From Zero To React

In this session, absolute beginners will be exposed to a modern web programming framework called React. In a few hours together, we will each complete the steps required to program a simple artificial intelligence app that will answer questions when asked.

You will be asked to type quite a bit of code from black boxes like the one below. If you don't like to type the code, or are having trouble, you can find the code here, to copy and paste: https://github.com/PhaedrusTheGreek/from-zero-to-react/tree/main/src

In this course, you will see these types of indications:

1. Numbered items are steps that must be completed

Comments in italics provide interesting information about the current step

["Grey boxes contain terminal commands that you need to type"]

$ Yellow boxes are examples of what you should see in the terminal. Checking these boxes will help you make sure you're doing things right.

```
<h1> Black Boxes are Code Examples </h1>
```

👀 Purple boxes are important things

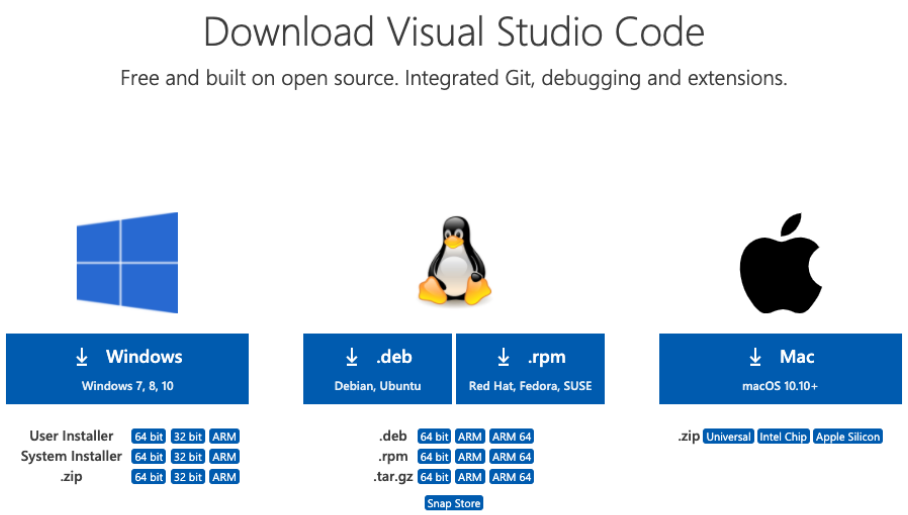
🤔 Blue boxes are interesting things that you can ignore if you want to.

Part A - Preparing the Development Environment

Install Visual Studio Code

Visual Studio Code is an app where you write code.

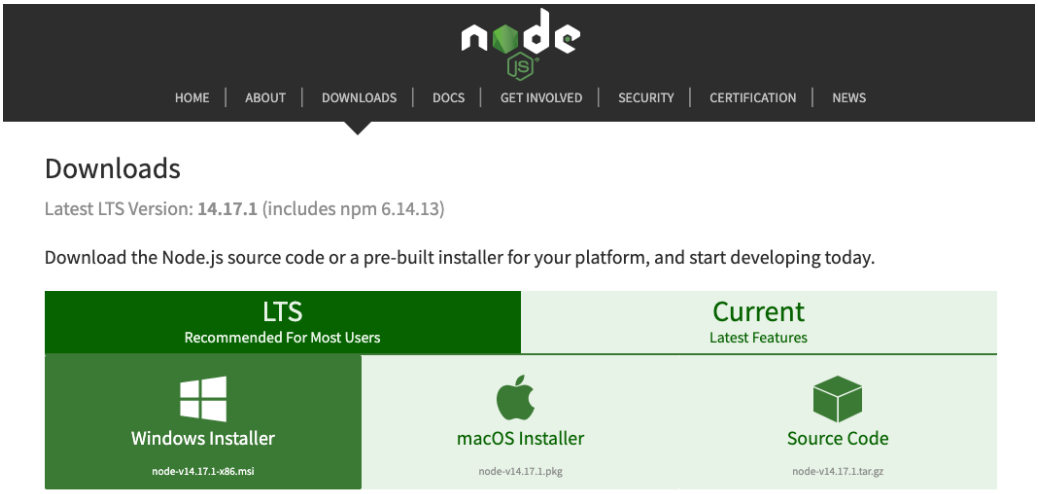
1. Search Google for “Download Visual Studio Code”
2. Click the correct download link on the correct page, and install the program.



Install NodeJS

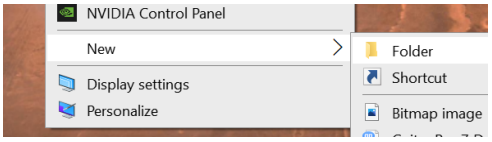
NodeJS is the server that runs your code.

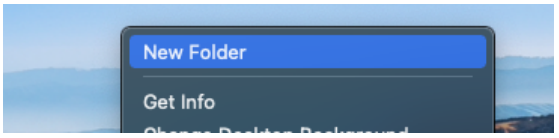
3. Search Google for “Download NodeJS”
4. Click the correct download link on the correct page, and install the program.



