JavaScript library developed by Facebook for building user interfaces. The Visual Studio Code editor supports React.js IntelliSense and code navigation out of the box.



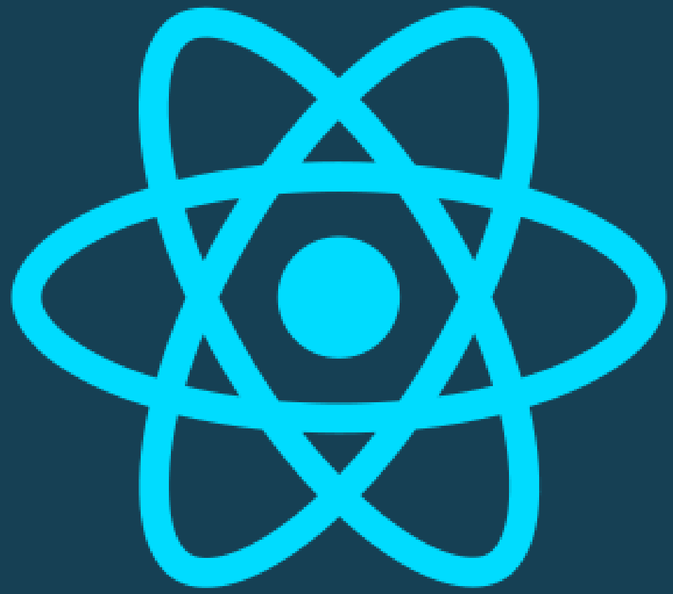
React JS

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

30 DAYS WEB DEVELOPMENT MASTERCLASS

REACT JS PROJECTS

We'll be using the create-react-app generator for this. To use the generator as well as run the React application server, you'll need Node.js JavaScript runtime and npm (Node.js package manager) installed. NPM is included with Node.js which you can download and install from Node.js downloads.



Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

REACT JS PROJECTS

Tip: To test that you have Node.js and npm correctly installed on your machine, you can type node --version and npm --version in a terminal or command prompt.

Skill AP
APSSDC

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

REACT JS PROJECTS

You can now create a new React application by typing:

```
npx create-react-app my-app
```

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

REACT JS PROJECTS

where my-app is the name of the folder for your application. This may take a few minutes to create the React application and install its dependencies.

Note: If you've previously installed create-react-app globally via npm install -g create-react-app, we recommend you uninstall the package using npm uninstall -g create-react-app to ensure that npx always uses the latest version.

Skill AP
APSSDC

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

REACT JS PROJECTS

Let's quickly run our React application by navigating to the new folder and typing npm start to start the web server and open the application in a browser:

```
Ncd my-app
npm start
```

REACT JS PROJECTS

You should see the React logo and a link to "Learn React" on http://localhost:3000 in your browser. We'll leave the web server running while we look at the application with VS Code.

To open your React application in VS Code, open another terminal or command prompt window, navigate to the my-app folder and type code .:

```
cd my-app
code .
```

Skill AP
APSSDC

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

REACT JS PROJECTS

1.MARKDOWN PREVIEW
2.Syntax highlighting and bracket matching
3.IntelliSense
4.Go to Definition, Peek definition
5.Hello World
6.Debugging React
7.Set a breakpoint
8.Configure the debugger
9.Live editing and debugging
10.Linting

Skill AP
APSSDC

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

MARKDOWN PREVIEW

In the File Explorer, one file you'll see is the application README.md Markdown file. This has lots of great information about the application and React in general. A nice way to review the README is by using the VS Code Markdown Preview. You can open the preview in either the current editor group (Markdown: Open Preview Ctrl+Shift+V) or in a new editor group to the side (Markdown: Open Preview to the Side Ctrl+K V). You'll get nice formatting, hyperlink navigation to headers, and syntax highlighting in code blocks.

