Coding Challenge – LivePerson

# How the solution fits together

The solution is comprised of an HTML file named index.html and a JavaScript file named LivePersonChallenge.js.

1. Within the HTML file, I’ve created some text along with an input text field wherein a submit button can call a function to open a connection, send the custom message and then close the connection with the LivePerson Agent console.
2. In the JavaScript file, the user can enter his/her account number at the top, followed by a line of code that is listening for the button click to call the function to open a connection, send the custom message and then close the connection with the LivePerson Agent console.
3. Once the button is clicked, the custom text is received, stored in a variable and then the sendMessage() function is called that can open a connection, send the custom message and then close the connection with the LivePerson Agent console.

In the sendMessage() function, the process outlined in the task is followed (comments in the code explain what each step does). Appropriate error handling has been put in place for each of the fetch blocks.

## Screenshots

We can copy the code from LivePersonChallenge.js to index.js on a Code Sandbox instance in the cloud for testing. In the screenshot below, I’ve sent the text messages “asdf” and “school time”.

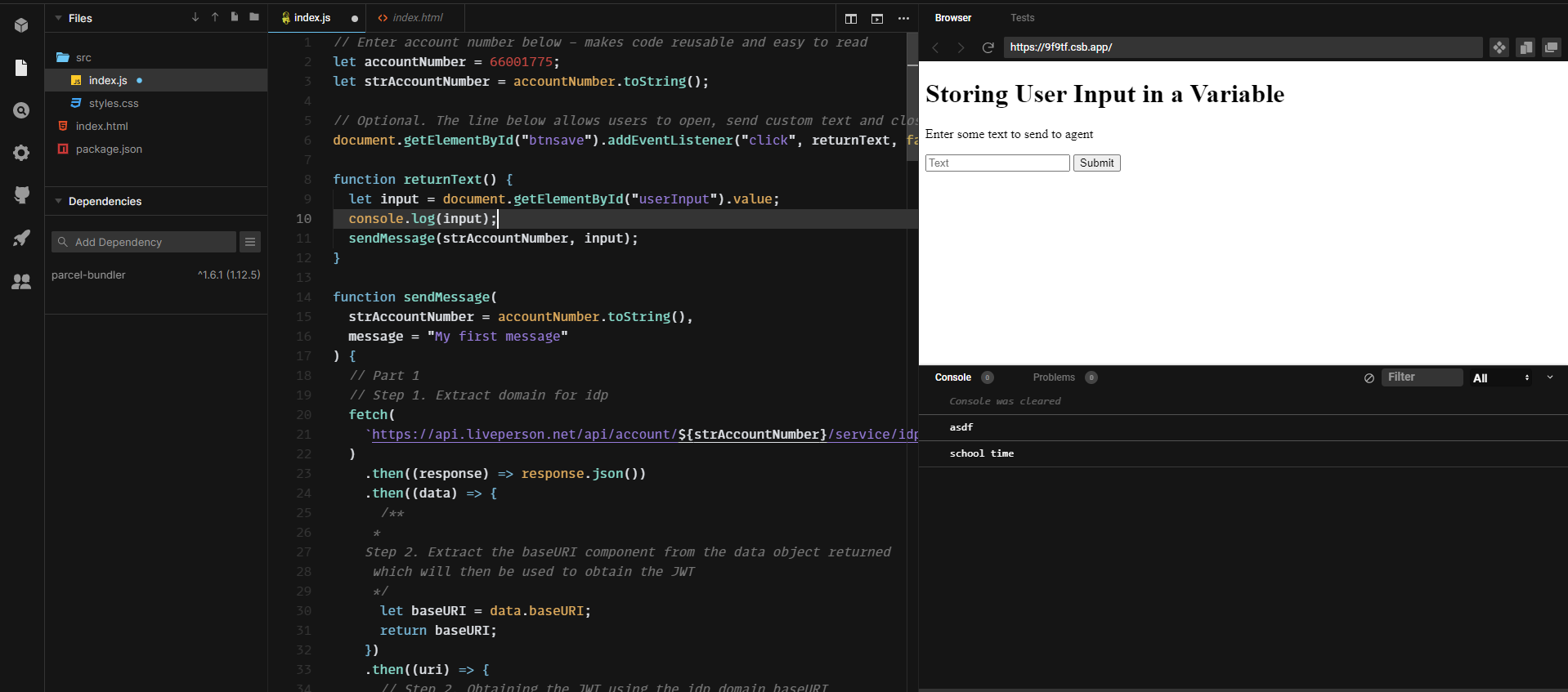


Figure : Testing the app on Code Sandbox – sent text ‘asdf’ and ‘school time’

The screenshots below show the pieces of text received by the Agent workspace console