Create a folder where you will store your project

5. Right-Click on the Desktop and choose New Folder

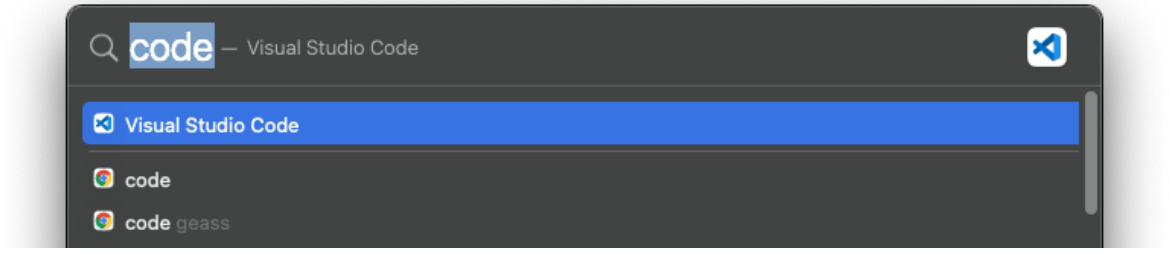


OR


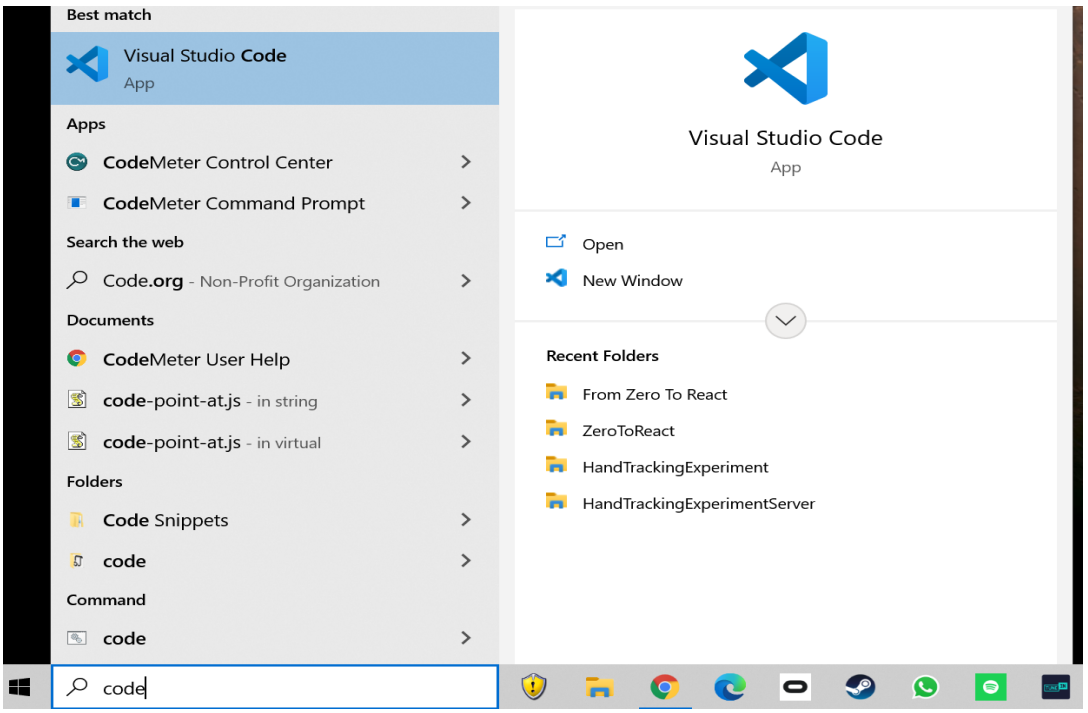
6. Name the folder “From Zero To React”

Open the new folder in Visual Studio Code

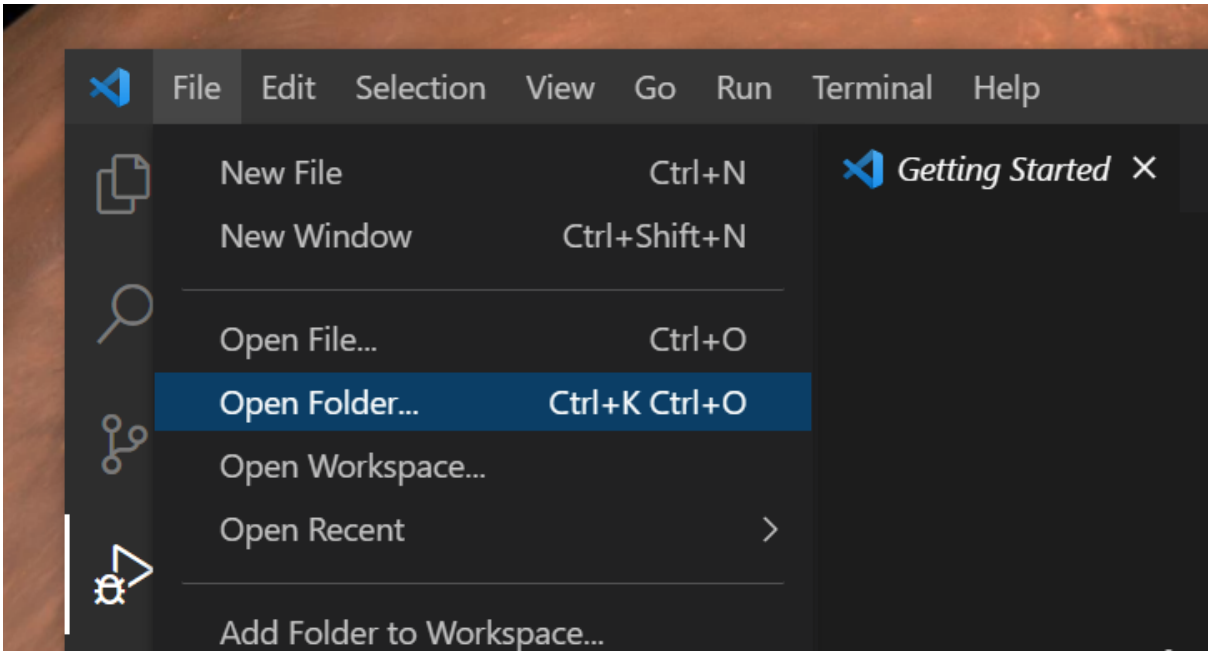
7. Open Visual Studio Code
   - On Mac, press Command-Space (⌘+Space) , and type “Code”



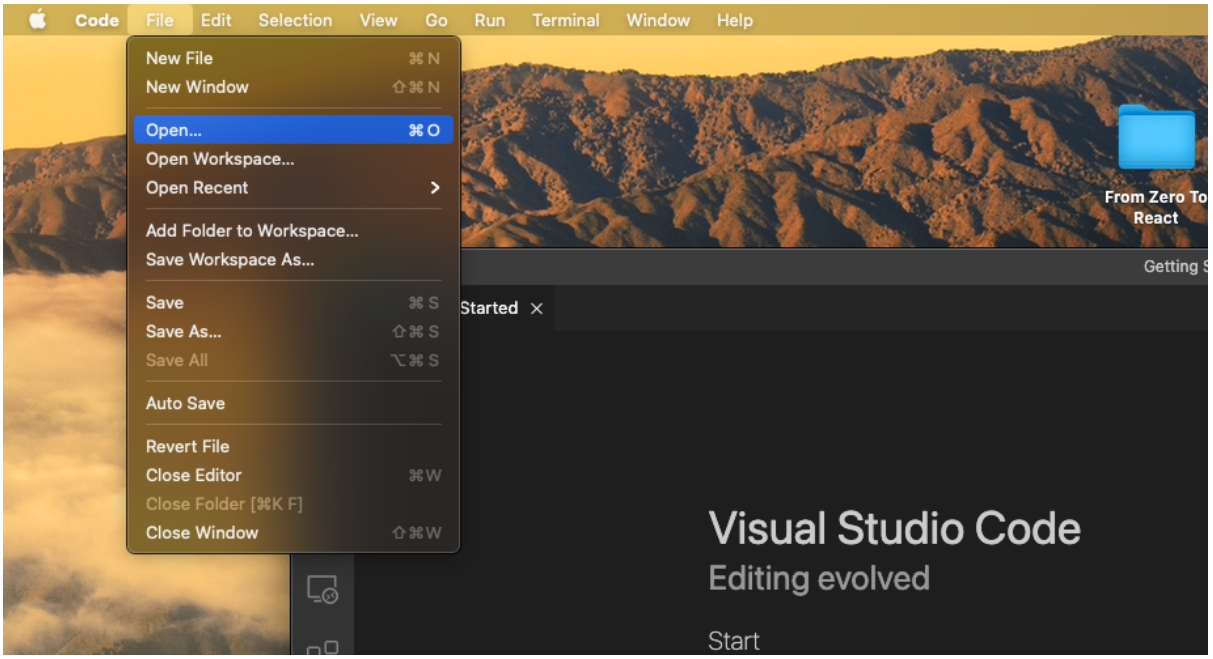
   - On Windows, Click down in the Search Area and type “Code”



