**Walkthrough Notes:**

Went through the tutorial offered at this website:

<https://www.taniarascia.com/getting-started-with-vue/>

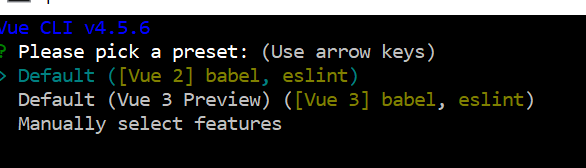
-Installed node and npm on my computer: <https://nodejs.org/en/download/>

-Installed Vue CLI on my computer:

npm i -g @vue/cli @vue/cli-service-global

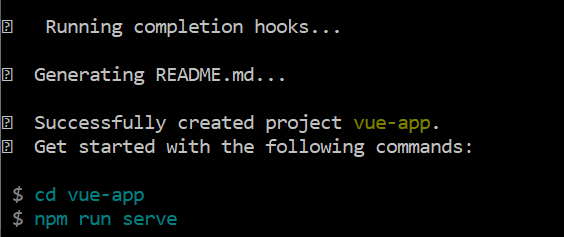
After installation of node.js and Vue CLI, typed in

> vue create vue-app



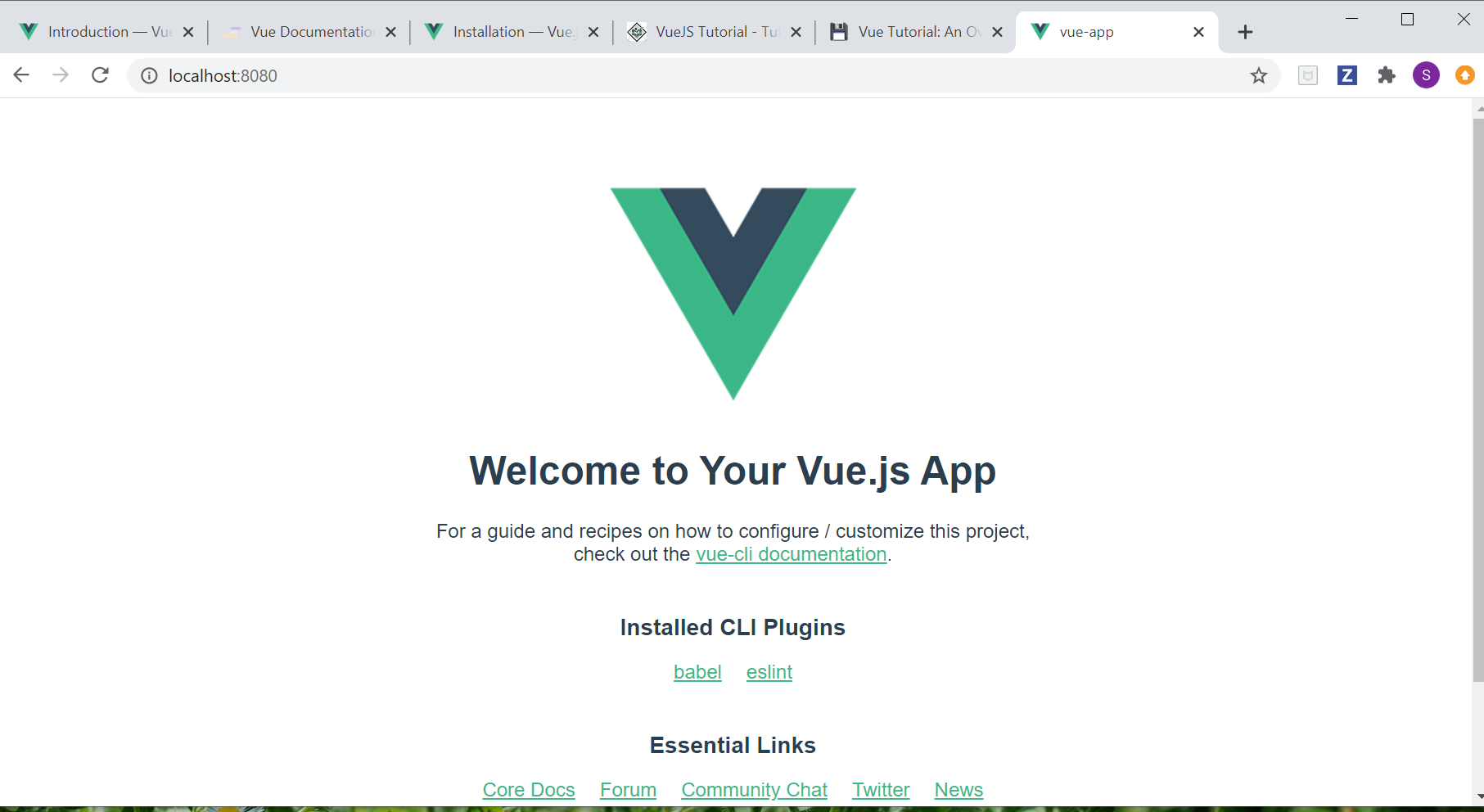
**Figure1. Chose the default preset for vue-app.**