Skill AP
APSSDC

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

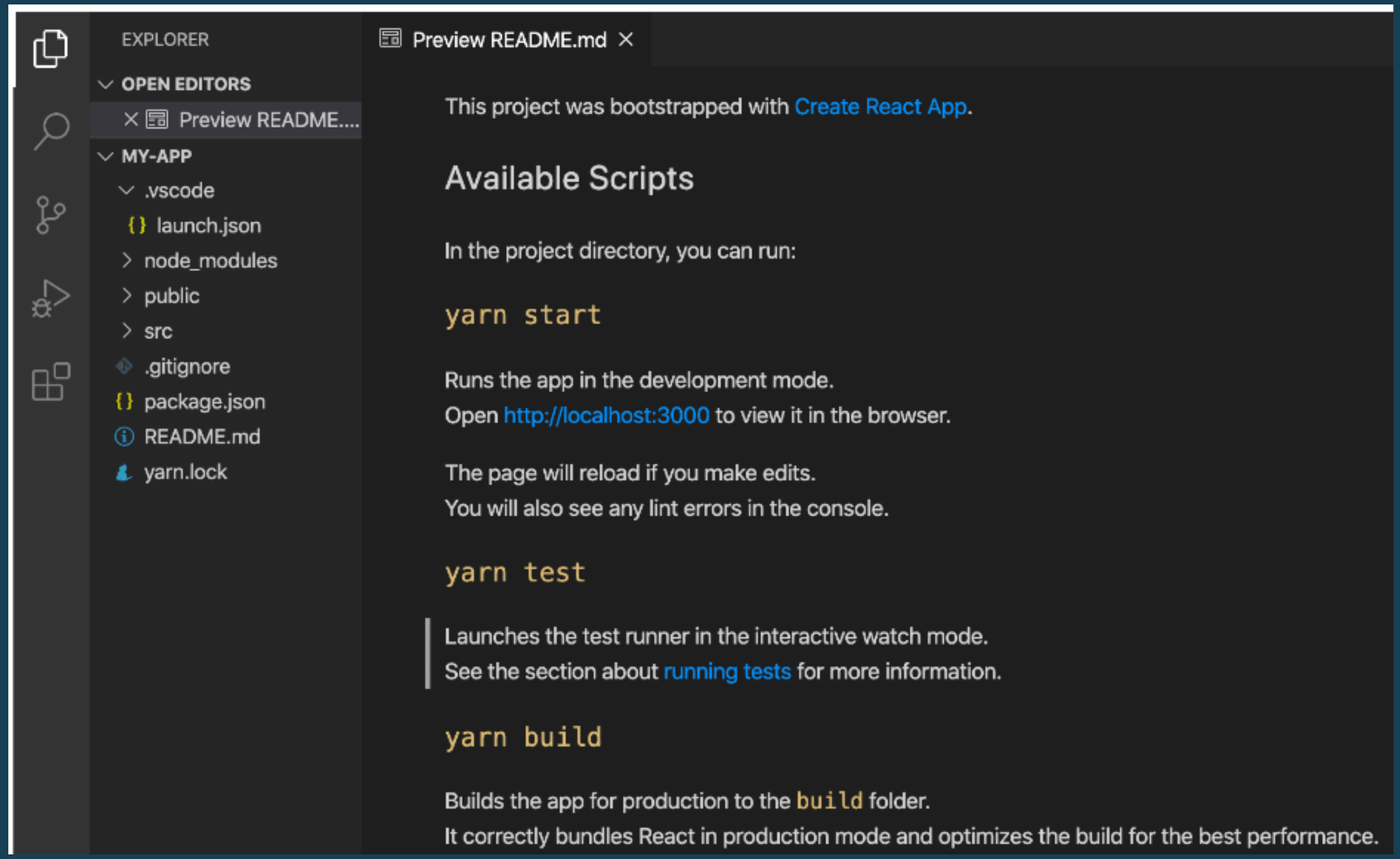
MARKDOWN PREVIEW

In the File Explorer, one file you'll see is the application README.md Markdown file. This has lots of great information about the application and React in general. A nice way to review the README is by using the VS Code Markdown Preview. You can open the preview in either the current editor group (Markdown: Open Preview Ctrl+Shift+V) or in a new editor group to the side (Markdown: Open Preview to the Side Ctrl+K V). You'll get nice formatting, hyperlink navigation to headers, and syntax highlighting in code blocks.

Skill AP
APSSDC

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

MARKDOWN PREVIEW

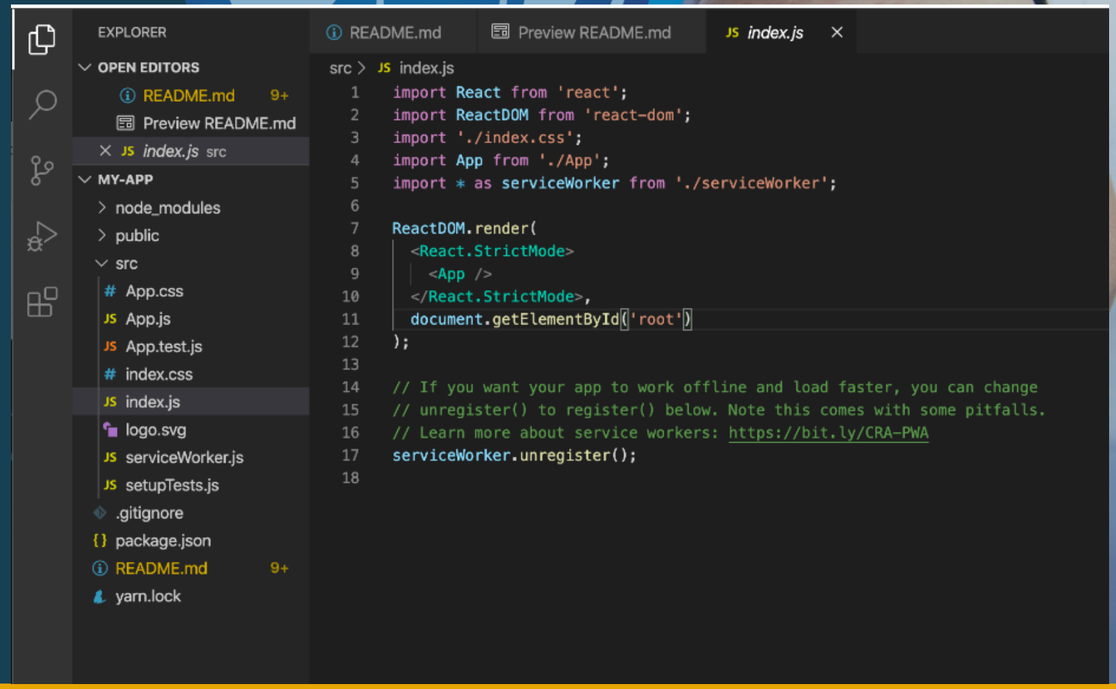


Skill AP
APSSDC

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

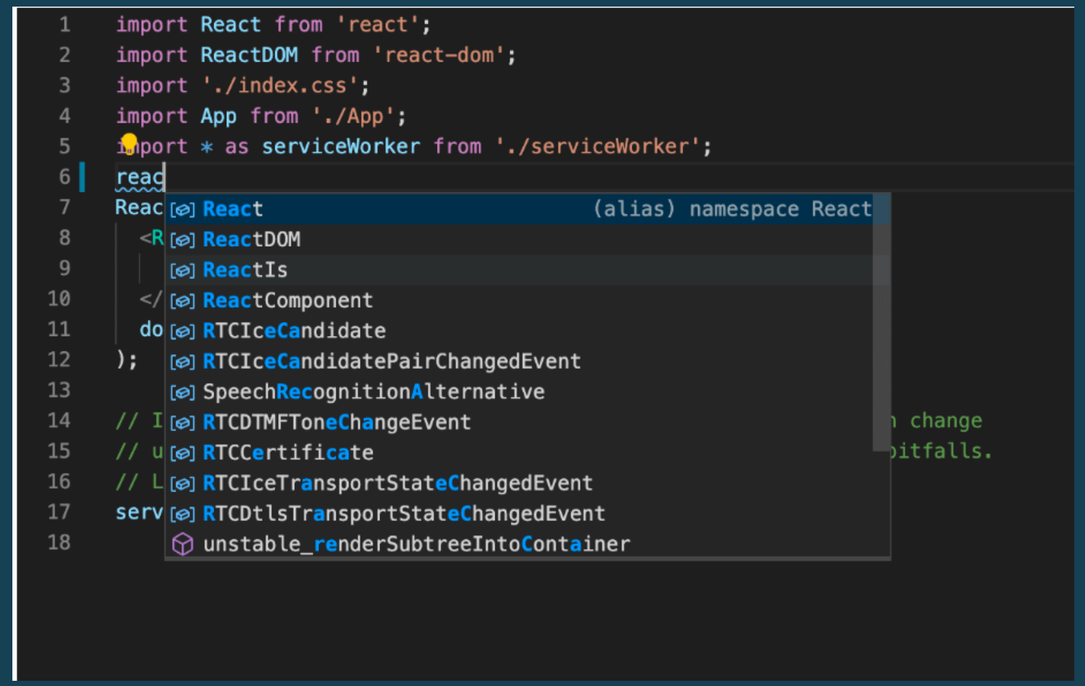
SYNTAX HIGHLIGHTING AND BRACKET MATCHING

Now expand the src folder and select the index.js file. You'll notice that VS Code has syntax highlighting for the various source code elements and, if you put the cursor on a parenthesis, the matching bracket is also selected.