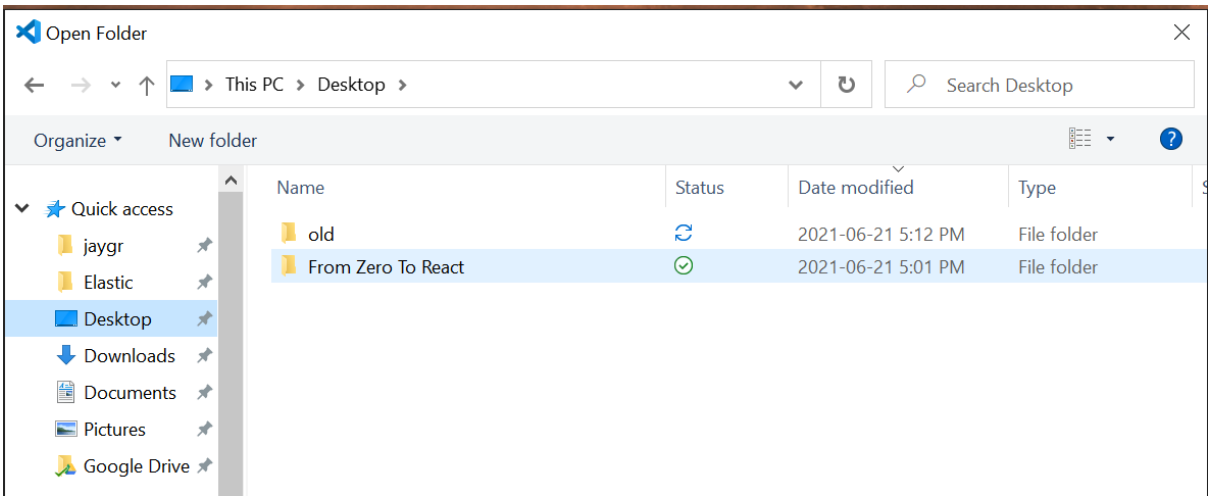
8. Open the newly created folder
    - In Windows by clicking File → Open Folder ...



    - In Mac OS by clicking File → Open…



9. Browse to and open the new folder.



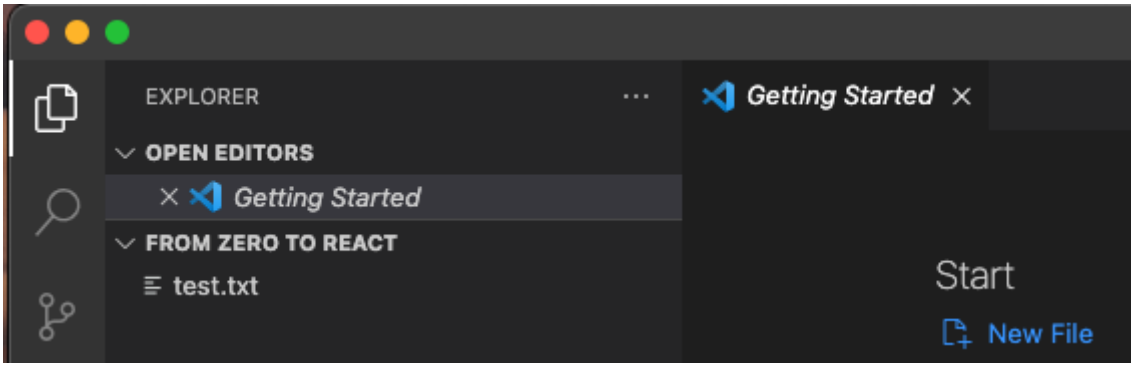
10. Start a Terminal by clicking Terminal → New Terminal

🤔 The terminal or a “command line” is a “low level” interface to the operating system. A graphical interface such as a button or a window is a “high level” interface.

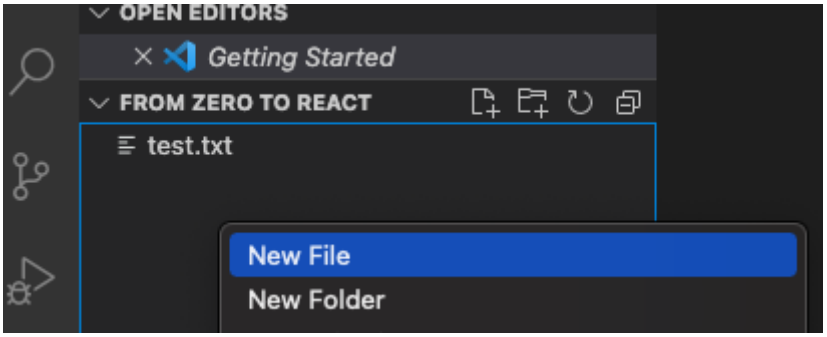
11. Use the terminal to create a file. Type the following command in the terminal, then hit the <Enter> key.

```
echo "hello" > test.txt
```

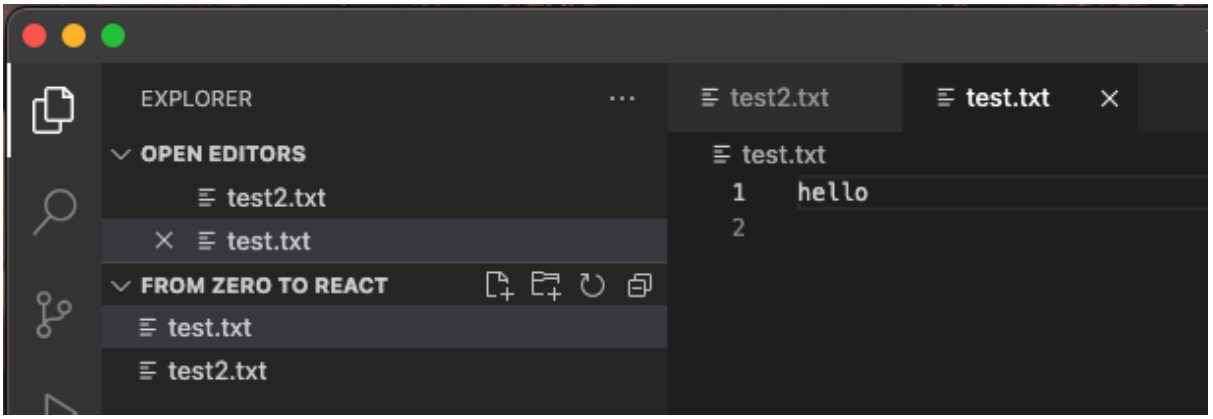
Notice the new file called "text.txt" appeared in the file explorer in Visual Studio Code



12. Creating a file by command line in step 11 was just for fun. Create a new file using the *User Interface* (UI) by right clicking in the file explorer and choosing "New File". Call this file "test2.txt"



13. You'll notice that VS Code automatically opens "test2.txt" for you in a code window. Double-click "test.txt" to open it too.