It initializes git repo and other things, then provides commands to get started:



**Figure 2. End of project setup output.**

Changing directories to ‘vue-app’ and launching server, I get the following on localhost:8080.

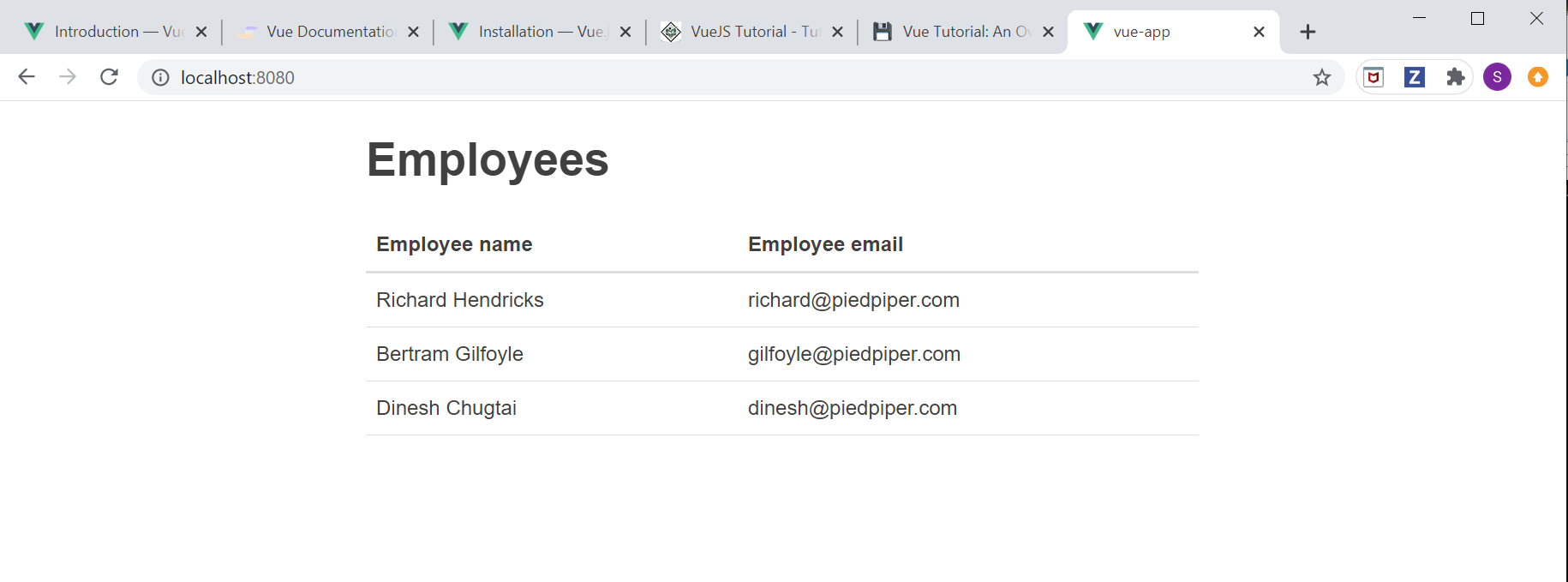


**Figure 3. Initial view of webpage on localhost:8080**

Next I added Vue js dev tools extension to chrome:

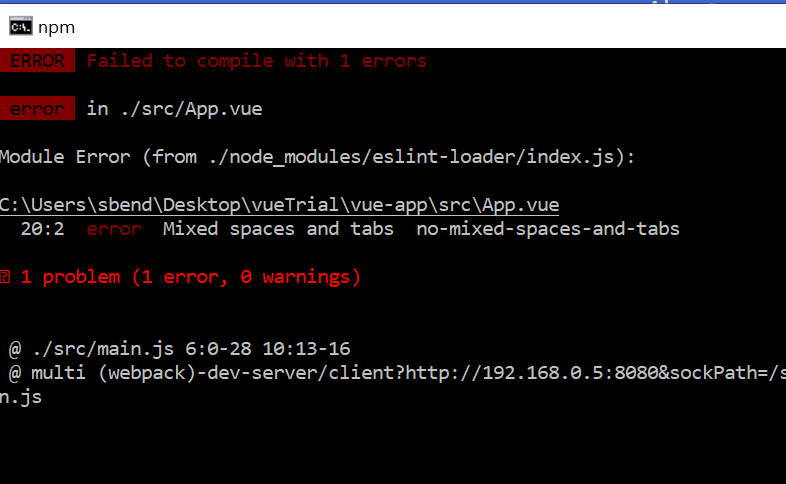
<https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd/related?hl=en>

Following her tutorial, I created a .vue file to hold a static table of employees. Then I updated app.vue to import this file. It’s interesting that html, js, and style (essentially css) is all in the same .vue file.



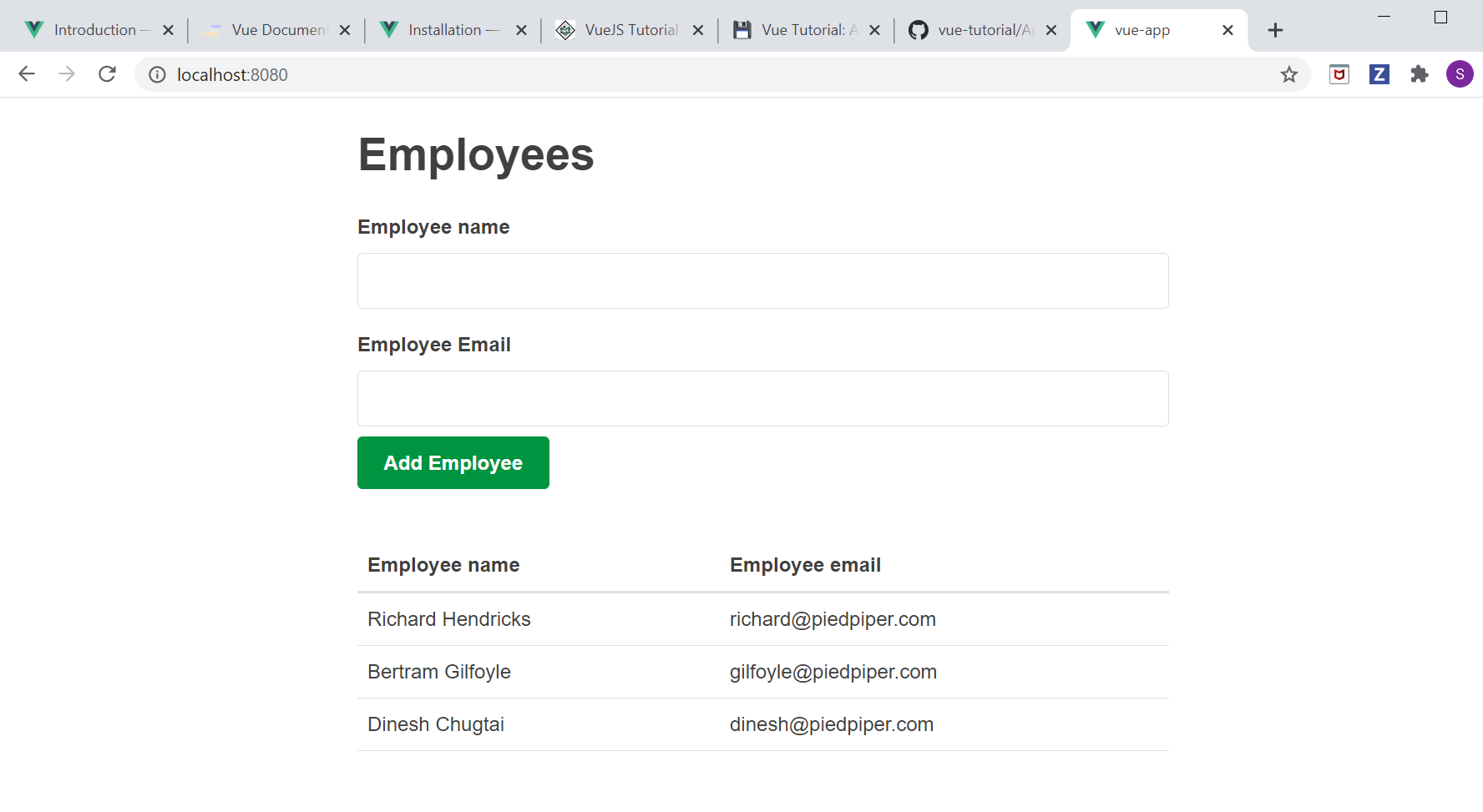
**Figure 4. Static table now appears on the page.**