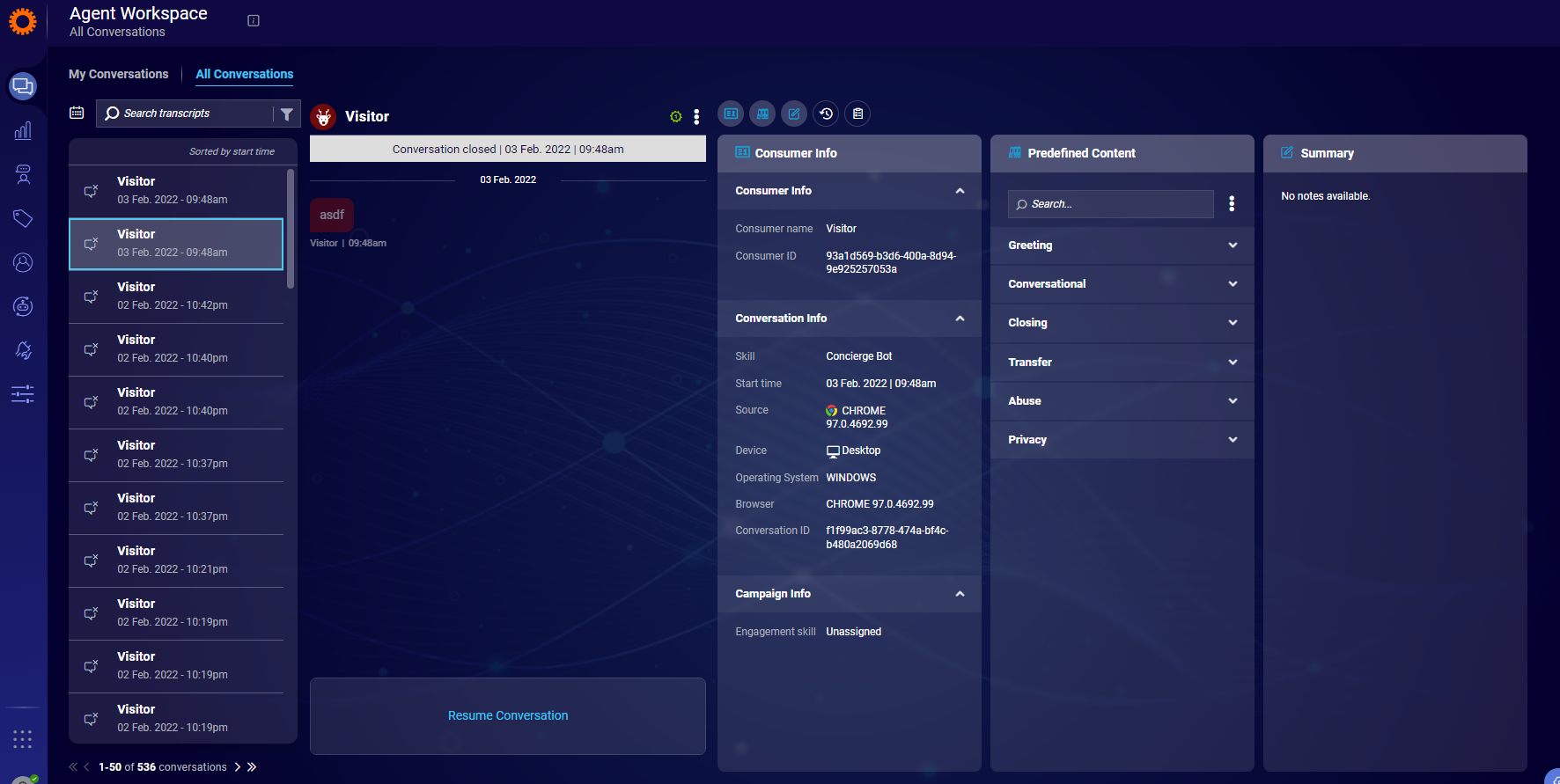


Figure : 'asdf' text received on Agent workspace console

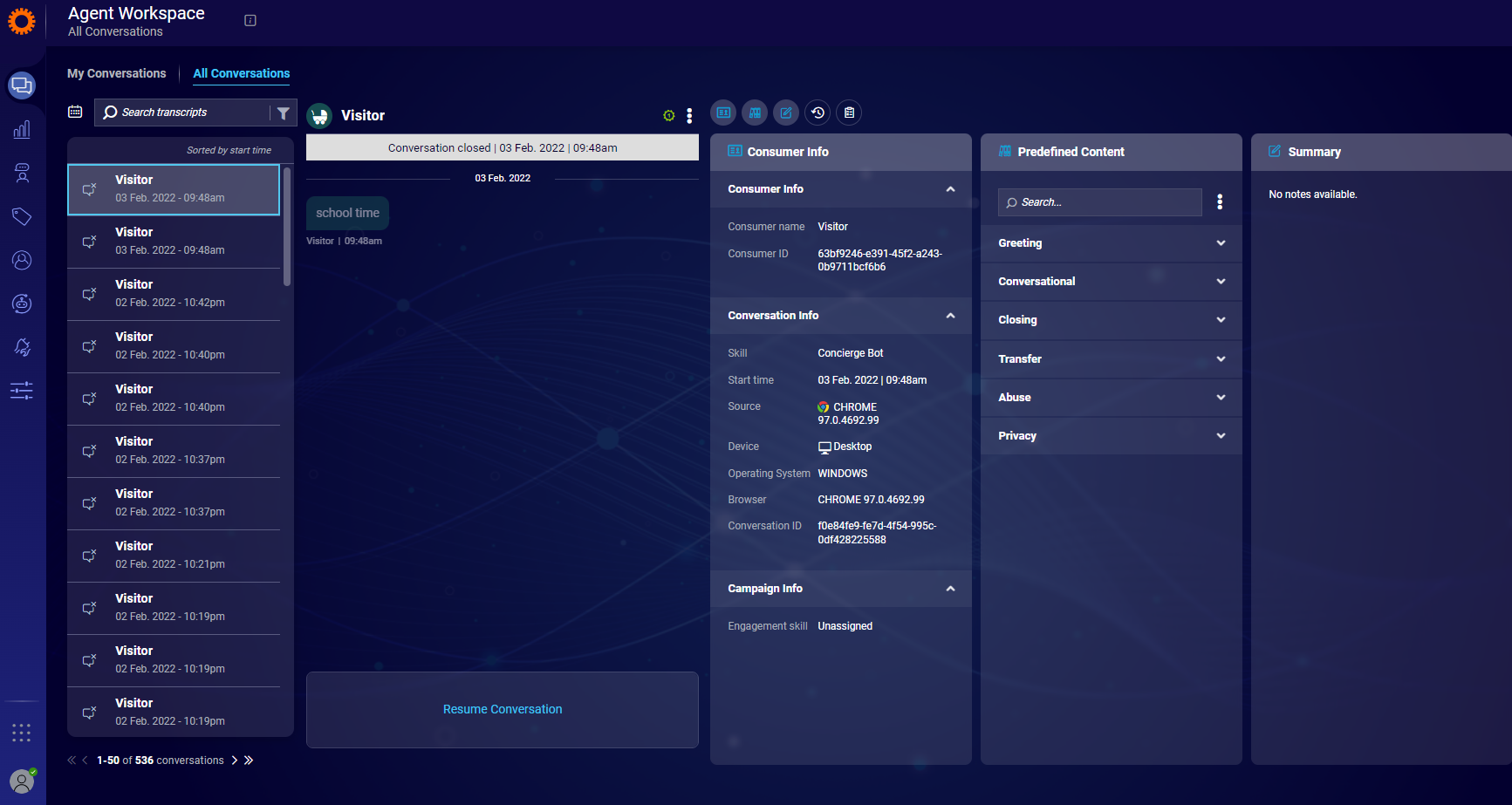


Figure : 'school time' text received on Agent workspace console

Furthermore, a unit testing framework has been developed to test the most important function sendMessage.js through the testing file sendMessage.test.js using the **jest** testing framework. This was new to me and I quite enjoyed it. The documentation is easy to learn and use the tool, and there are quite a few resources (videos, forums, posts) of people outlining how to use the tool and tackle any problems while getting started.

# The Good, the Bad and the Ugly

1. The main exercise was straightforward to implement, although ideally, I would’ve liked to have been able to store the output of each fetch block separately or modularize the various functions for easy testing and readability, unlike the monolith you see before you. I tried my best to use async/await functionality for this, but had no luck. I’m sure there is a more elegant way to code this solution.