Notice that test.txt contains the word "hello", which we wrote to this file using the Command-Line Interface in step 11. Remember?

```
echo "hello" > test.txt
```

🤔 **echo** is just a program that just repeats the text you give it. The **>** or "redirect" operator means "send the output of this program to a file". For fun, try the echo command without the redirect.

14. Finally, open the "From Zero To React" folder on your desktop by double-clicking it. See anything familiar? You can close that folder after checking.

Part B - Learning to Navigate by Command-Line

15. Install a program called create-react-app on the Command-Line by issuing the following command:

```
npm install -g create-react-app
```

npm is a Command-Line app installed by the NodeJS installer. If it works, you should see output like this:

```
$ npm install -g create-react-app
/Users/jay/.nvm/versions/node/v12.20.1/bin/create-react-app →
/Users/jay/.nvm/versions/node/v12.20.1/lib/node_modules/create-react-app/index.js
+ create-react-app@4.0.3
added 67 packages from 25 contributors in 2.939s
```

🤔 **npm** is the "NodeJS Package Manager". Its purpose is to manage *dependencies*. A dependency is code that other people write that you want to include in your project. There is a *very* large community of software programmers on the internet that share code with each other for free to just be nice and helpful! This is called "The Open-Source Community". Notice in the above output - *"67 packages from 25 contributors"*

16. Run create-react-app, which will create the basic parts of our app.

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

🤔 **npx** is a program that runs other programs installed by **npm.**

If it works, you should see something like the following at the end of a long output:

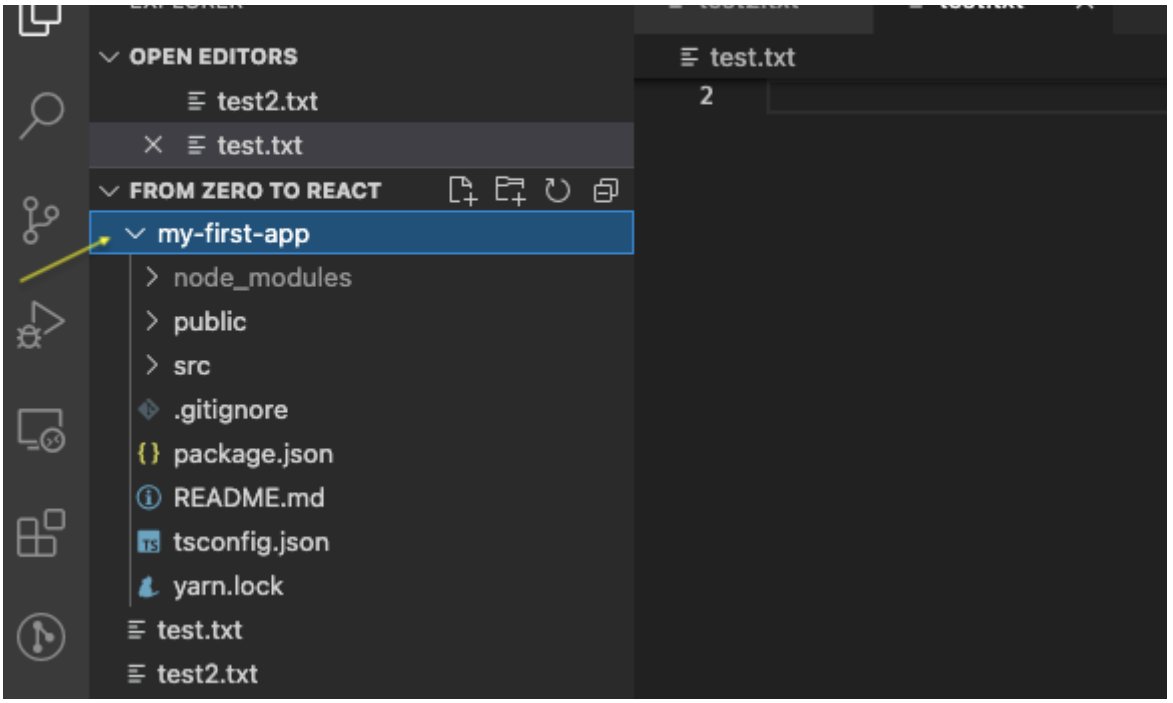
```
We suggest that you begin by typing:

  cd my-first-app
  npm start

Happy hacking!
PS C:\Users\jaygr\OneDrive\Desktop\From Zero To React>
```

17. Our project has been created in the "my-first-app" folder. Let's move into that folder now.

a) On the command line, you can *change directory* with the **cd** command.

```
cd my-first-app
```

b) Next, in VS Code:



18. Notice the files that are in the "my-first-app" folder. These are the files that create-react-app has created for you.

a) In the UI, open some other folders and see what is inside them.
b) Close all the folders, but keep the "my-first-app" folder open.
c) Find and open the "src" folder. This is where our code will go.

🤔 What are these files for?

```
> node_modules    // This is where npm installs dependencies
> public          // This is where any file can go, for example a video or picture
> src             // This is where the source code goes
.gitignore        // A special file used by git
package.json      // This describes the project and its dependencies
README.md         // This is like a title page for the project
<anything>.lock   // This helps npm remember which dependencies are installed
```

19. *List* the files from the Command-Line

ls

```
$ ls
README.md       node_modules    package.json    public          src
yarn.lock
```

20. *Long-List* the files from the Command-Line

ls -alh

```
$ ls -alh
total 1032
drwxr-xr-x    11 jay  staff   352B  6 Jul 19:27 .
drwxr-xr-x@    5 jay  staff   160B  6 Jul 19:26 ..
drwxr-xr-x    12 jay  staff   384B  6 Jul 19:27 .git
-rw-r--r--     1 jay  staff   310B  6 Jul 19:27 .gitignore
-rw-r--r--     1 jay  staff   2.1K  6 Jul 19:27 README.md
drwxr-xr-x  1055 jay  staff    33K  6 Jul 19:29 node_modules
-rw-r--r--     1 jay  staff   971B  6 Jul 19:27 package.json
drwxr-xr-x     8 jay  staff   256B  6 Jul 19:27 public
drwxr-xr-x    11 jay  staff   352B  6 Jul 19:27 src
-rw-r--r--     1 jay  staff   498K  6 Jul 19:27 yarn.lock
```

🤔 What does each flag do in this long listing?

```
-a   // List all files, even hidden ones.   Hidden files start with a dot (.)
-l   // Do a long listing (shows extra information)
-h   // Use human readable file sizes (Bytes, Kilobytes, Megabytes, Gigabytes)
```