I then added code to make the Employee table using an array. Next I created an employee form. When I tried to build this, I got the following error:



**Figure 5. First error of tutorial: mixed spaces aren’t allowed!**

It turns out mixing spaces and tabs is not allowed. I went into App.vue and corrected line 20. Success! The Employee Form is now on the page:



**Figure 6. Error corrected, Employee Form appears above Employee table.**

The form is set up, but it doesn’t actually store the new employee info. To do that, we need to use ‘v-model’. In EmployeeForm.vue, instead of

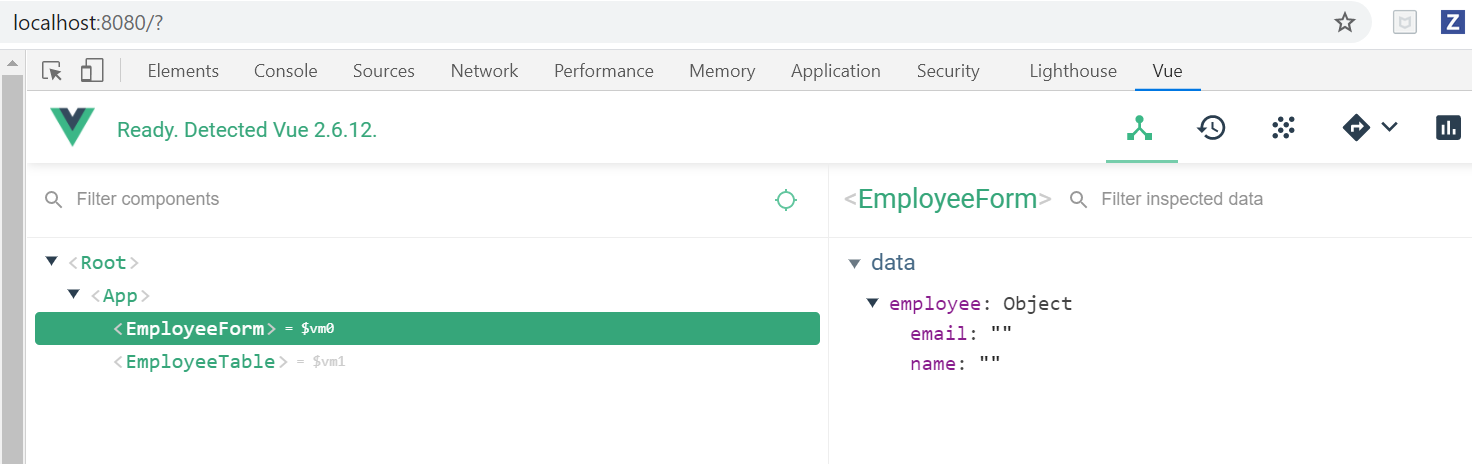
<input type="text" />

For each input type, use:

<input v-model="employee.name" type="text" />

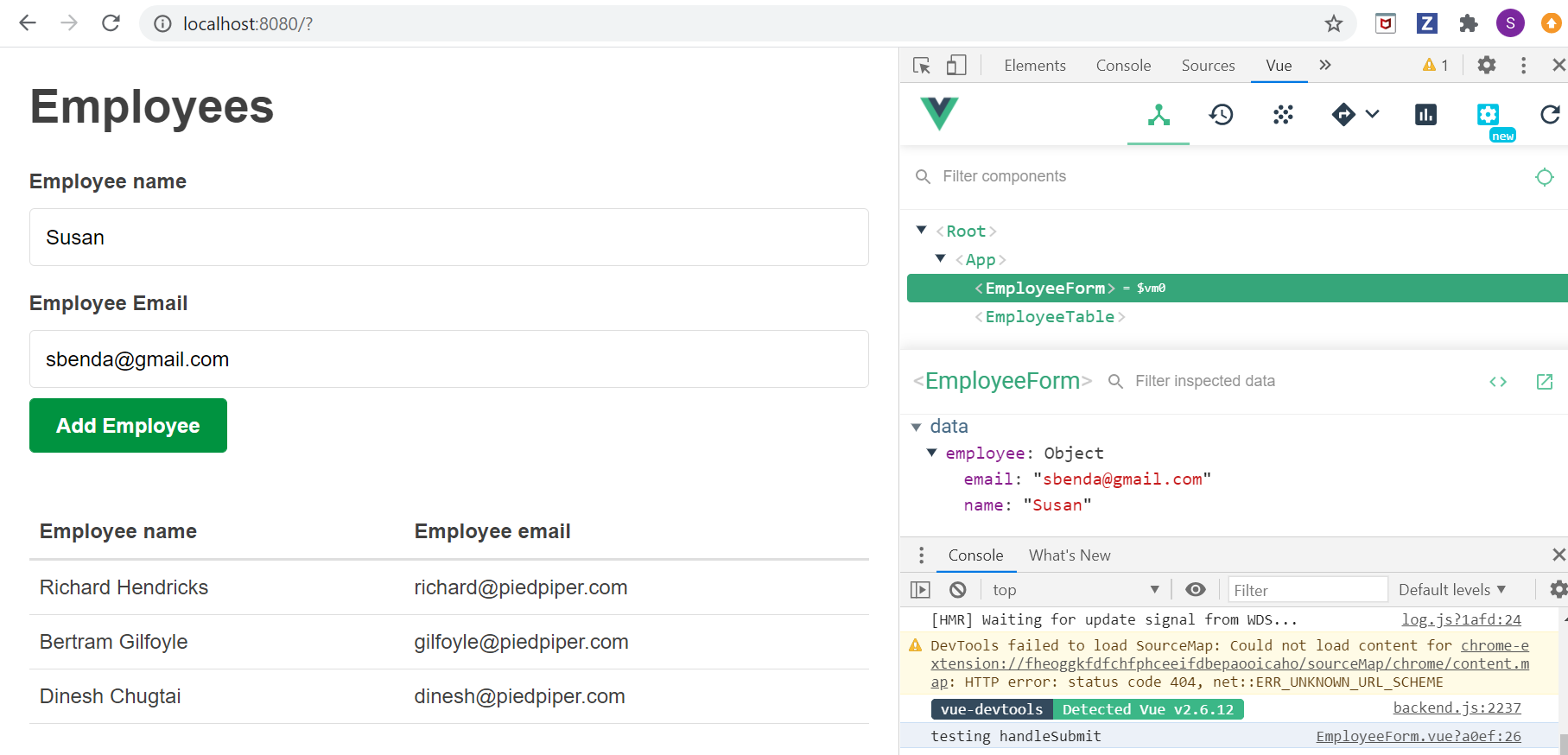
<input v-model="employee.email" type="text" />

I compiled and checked the component with Vue Dev tools (below), but I couldn’t see the difference she mentioned, where the “state of the component changes”. Mostly because I don’t know where to look.



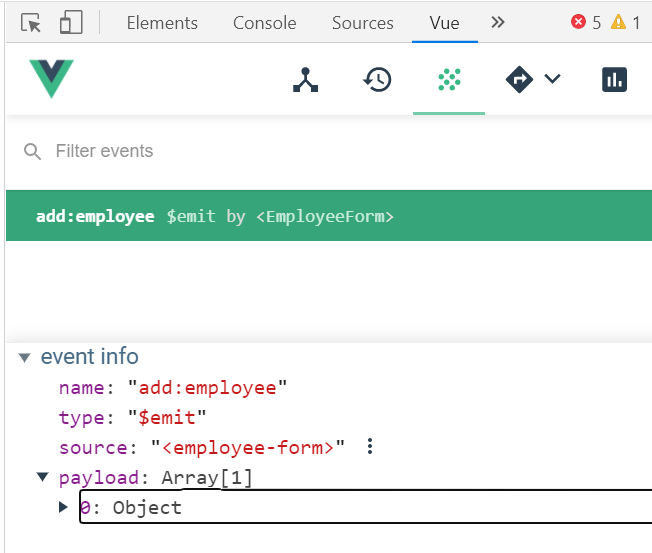
**Figure 7. Vue Dev Tools output of Employee Form.**

Next I added a method called handleSubmit; an event listener for submissions. To test it out, we had it print ‘testing handleSubmit’ to the console (bottom right of picture) when the button was clicked.