2. For the optional elements of the exercise, again, easy to learn and implement. My first hurdle here was with the onclick listener. It refused to work if I placed the onclick listener in the HTML button element. So, I had to create a “click” listener in the JavaScript code instead after [some advice from StackOverflow](https://stackoverflow.com/questions/17378199/uncaught-referenceerror-function-is-not-defined-with-onclick).
3. Unit test functionality was a new area of learning for me, so I’m thankful that this challenge helped me learn something. Jest documentation is pretty straightforward to get started with thankfully. I just need a better logic to test so I know what specific functionality I need to test against, however the framework created here is just as an example. Because Jest requires a return value to compare against, I just returned the conversationId obtained during the function execution while testing the sendMessage() function.

# See the code in action

1. Create an account on [Code Sandbox](https://codesandbox.io/), login to your account and then create a new Vanilla JavaScript instance as shown in the figure 4 below.
2. Copy and paste the code from LivePersonChallenge.js in the GitHub repository shared with you earlier into the index.js file in this Code Sandbox instance.
3. Copy and paste the code from the file index.html in the GitHub repository shared with you earlier into index.html file created in this instance (only the four HTML elements in the <body> tag please).
4. Ensure that index.html and index.js files are linked to each other via the <script> tag in index.html on Code Sandbox (see figure 5 and 6 for reference)
5. Enter some text into the text field and hit submit. Watch the message come through on the Agent workspace console as shown in figure 1, 2 and 3 above. You can also use the “Live Server” extension to host this application and watch the output locally on your computer.