21. Change directory to the "src" directory , and do a long listing.

```
cd src
ls -alh
```

```
$ cd src/
~/Desktop/From Zero To React/my-first-app/src
$ ls -alh
total 64
drwxr-xr-x  10 jay  staff   320B  6 Jul 21:13 .
drwxr-xr-x  10 jay  staff   320B  6 Jul 21:13 ..
-rw-r--r--   1 jay  staff   564B  6 Jul 21:13 App.css
-rw-r--r--   1 jay  staff   528B  6 Jul 21:13 App.js
-rw-r--r--   1 jay  staff   246B  6 Jul 21:13 App.test.js
-rw-r--r--   1 jay  staff   366B  6 Jul 21:13 index.css
-rw-r--r--   1 jay  staff   500B  6 Jul 21:13 index.js
-rw-r--r--   1 jay  staff   2.6K  6 Jul 21:13 logo.svg
-rw-r--r--   1 jay  staff   362B  6 Jul 21:13 reportWebVitals.js
-rw-r--r--   1 jay  staff   241B  6 Jul 21:13 setupTests.js
```

22. Go back to the "my-first-app" directory.

In Command-Line language, two dots like this [**..**] is a cute emoji, but really it means "the parent folder". You could also say "go up a folder", or "go to the higher folder"

```
cd ..
```

```
~/Desktop/From Zero To React/my-first-app/src
$ cd ..
~/Desktop/From Zero To React/my-first-app
$
```

23. Double check that you are in the "my-first-app" directory by using the pwd program which stands for "Print Working Directory"

```
pwd
```

```
$ pwd
/Users/jay/Desktop/From Zero To React/my-first-app
~/Desktop/From Zero To React/my-first-app
$
```

24. Start the App.

```
npm start
```

Your web browser should automatically open, and you should see this:

Part C - Coding in React

React is a modern and powerful language designed to make building web apps easy.

🤔 ... The Parts of a Web App ...
A **Web Browser** is a program used to *render* **HTML**.
HTML is a web page formatting language (.html files)
CSS is a web page styling language (.css files)
Javascript is a web page programming language (.js files)
React combines *all of the above* into a single language (.js files).

👀 In this section you are going to see web-programming languages called HTML, Javascript, and CSS! **Don't worry** about understanding everything you're typing. What is important is that you **have some fun** playing with these parts.

25. Read the following

A *React* app is made up of **Components**. A component looks like this:

```
<Question/>
```

Components are like building blocks in a big puzzle. The App is the main component. **Our program has a Question and and Answer component** in the app, so the main structure of the React app is like this:

```
<App>
  <Question/>
  <Answer/>
</App>
```

🤔 React is powerful because you can code your own Components.

👀 *React code goes in files with **.js** extension.*

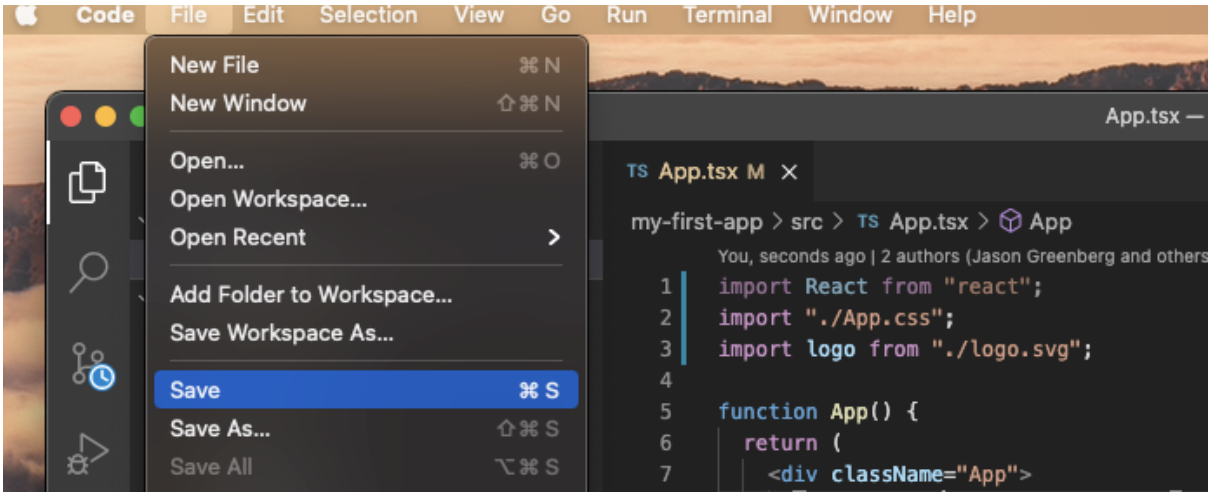
26. **Find** and **Open** (double click) the file "**App.js"** and open it in VS Code. It's in the "**src**" directory.
27. **Erase** these lines**:**

```
    <p>
      Edit <code>src/App.js</code> and save to reload.
    </p>
    <a
      className="App-link"
      href="https://reactjs.org"
      target="_blank"
      rel="noopener noreferrer"
    >
      Learn React
    </a>
```

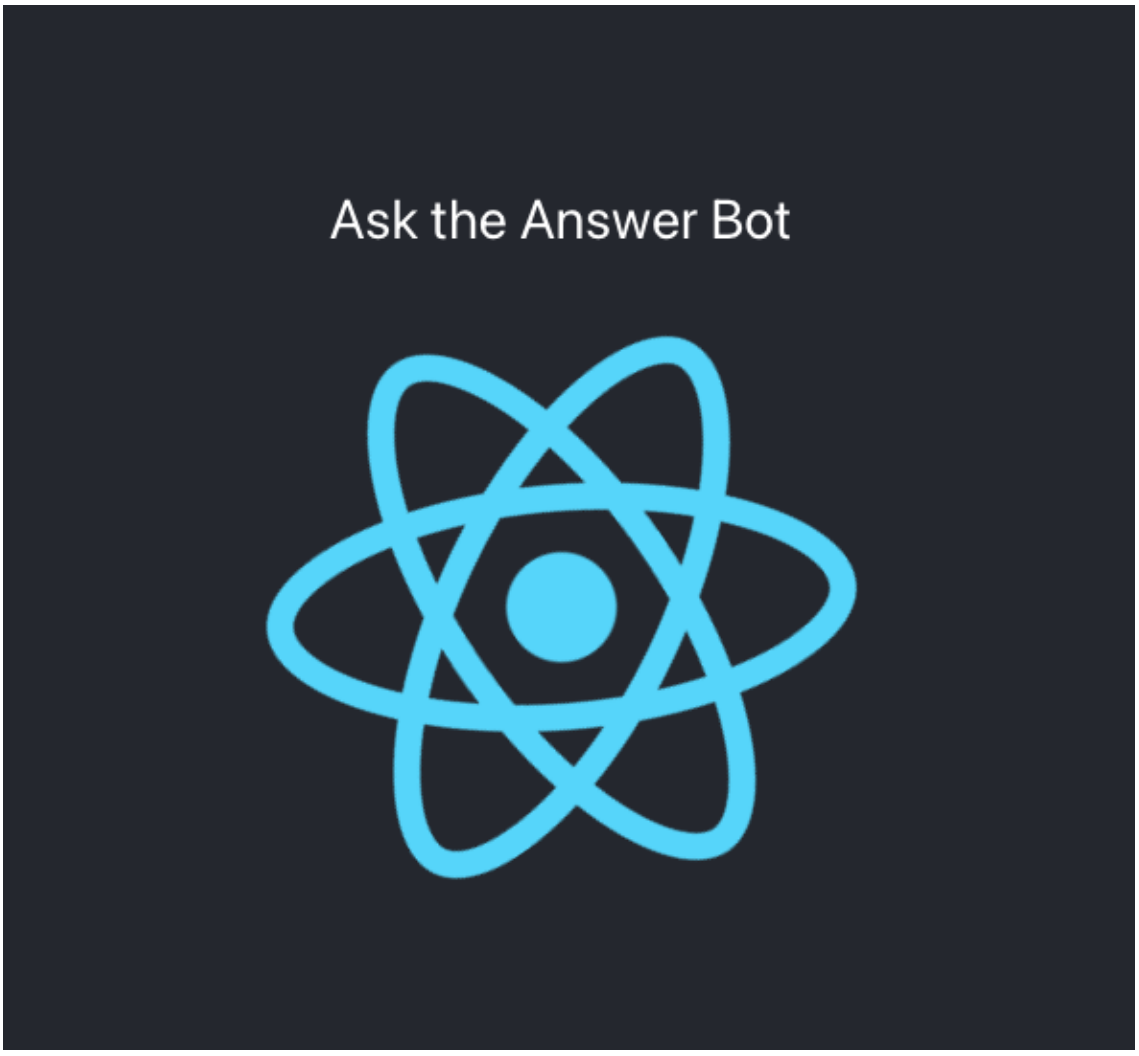
28. Write "Ask the Answer Bot" above the <img> tag. It should look like this now:

```
function App() {
 return (
   <div className="App">
     <header className="App-header">
       Ask the Answer Bot
       <img src={logo} className="App-logo" alt="logo" />
     </header>
   </div>
 );
}
```