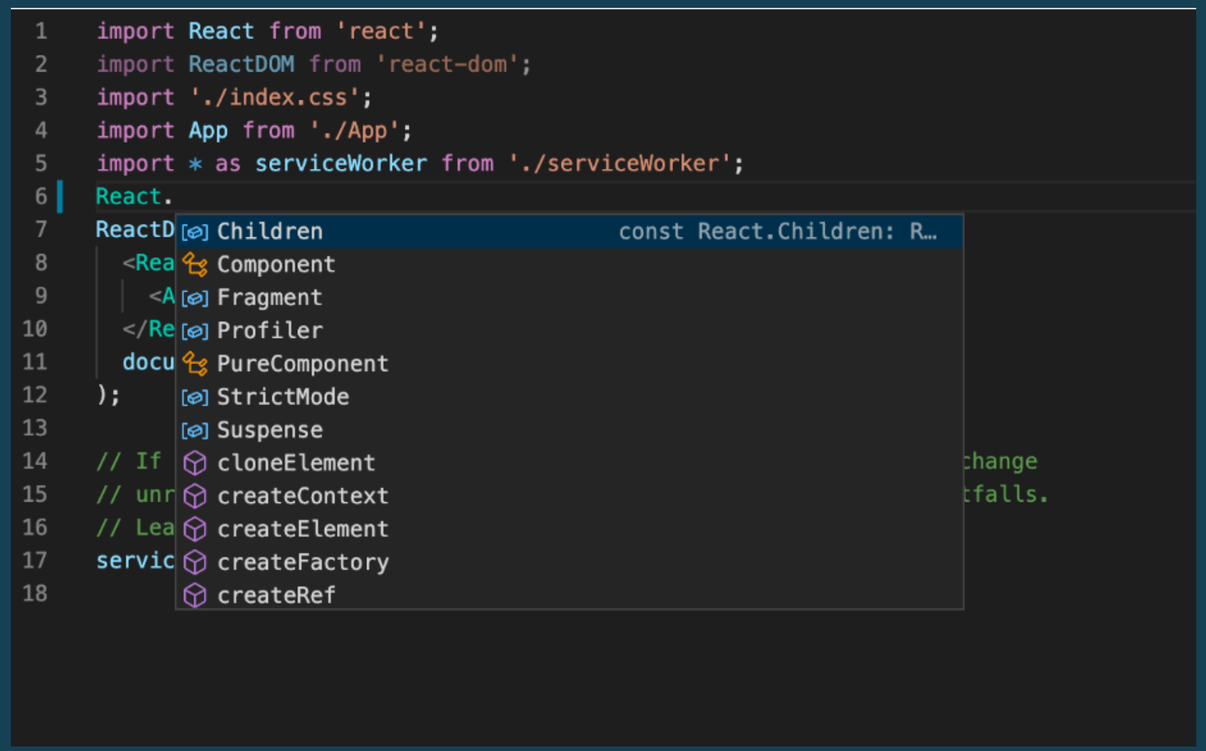
INTELLISENSE

As you start typing in index.js, you'll see smart suggestions or completions.



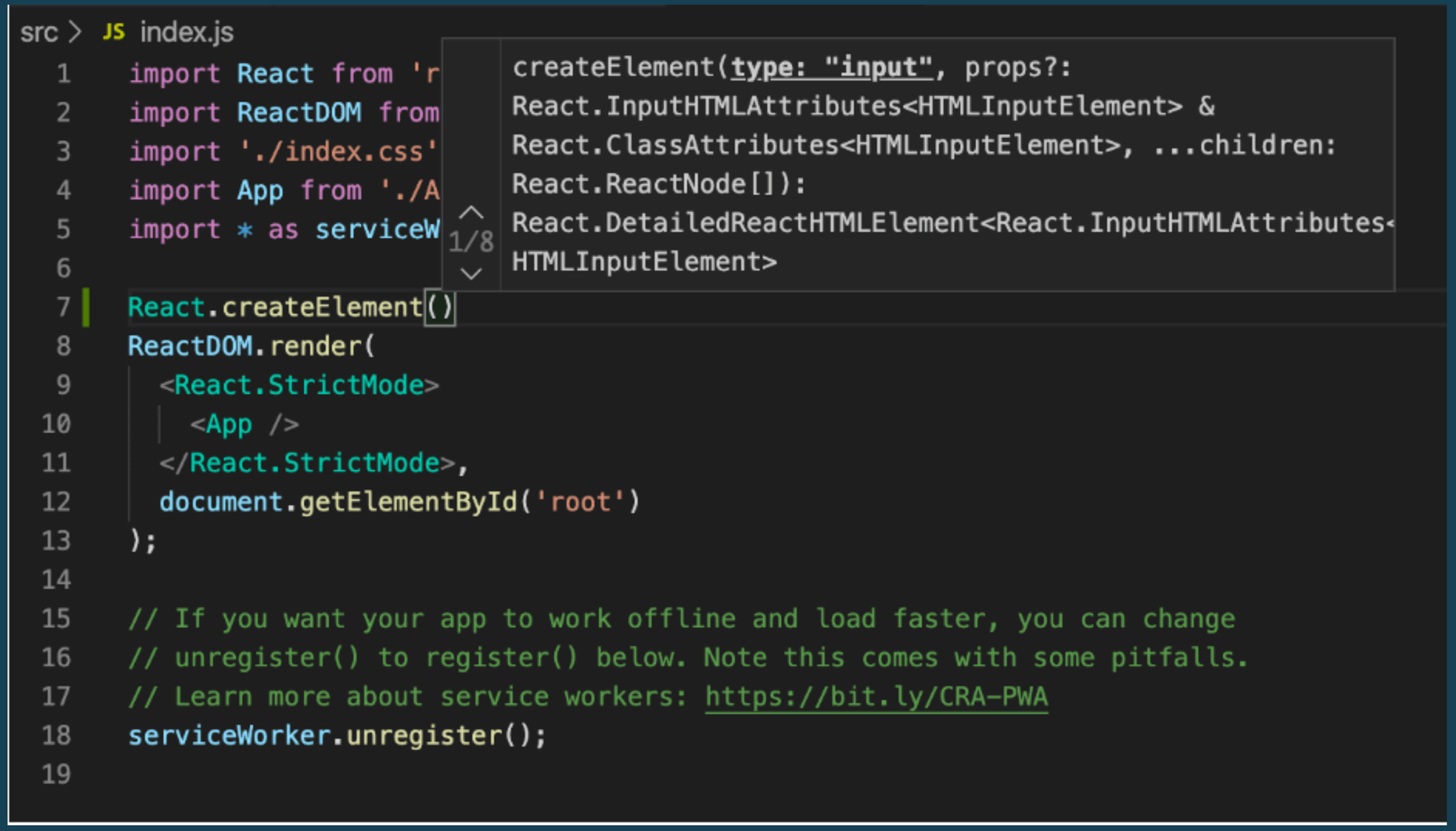
After you select a suggestion and type ., you see the types a methods on the object through IntelliSense.

ATA



VS Code uses the TypeScript language service for its JavaScript code intelligence and it has a feature called Automatic Type Acquisition (ATA). ATA pulls down the npm Type Declaration files (*.d.ts) for the npm modules referenced in the package.json.