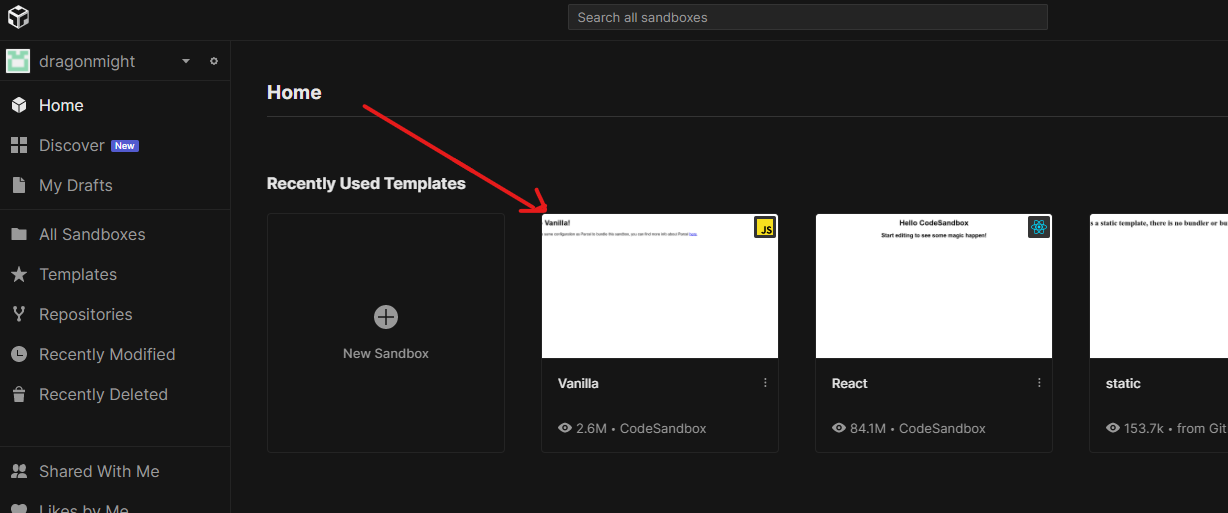


Figure : Create vanilla JS instance on Sandbox