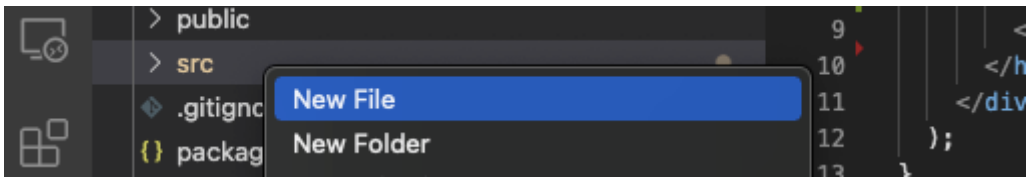
29. Save this file (File → Save). (When you save a file, the web browser will automatically update the running app)

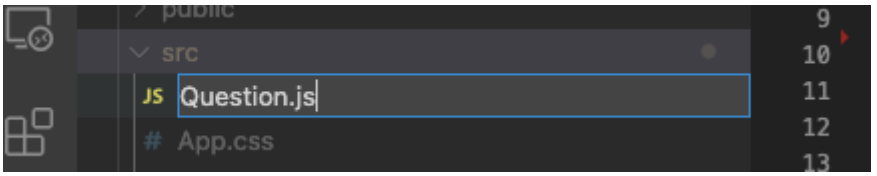


30. Check your web browser, it should look like this now:



31. Create a new file called "**Question.js**" in the "**src**" directory. (Right Click "src" → New File)





32. Code **a React Component** called "**Question"** !

a) Read this important information on finding the right keys to use. You don't have to memorize it, but you can refer back to it when you're coding.

> 👀 **Here's how to find the right keys to use!**
>
> Functions, like **function Question () { }** use both **parentheses** and **curly braces**.
>
> The **return( )** statement uses **parentheses** . That is Shift-9 and Shift-0
>
> The **onChange={ }** statement uses **curly braces.** That is shift-[and shift-]. You can find them beside the P key.
>
> Tags like ***<input>*** use **greater-than and less-than symbols**. That is Shift-Comma and Shift-Period. You can find them beside the M key.

b) This is an example of code with an error. Notice that it has many parts red underlined - it means you got something wrong - don't worry, this happens all the time, even to professional programmers!

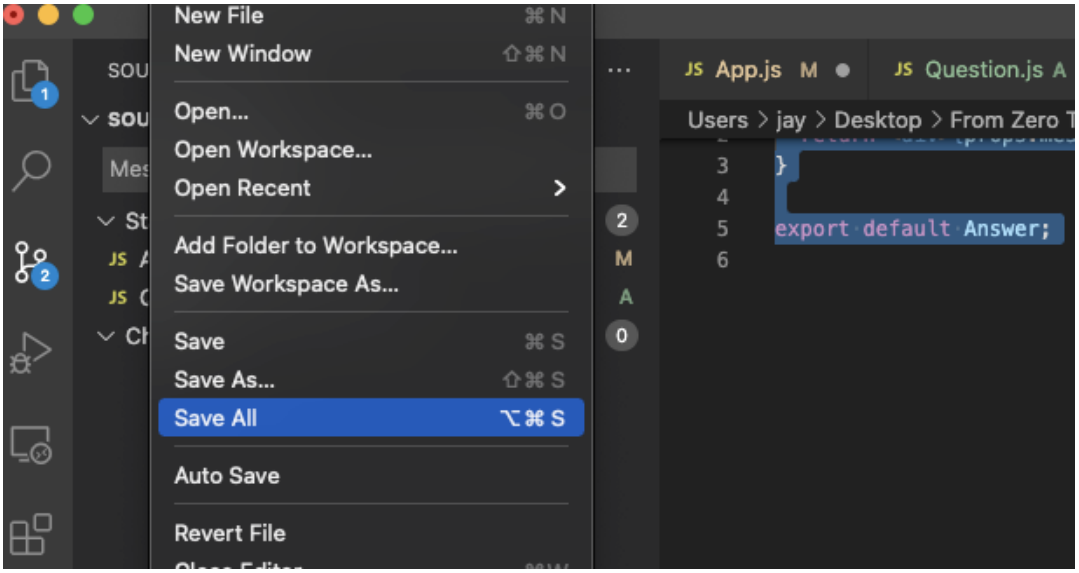
my-first-app > src > JS Question.js > default

```
1  import "./Question.css";
2
3  function Question(props) {
4    return {
5      <input className="MyQuestionClass" onChange={props.handleQuestion}></input>
6    );
7  }
8
9  export default Question;
```

c) This is the correct code. Enter it into the **Question.js** file now. You can Copy and Paste this file from Github if you want.

```
import "./Question.css";

function Question(props) {
 return (
   <input className="MyQuestionClass" onChange={props.handleQuestion}></input>
 );
}

export default Question;
```

33. Save your work (File → Save All).



34. Create a new file called "Question.css" in the "src" directory and enter the following code into the new file. You can Copy and Paste this file from Github if you want.

```
.MyQuestionClass {
  font-size: 25px;
  width: 75%;
  background-color: grey;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
}
```

35. Code the Answer Component! Create a new file called "Answer.js" in the "src" directory, and enter the following code into the new file. You can Copy and Paste this file from Github if you want.

```
function Answer(props) {
 return <div>{props.message}</div>;
}

export default Answer;
```

36. Open the **src/App.js** file for editing
37. **Import our new components** by adding these 2 lines to the top of the file, after the other **import** statements.

```
import Question from './Question'
import Answer from './Answer'
```

🤔 Import statements tell Javascript that we want to use code from other files.