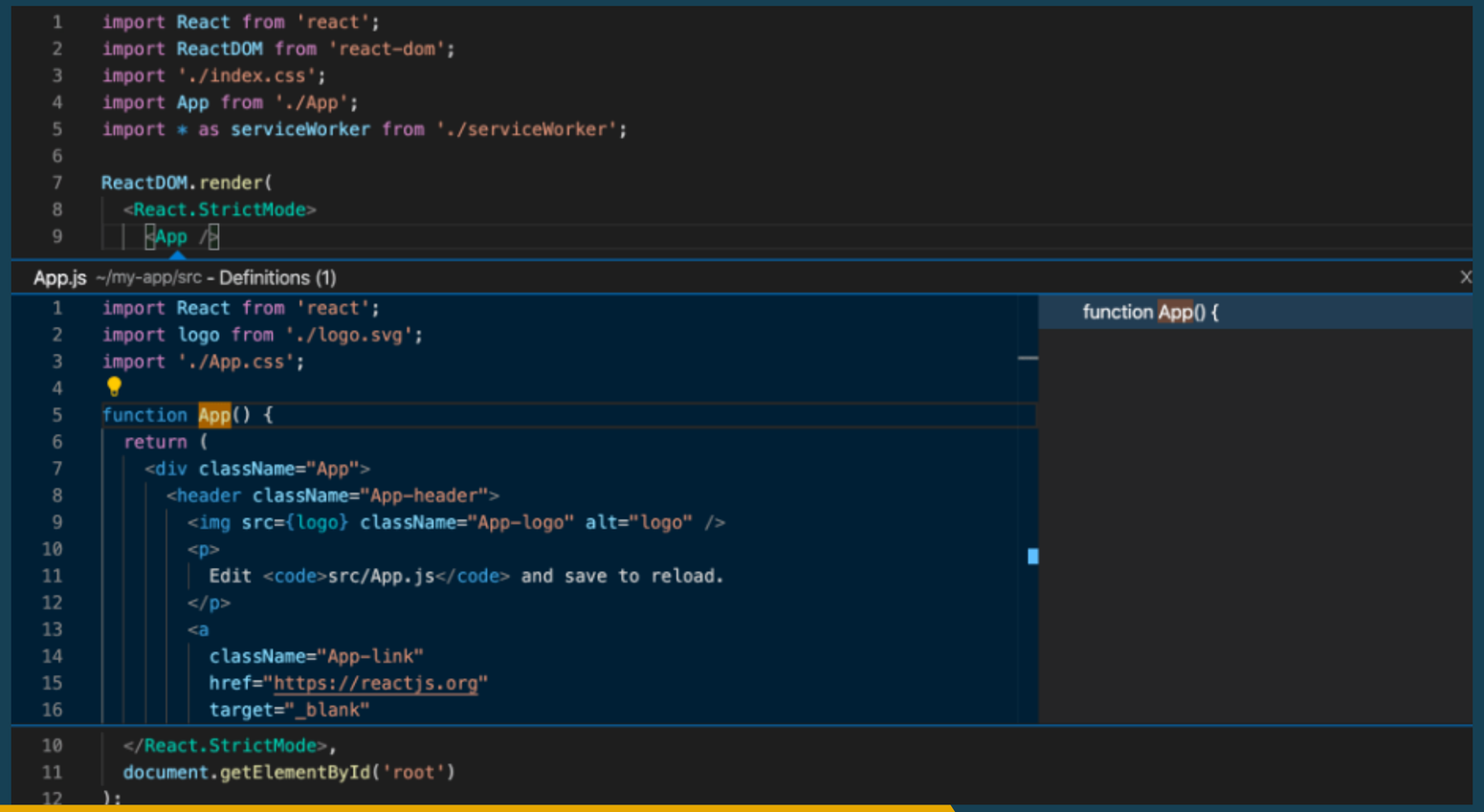
If you select a method, you'll also get parameter help:



GO TO , PEEK

Go to Definition, Peek definition

rough the TypeScript language service, VS Code can also provide type definition information in the editor through Go to Definition (F12) or Peek Definition (Alt+F12). Put the cursor over the App, right click and select Peek Definition. A Peek window will open showing the App definition from App.js.



HELLO WORLD !

Hello World Let's update the sample application to "Hello World!". Create a new H1 header with "Hello, world!" and replace the <App /> tag in ReactDOM.render with element.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

var element = React.createElement('h1', { className: 'greeting' }, 'Hello, world!');
ReactDOM.render(element, document.getElementById('root'));

reportWebVitals();
```

HELLO WORLD !

Once you save the index.js file, the running instance of the server will update the web page and you'll see
"Hello World!"
when you refresh your browser.

HELLO WORLD !

Tip: VS Code supports Auto Save, which by default saves your files after a delay.
Check the Auto Save option in the File menu to turn on Auto Save or directly configure the files.autoSave user setting.

DEBUGGING REACT

To debug the client side React code,
we'll use the built-in JavaScript debugger.

SET A BREAKPOINT

To set a breakpoint in index.js, click on the gutter to the left of the line numbers. This will set a breakpoint which will be visible as a red circle.

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import './index.css';
4   import App from './App';
5   import * as serviceWorker from './serviceWorker';
6
7   var element = React.createElement('h1', { className: 'greeting'}, 'Hello, world!');
8   ReactDOM.render(element, document.getElementById('root'));
9
10  // If you want your app to work offline and load faster, you can change
11  // unregister() to register() below. Note this comes with some pitfalls.
12  // Learn more about service workers: https://bit.ly/CRA-PWA
13  serviceWorker.unregister();
```

CONFIGURE THE DEBUGGER

We need to initially configure the debugger. To do so, go to the Run and Debug view (Ctrl+Shift+D) and select the create a launch.json file link to create a launch.json debugger configuration file. Choose Web App (Edge) from the Select debugger dropdown list. This will create a launch.json file in a new .vscode folder in your project which includes a configuration to launch the website.