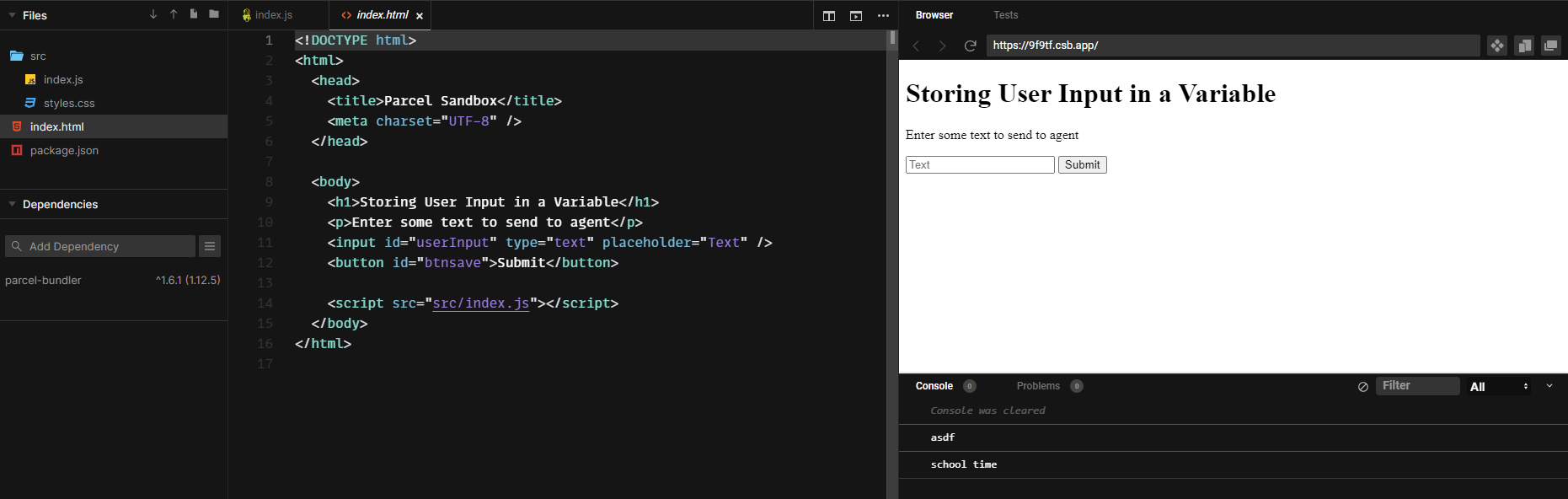


Figure : HTML file

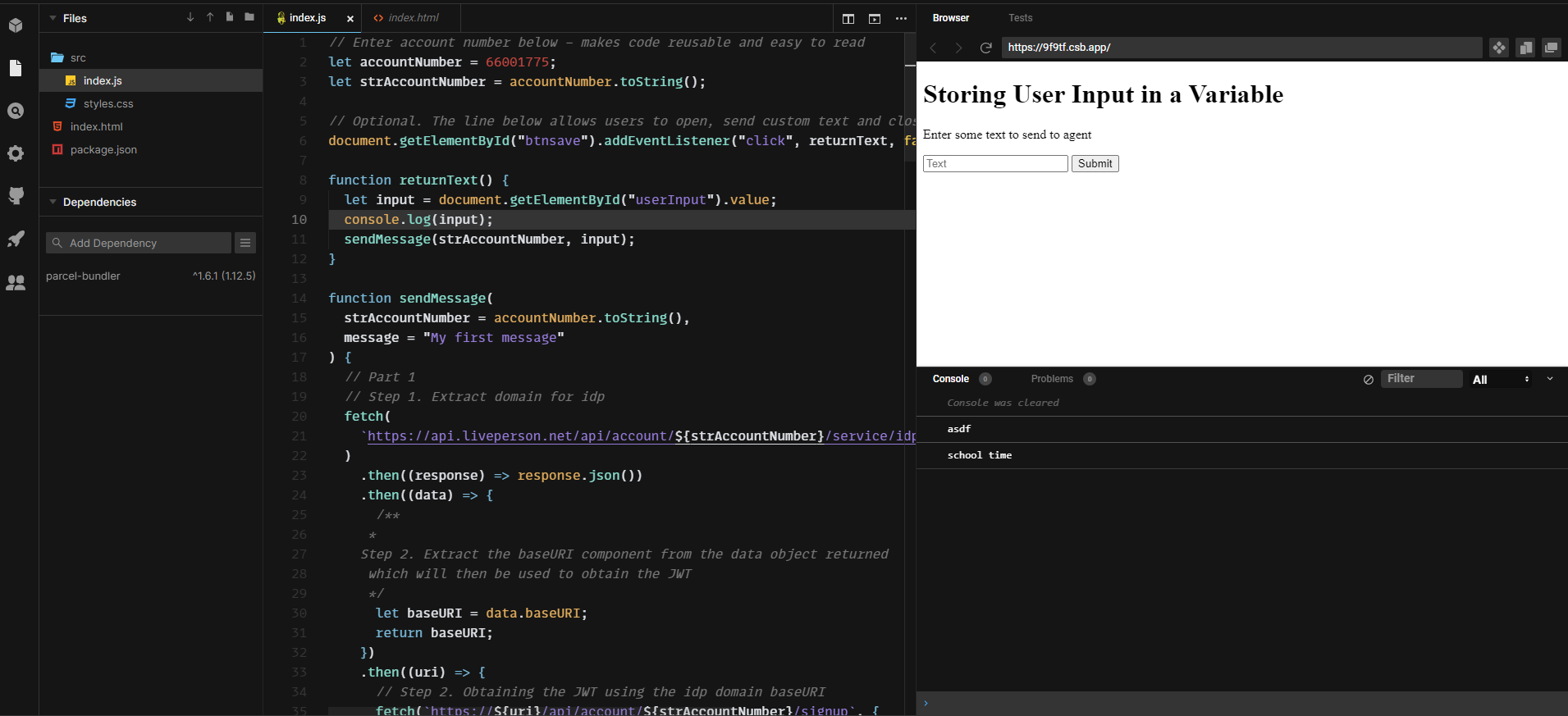


Figure : JavaScript file