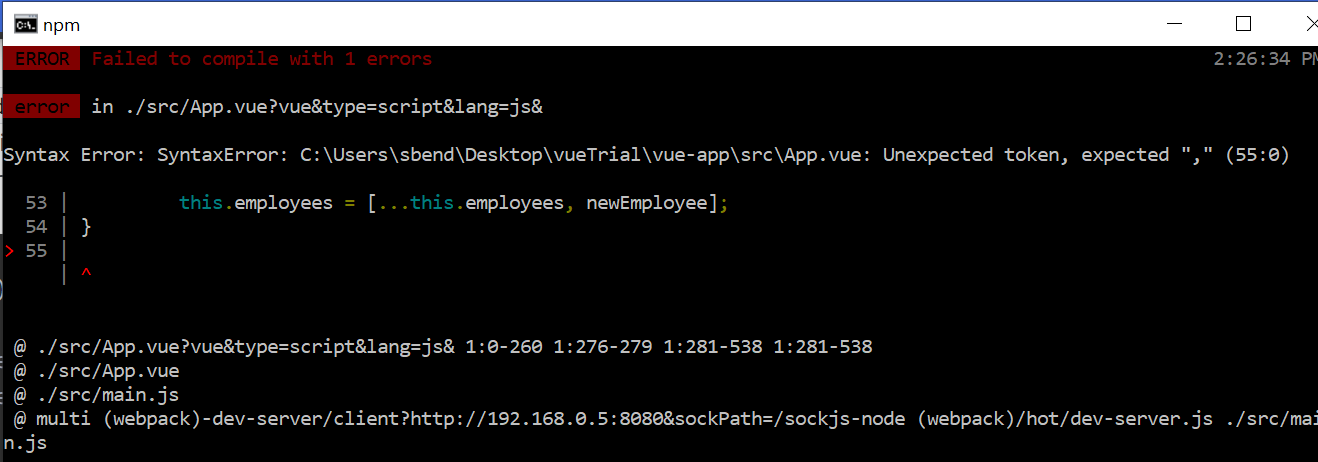
**Figure 8. Method handleSubmit prints to console**

We can pass data to App.vue using $emit. This will “broadcast a name of an event and data to its parent component”. Now when the submit button is clicked, an event of type ‘emit’ appears in the vue dev tools event section:



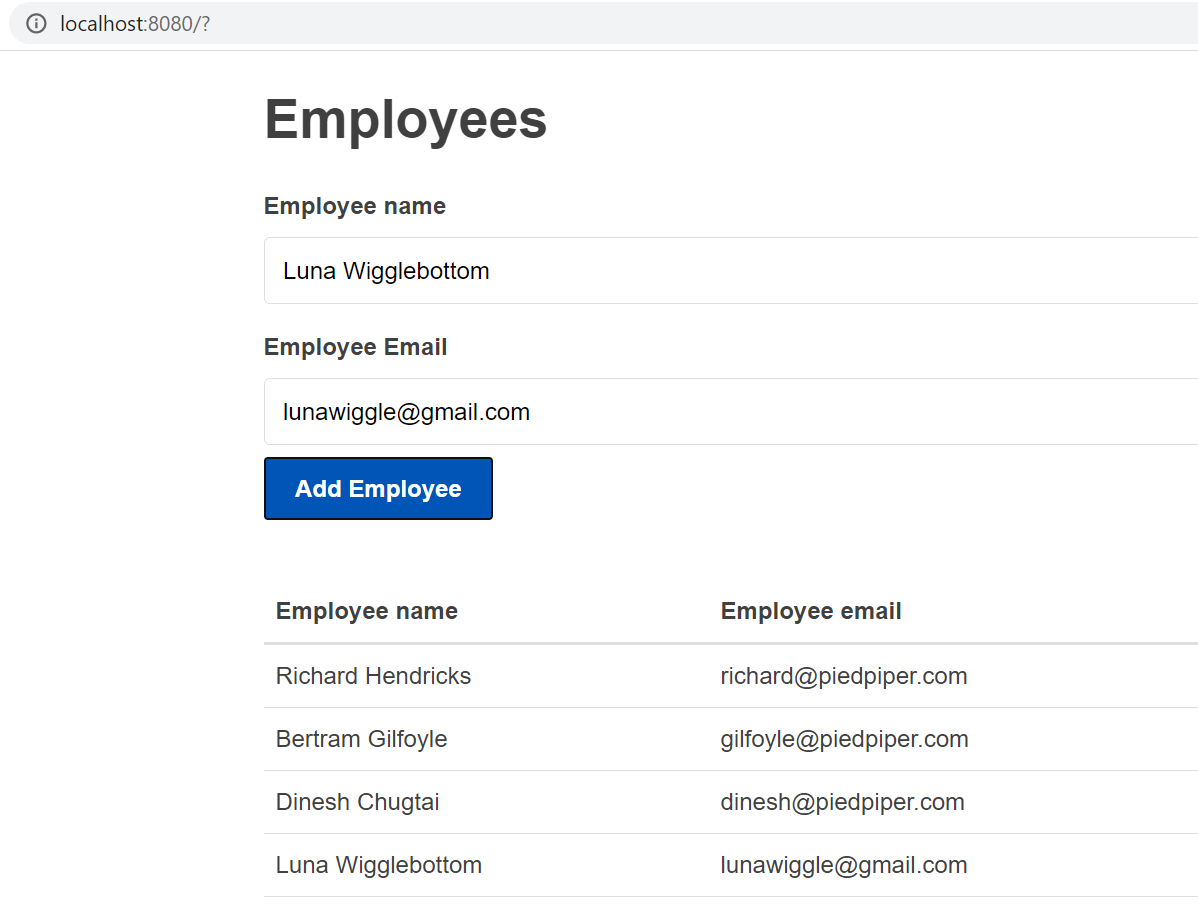
**Figure 9. Event of type ‘emit’ now appears for Employee Form.**

I created an addEmployee method in app.vue. Received a syntax error when compiling:



**Figure 10. Syntax error. There are a lot of brackets in Vue code.**

After fixing the syntax error, I was able to add an employee to the list (only on the front end, it’s not stored anywhere).

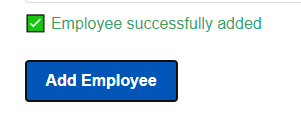
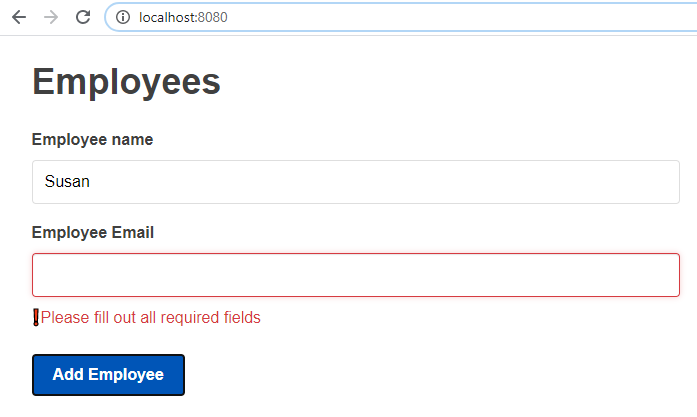


**Figure 11. New employee is added to Employee table**

Note: I had originally kept the previous ‘component :employee form’ in app.vue. This meant I ended up with two forms. So when I tried to submit, nothing happened. Once I removed the outdated component, it updated the table on the page.

## Computed properties

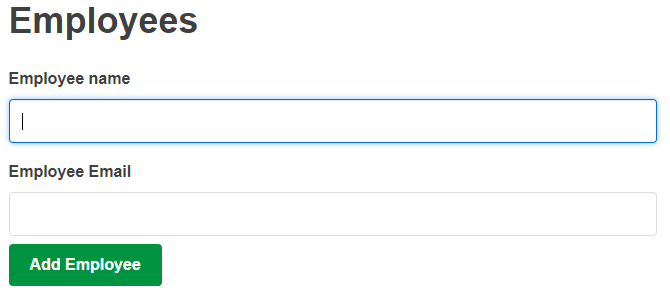
Added some form validation to confirm a successful and unsuccessful entry.