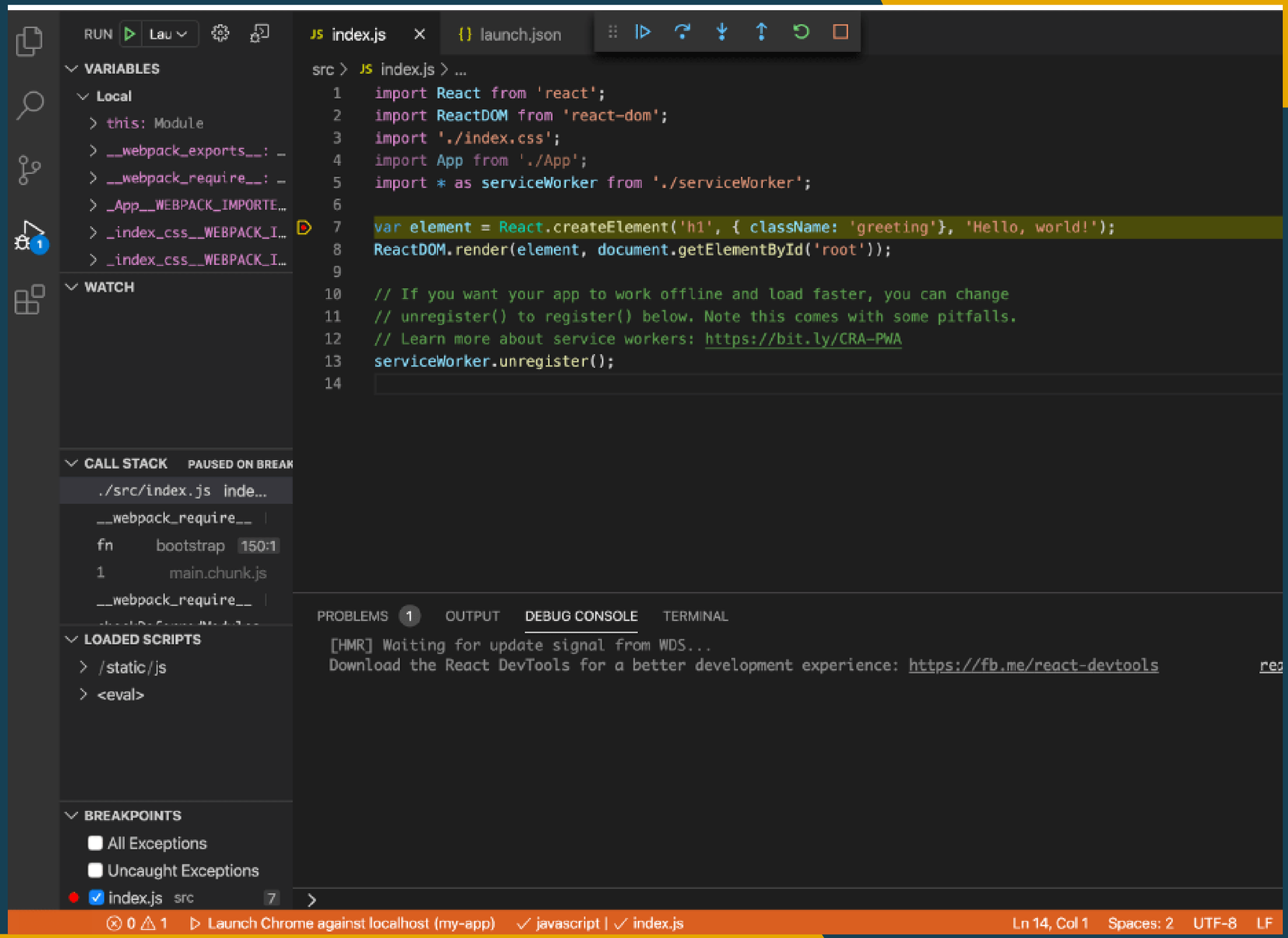
We need to make one change for our example: change the port of the url from 8080 to 3000. Your launch.json should look like this:

CONFIGURE THE DEBUGGER

```
{
"version": "0.2.0",
"configurations": [
{
"type": "msedge",
"request": "launch",
"name": "Launch Edge against localhost",
"url": "http://localhost:3000",
"webRoot": "${workspaceFolder}"
}
]
}
```

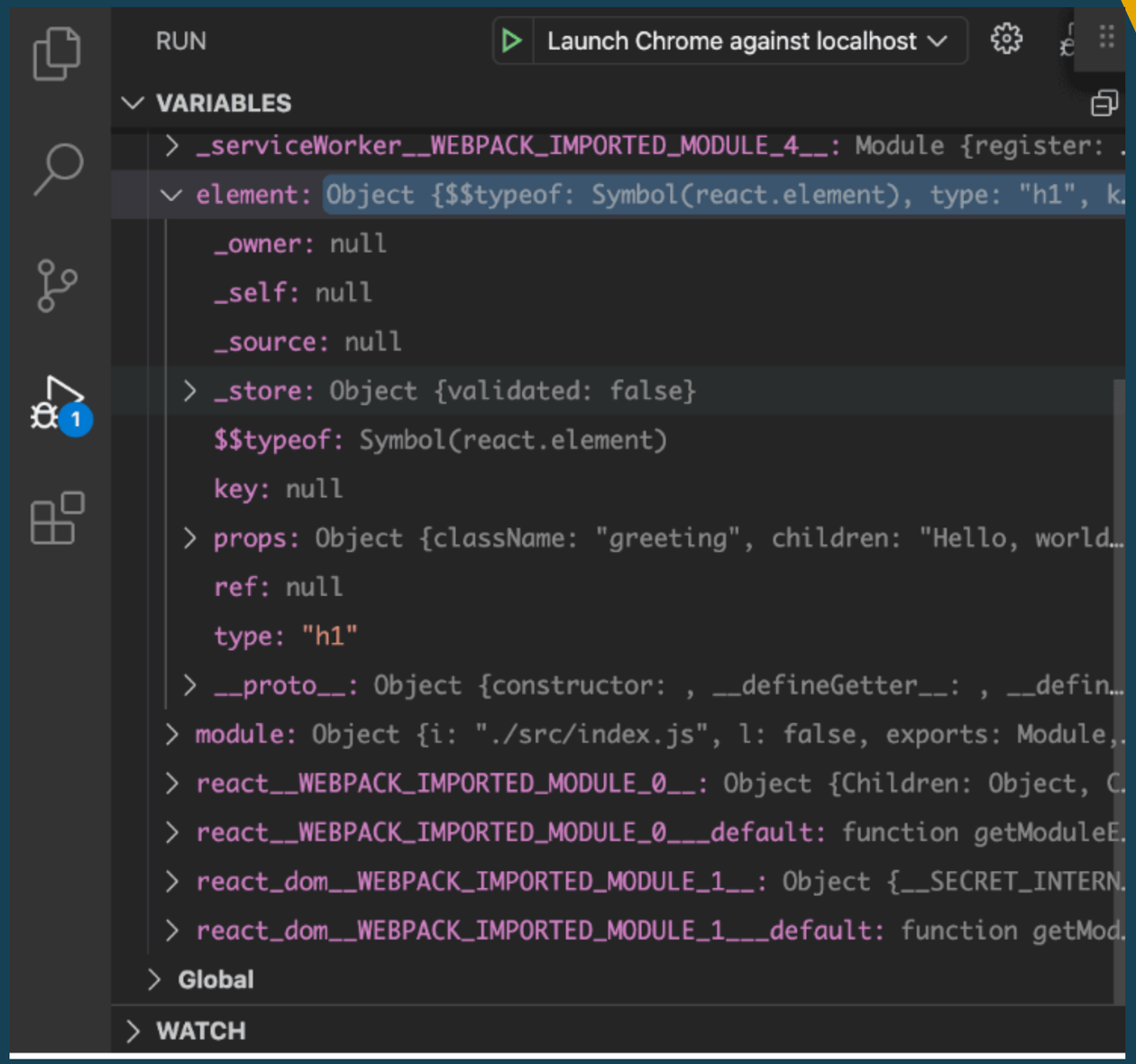
CONFIGURE THE DEBUGGER

Ensure that your development server is running (npm start). Then press F5 or the green arrow to launch the debugger and open a new browser instance. The source code where the breakpoint is set runs on startup before the debugger was attached, so we won't hit the breakpoint until we refresh the web page. Refresh the page and you should hit your breakpoint.