38. Insert the **Question** and **Answer** components into the App

Add the following 2 lines

```
<Question handleQuestion={answerQuestion}></Question>
<Answer message={message}></Answer>
```

As shown here:

```
import logo from "./logo.svg";
import "./App.css";
import Question from "./Question";
import Answer from "./Answer";

function App() {
  return (
    <div className="App">
      <header className="App-header">
        Ask the Answer Bot
        <Question handleQuestion={answerQuestion}></Question>
        <img src={logo} className="App-logo" alt="logo" />
        <Answer message={message}></Answer>
      </header>
    </div>
  );
}

export default App;
```

Notice you will have some errors at this point. Javascript is mad because answerQuestion and message don't exist, so let's create them in the next step.

39. Make Javascript happy by typing these lines:

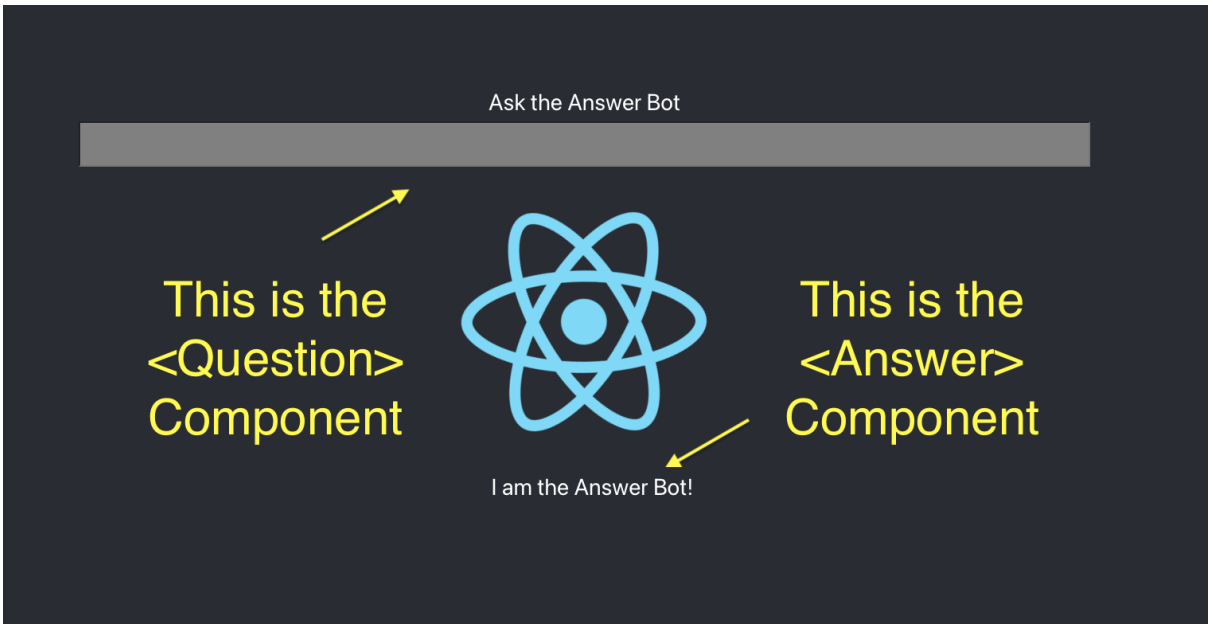
```
const message = "I am the Answer Bot!";
const answerQuestion = function ( ) { };
```

Enter those lines right below the App() function declaration like this:

```
function App() {
  const message = "I am the Answer Bot!";
  const answerQuestion = function () {};
```

🤔 **message** is a *string variable*, and **answerQuestion** is a *function*

40. **Save All** your work and check the web browser, it should look like this:



Part D - Coding the Artificial Intelligence (AI)

41. Create the training data.

> 👀 We have to train our AI on how to answer questions. We will teach it to say certain things when it is asked a question with certain words. Notice that the list of words is surrounded in **square brackets** like this: **[ ].** They are found next to the P, and are the same keys used for the curly braces, but without the shift key.

Here is the basic training data file:

```
{
 "Hello, Human": ["hello", "hi"],
 "I'm the answer Bot": ["name"],
 "Montreal Canadiens is my favourite hockey team": ["hockey", "playoffs"],
 "Vegetarian pizza is my favourite": ["pizza"]
}
```

42. Create a new file called **"TrainingData.json"** in **"src",** and type the above in that file.

> 🤔 **.json** files stand for **"Javascript Object Notation"**, and represent data structures, the same as would be stored in a real *database*.

43. Edit App.js and import the training data (put this at the top of the file):

```
import trainingData from "./TrainingData.json";
```

44. In the same file, erase this line:

```
const answerQuestion = function () {};
```

And replace it with this (the real answerQuestion function) , in the same place:

```
function answerQuestion(event) {
  const asked = event.target.value;
  const askedWords = asked
    .toLowerCase()
    .replace(/[^a-zA-Z ]/g, "")
    .split(/ +/);

  for (const [answer, keywords] of Object.entries(trainingData)) {
    if (asked.length > 2) {
      for (const askedWord of askedWords) {
        if (keywords.includes(askedWord)) {
          setMessage(answer);
        }
      }
    } else {
      setMessage("...");
    }
  }
}
```

The above is a lot to type! Feel free to copy and paste this part from Github.

🤔 the **answerQuestion** function is the most interesting part of the App. It compares each word that the asker asks, and checks to see if any word matches any word in our Training Data file. If so, then it sets the message to be our sentence. For example, if the asker says "**hi** bot", then the bot says "**Hello, Human**"

45. In the ame file, add this import at the top:

```
import { useState } from "react";
```

🤔 **useState** is a special React feature that manages the current state of a component. We can set a default value for any variable , such as "Welcome, human!" for message, and it also provides us with a function, **setMessage( ),** and access to the **{ message }** variable.