**Figure 12. Form validation added**

Conditionals in vue: can use v-if, v-else-if and v-else to get <p> fields to appear on page.

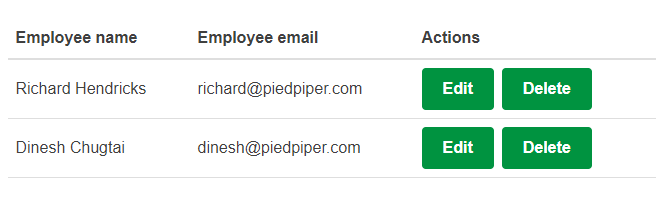
References: Can use these to highlight a field. Example: after successful entry, draw focus back to Employee Name field:



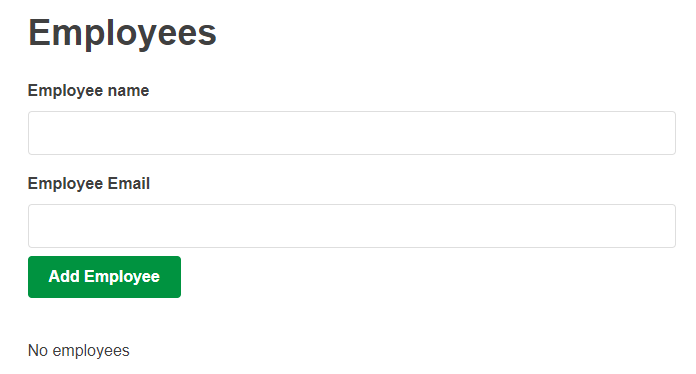
**Figure 13. Employee added, focus goes back to Employee Name**

## Deleting Items:

V-if, v-else is useful here. If there are not employees, don’t display the table. :)



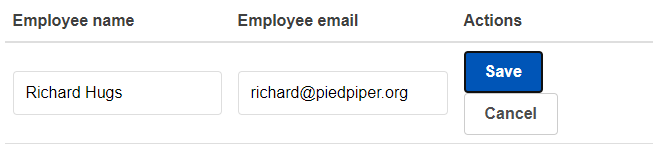
**Figure 14. Added Action Column, Delete button works**

****

**Figure 15. If there are no employees, don’t display the table.**

## Editing:

For some reason, my edit buttons aren’t spaced correctly. Will need to troubleshoot. Otherwise, this is similar to delete buttons; use conditionals to determine what to display. A new feature is the use of a cache to hold previous value in case cancel is pressed.



**Figure 16. Edit buttons worked, but spacing is off.**

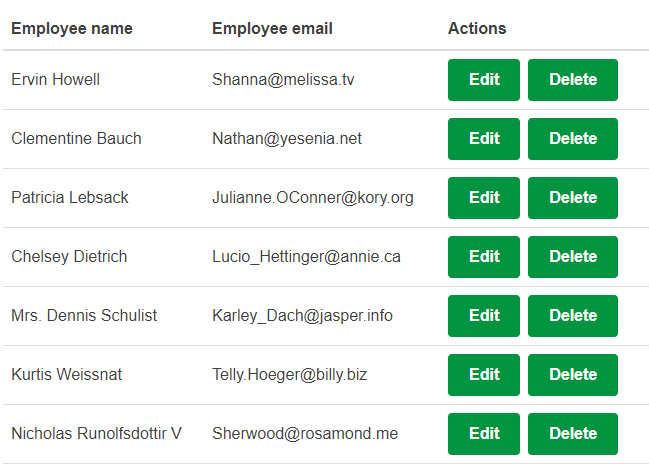
## Asynchronous REST API Call:

<https://jsonplaceholder.typicode.com/> This website has placeholder data we can use to test out our API calls. Awesome!

<https://code.tutsplus.com/tutorials/code-your-first-api-with-nodejs-and-express-understanding-rest-apis--cms-31697> More info about rest APIs.

I think this is what we would use to make calls to our database? Uses get, post, put, delete.

Once the async calls are added, we get the user data from jsonplaceholder website. Neat.



**Figure 17. Pulling data from placeholder website.**

## Deploying to GitHub Pages:

Something we could consider doing for our project, although I think Heroku is sufficient. It would be similar to our username.github.io page, but for a project: username.github.io/repoName.

Good to know, but I don’t think we’ll use it.