RUN
Lau
VARIABLES
Local
this: Module
__webpack_exports__: …
__webpack_require__: …
_App__WEBPACK_IMPORTE…
_index_css__WEBPACK_I…
_index_css__WEBPACK_I…
WATCH
CALL STACK
PAUSED ON BREAK
./src/index.js inde…
__webpack_require__
fn bootstrap 150:1
1 main.chunk.js
__webpack_require__
LOADED SCRIPTS
/static/js
<eval>
BREAKPOINTS
All Exceptions
Uncaught Exceptions
index.js src 7
index.js
launch.json
src > index.js > …
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import * as serviceWorker from './serviceWorker';
6
7 var element = React.createElement('h1', { className: 'greeting'}, 'Hello, world!');
8 ReactDOM.render(element, document.getElementById('root'));
9
10 // If you want your app to work offline and load faster, you can change
11 // unregister() to register() below. Note this comes with some pitfalls.
12 // Learn more about service workers: https://bit.ly/CRA-PWA
13 serviceWorker.unregister();
14
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
[HMR] Waiting for update signal from WDS...
Download the React DevTools for a better development experience: https://fb.me/react-devtools
⊗ 0 ⚠ 1
Launch Chrome against localhost (my-app)
javascript | index.js
Ln 14, Col 1
Spaces: 2
UTF-8
LF

CONFIGURE THE DEBUGGER

You can step through your source code (F10), inspect variables such as element, and see the call stack of the client side React application.


RUN
Launch Chrome against localhost
VARIABLES
_serviceWorker__WEBPACK_IMPORTED_MODULE_4__: Module {register: .
element: Object {$$typeof: Symbol(react.element), type: "h1", k.
_owner: null
_self: null
_source: null
_store: Object {validated: false}
$$typeof: Symbol(react.element)
key: null
props: Object {className: "greeting", children: "Hello, world…
ref: null
type: "h1"
__proto__: Object {constructor: , __defineGetter__: , __defin…
module: Object {i: "./src/index.js", l: false, exports: Module,.
react__WEBPACK_IMPORTED_MODULE_0__: Object {Children: Object, C.
react__WEBPACK_IMPORTED_MODULE_0___default: function getModuleE.
react_dom__WEBPACK_IMPORTED_MODULE_1__: Object {__SECRET_INTERN.
react_dom__WEBPACK_IMPORTED_MODULE_1___default: function getMod.
Global
WATCH

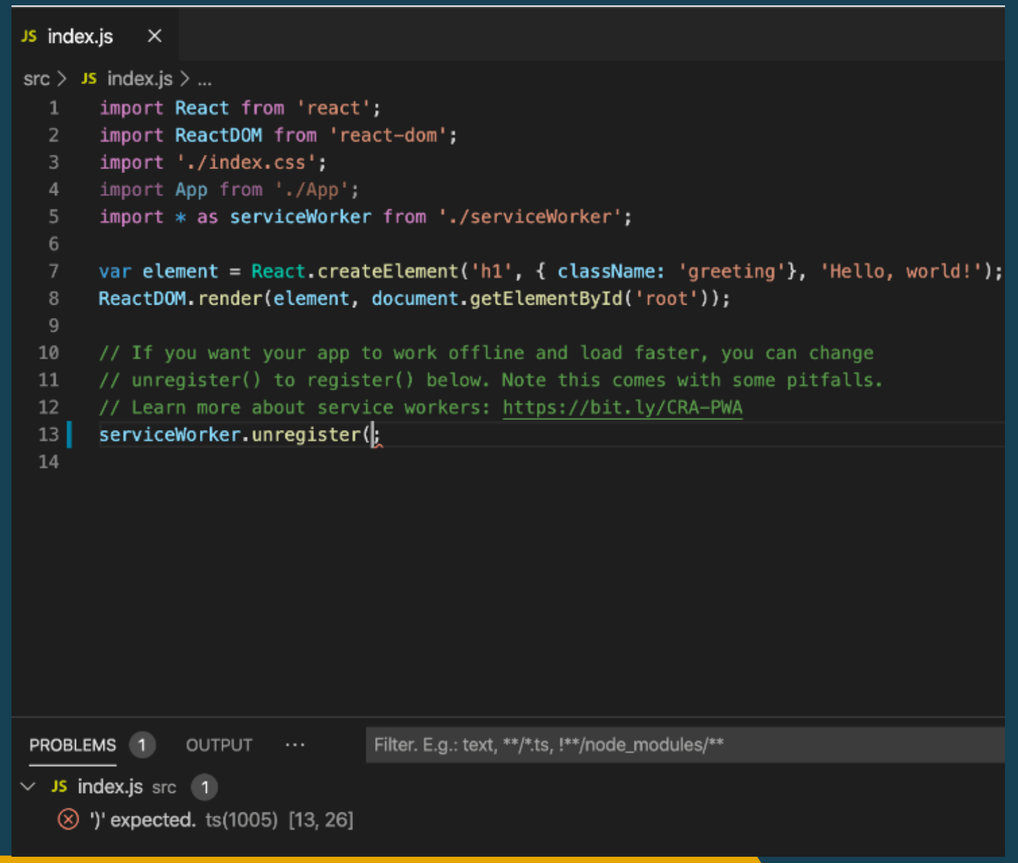
Live editing and debugging

If you are using webpack together with your React app, you can have a more efficient workflow by taking advantage of webpack's HMR mechanism which enables you to have live editing and debugging directly from VS Code

Linting

Linters analyze your source code and can warn you about potential problems before you run your application. The JavaScript language services included with VS Code has syntax error checking support by default, which you can see in action in the Problems panel (View > Problems Ctrl+Shift+M).

Try making a small error in your React source code and you'll see a red squiggle and an error in the Problems panel.