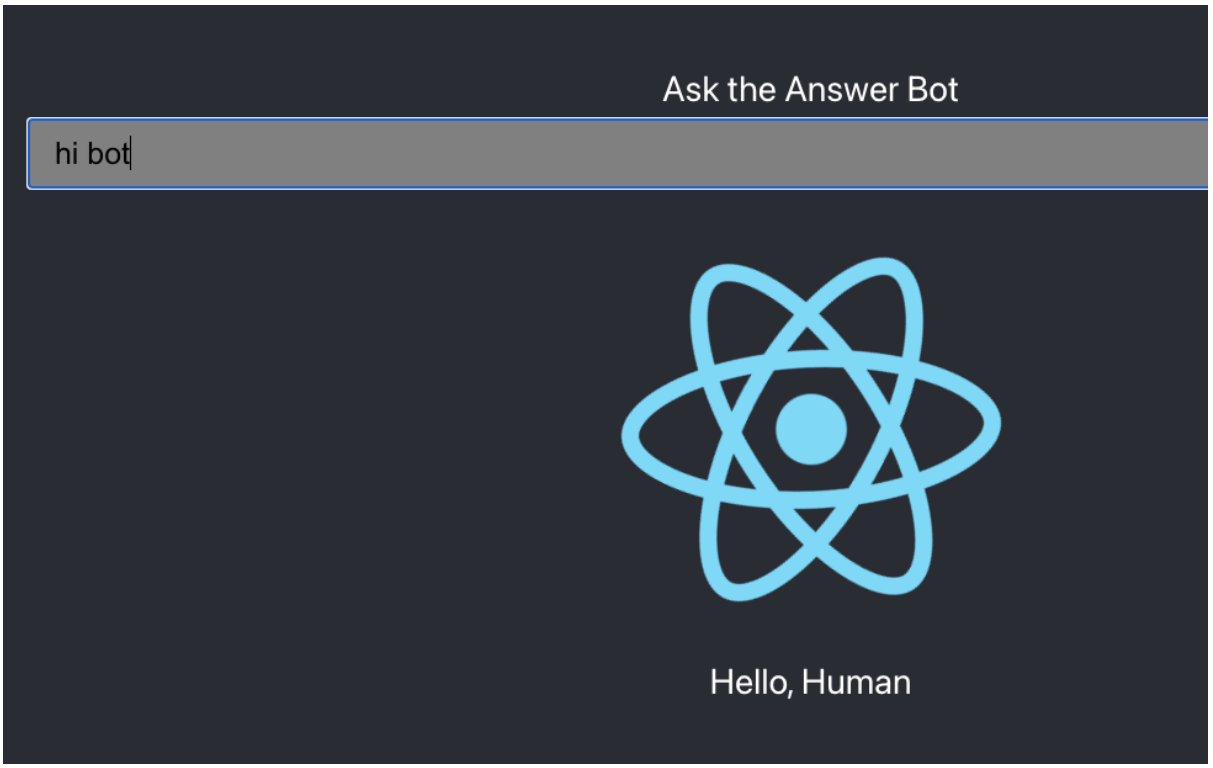
46. In the same file, erase this line:

```
const message = "I am the Answer Bot!";
```

And replace it with this:

```
const [message, setMessage] = useState("Welcome, human!");
```

47. **Save All** your work and check the browser. Talk to the bot!



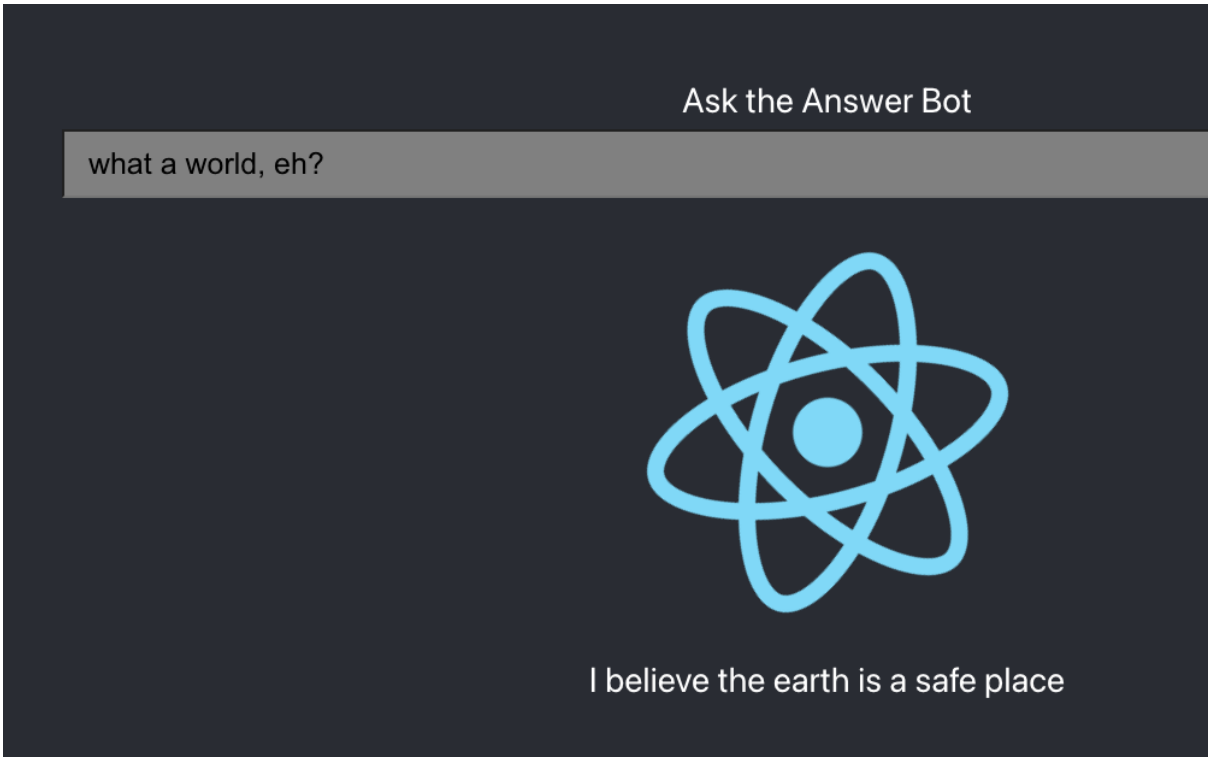
48. **Train the bot more!** Add the following to the **TrainingData.json** file:

```
"I believe the earth is a safe place": ["world", "earth", "planet"]
```

```
{
  "Hello, Human": ["hello", "hi"],
  "I'm the answer Bot": ["name"],
  "Montreal Canadiens is my favourite hockey team": ["hockey", "playoffs"],
  "Vegetarian pizza is my favourite": ["pizza"],
  "I believe the earth is a safe place": ["world", "earth", "planet"]
}
```

👀 If this turns your file red with errors, don't worry, you need to fix up the formatting. There should be a comma at the end of every line except the last one, like above. See if the 2nd last line needs a comma.

49. **Save All** your work and test the new training:



50. Congratulations! You completed the course. Did you know you can deploy your app for free online? **https://blog.logrocket.com/8-ways-to-deploy-a-react-app-for-free/**

Instructor Notes

Beginning of lesson instructions:

Point out in advance that #32 is probably the hardest step, so take your time. In fact it might make sense to review this step in advance with the students, pointing out what errors look like, encouraging they ask for help at this step, etc.

Point out that while typing the code is the best way to learn, all code is available for copy and paste here:
https://github.com/PhaedrusTheGreek/from-zero-to-react/tree/main/src (link is on first page).
At this point, explain what Github is, make sure everyone has it open in their browser.
Some steps will be annotated that copying will work well.