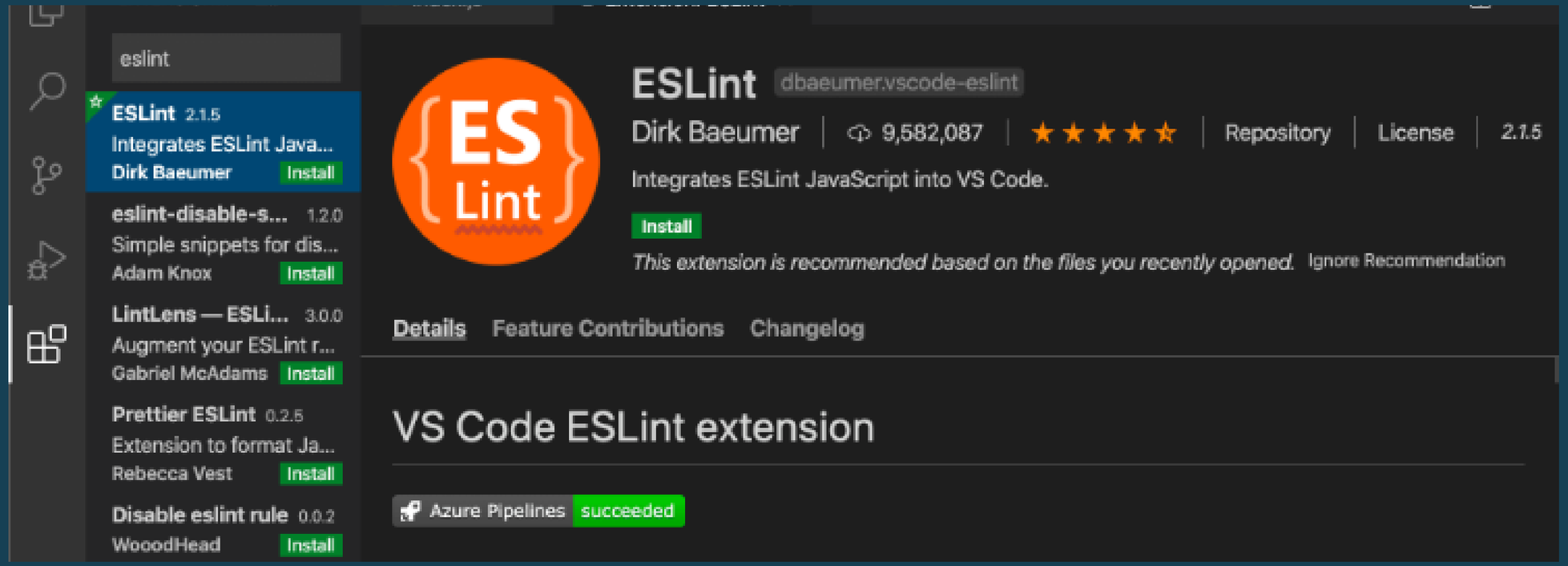
Linters

Linters can provide more sophisticated analysis, enforcing coding conventions and detecting anti-patterns. A popular JavaScript linter is ESLint. ESLint, when combined with the ESLint VS Code extension, provides a great in-product linting experience.

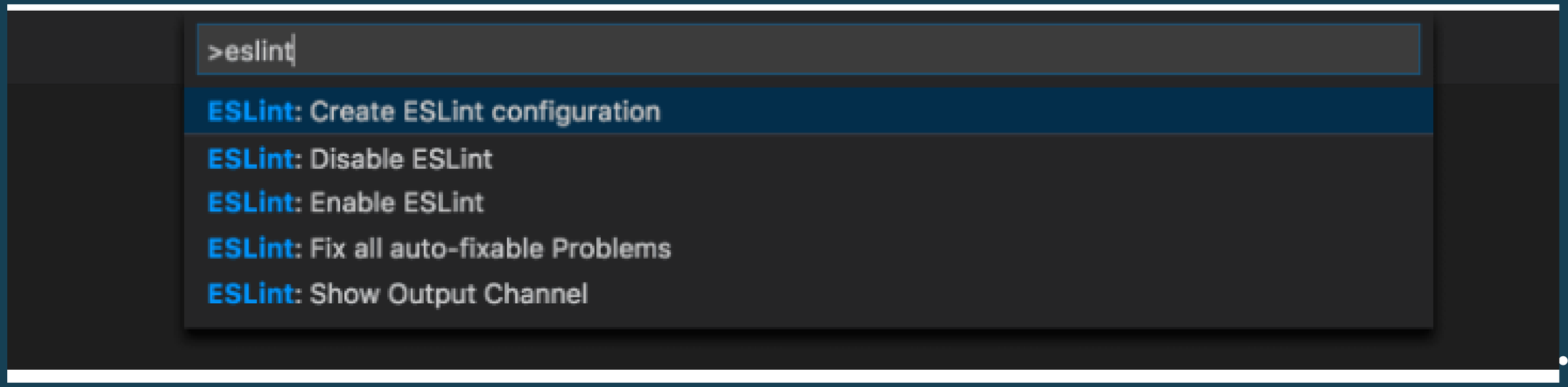
First, install the ESLint command-line tool:

npm install -g eslint

Then install the ESLint extension by going to the Extensions view and typing 'eslint'.



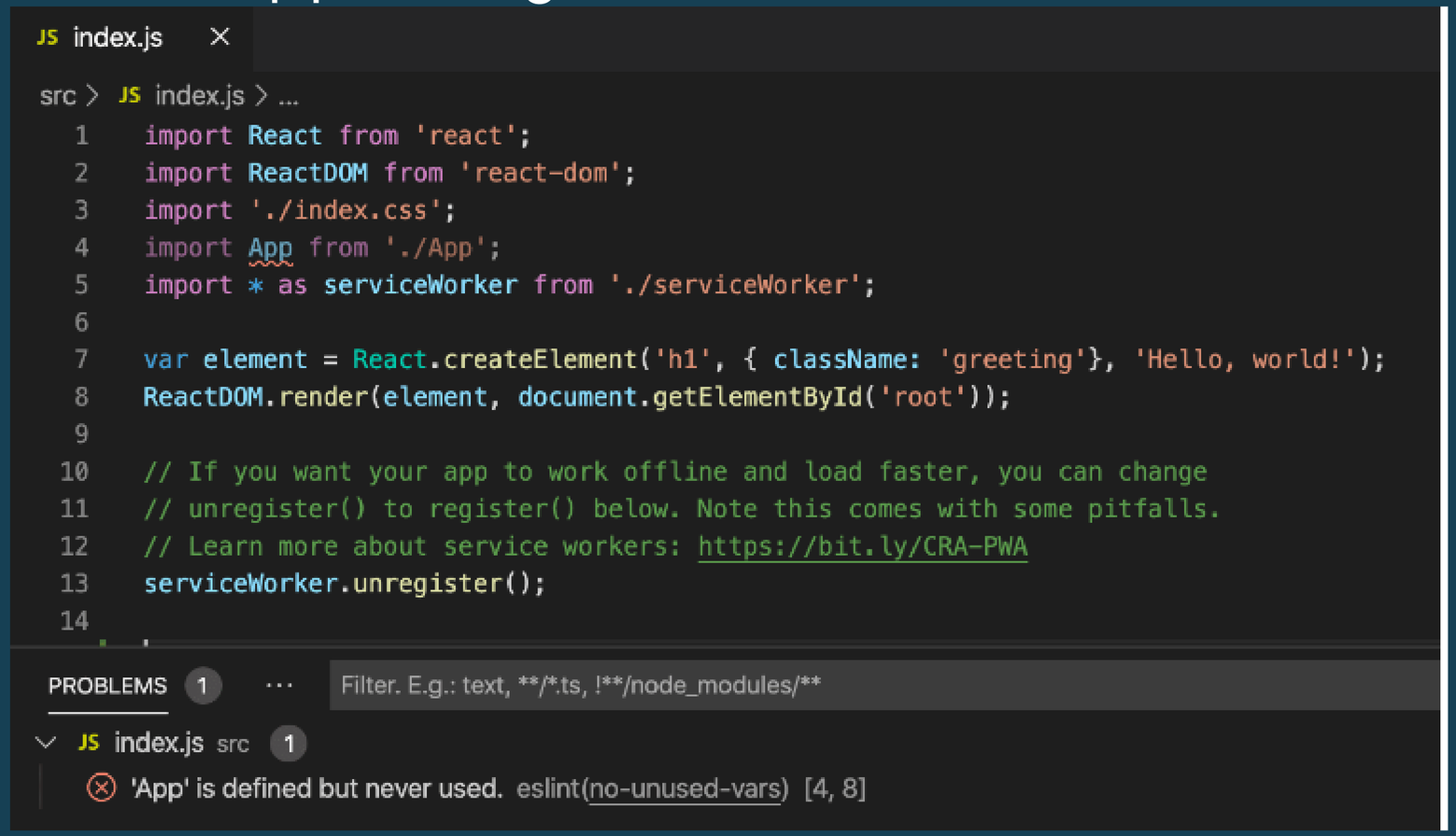
Once the ESLint extension is installed and VS Code reloaded, you'll want to create an ESLint configuration file, .eslintrc.js. You can create one using the extension's ESLint: Create ESLint configuration command from the Command Palette (Ctrl+Shift+P).



The command will prompt you to answer a series of questions in the Terminal panel. Take the defaults and it will create a .eslintrc.js file in your project root that looks something like this:

```
module.exports = {
env: {
browser: true,
es2020: true
},
extends: ['eslint:recommended', 'plugin:react/recommended'],
parserOptions: {
ecmaFeatures: {
jsx: true
},
ecmaVersion: 11,
sourceType: 'module'
},
plugins: ['react'],
rules: {}
};
```

ESLint will now analyze open files and shows a warning in index.js about 'App' being defined but never used.

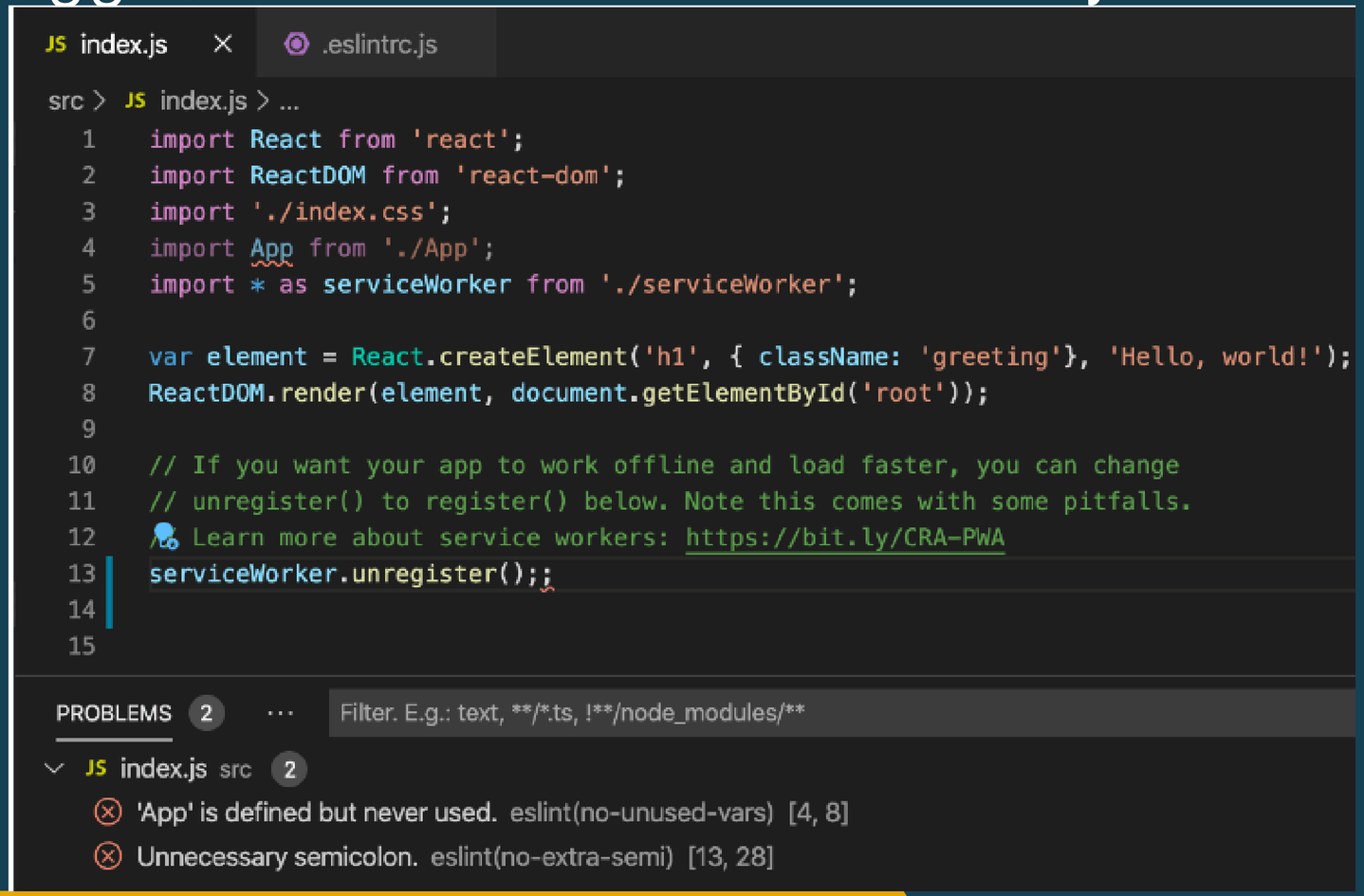


You can modify the ESLint rules in the .eslintrc.js file.

Let's add an error rule for extra semi-colons:

```
"rules": {
"no-extra-semi":"error"
}
```

Now when you mistakenly have multiple semicolons on a line, you'll see an error (red squiggle) in the editor and error entry in the Problems panel.



Skill AP
APSSDC

ANY QUERIES ?

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

Skill AP
APSSDC

THANK YOU

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED