**Section 12 – Sending HTTP Request**

Chapter 148 – Starting App And Why We Need A Backend

1. To save data and it will not disappear after reloading the page.

Chapter 149 – Adding A Backend

1. Using Firebase Console from Google for practice, using in Develop, Database.

Chapter 150 – How To (Not) Send HTTP Requests

1. Using fetch

fetch('https://vue-http-demo-a1d4c.firebaseio.com/');

Chapter 151 – Sending A POST Request To Store Data

1. Comment all the method and to this.

fetch('https://vue-http-demo-a1d4c.firebaseio.com/surveys.json',

          {

            method: 'POST',

            headers: {

              'Content-Type': 'application/json'

            },

            body: JSON.stringify({

                name: this.enteredName,

                rating: this.chosenRating,

              }),

          }).then(response => {

            console.log("THIS SHOULD HAVE DATA", response.data);

            this.result = response.data

          }).catch(error => {

            console.log('fetch',error)

          });

Chapter 152 – HTTP Requests And HTTP Methods (Verbs)

* In the last lecture, we sent a POST request to a REST API.
* What is that? A POST request? And a REST API?
* There are different "kinds" of Http requests you could say - defined by the method (POST, GET, DELETE, ...) you attach to them (via the "method" you define on an outgoing request).
* And the server to which you're sending those requests may then react in which ever way it is configured to react to incoming requests with different methods.
* It may store data in a database for an incoming POST request, it may fetch data for a GET request.
* Typically, servers are built to work as a "REST API" - that means they have clearly defined "endpoints" (URL + Http method combinations) for which they do different things.

Chapter 153 – Using Axios Instead Of “fetch()”

* In this course, we use the native fetch() API for sending Http requests. It's built into the browser and hence we don't have to install any extra package to use it.
* If you prefer third-party libraries like **Axios** (<https://github.com/axios/axios>) you can of course also use such libraries though.
* For example, you could replace the fetch() code from the last lecture with this Axios code:
* Instead of:

fetch('https://vue-http-demo-85e9e.firebaseio.com/surveys.json', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

name: this.enteredName,

rating: this.chosenRating,

}),

});

you can write this code with Axios:

import axios from 'axios'; // at the start of your <script> tag, before you "export default ..."

...

axios.post('https://vue-http-demo-85e9e.firebaseio.com/surveys.json', {

name: this.enteredName,

rating: this.chosenRating,

});

* As you can see, with Axios, you have to **write less code**. It automatically sets the Content-Type header for you, it also automatically converts the body data to JSON.
* **BUT - as a downside - you have to add an extra package**, which ultimately increases the size of the web app you're shipping in the end.
* Later in the module, we'll also start reacting to the response returned by the request.
* fetch() returns a Promise, hence we can use then(), catch() and async/ await there. For Axios, it's just the same - it also returns a Promise.

Chapter 154 – Getting Data (GET Request) And Transforming Response Data

1. To load data from firebase

loadExperiences() {

      fetch('https://vue-http-demo-a1d4c.firebaseio.com/surveys.json')

        .then((response) => {

          if (response.ok) {

            return response.json();

          }

        }).then((data) => {

          // console.log(data); // JavaScript object

          const results = [];

          for (const id in data) {

            results.push({

              id: id,

              name: data[id].name,

              rating: data[id].rating,

            });

          }

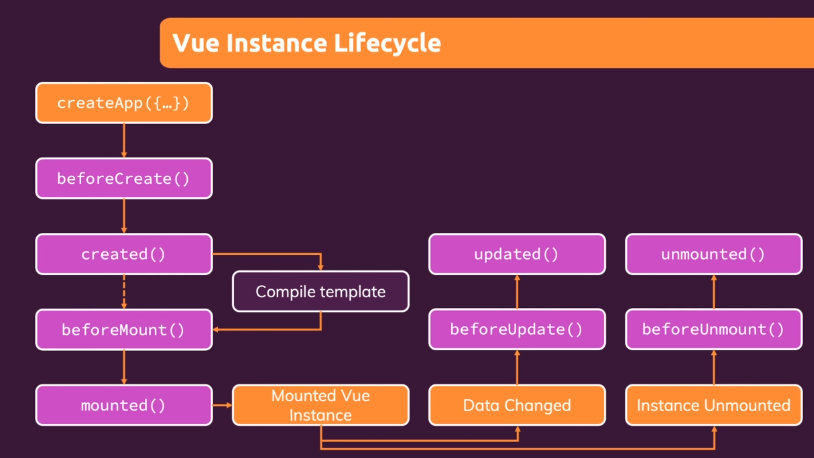
          this.results = results;

        });

    },

Chapter 155 – Loading Data When A Component Mounts

1. Every component follows the Vue Instance Lifecycle like below.



Therefore, if we want to load at the beginning when we want to access the page, we can add this code under the methods.

mounted() {

    this.loadExperiences();

  }

Chapter 156 – Showing A “Loading…” Message

1. Use this parameter

<p v-if="isLoading">Loading ...</p>

Chapter 157 – Handling The “No Data” State

1. Use this parameter

<p v-else-if="!isLoading && (!results|| results.length === 0)"> No stored experiences found. Start adding some survey results first.</p>

Chapter 158 – Handling Technical / Browser - Side Errors

1. For example we download file without .json, we need to handle this error like below.

methods: {

    loadExperiences() {

      this.isLoading = true;

      this.error = null;

      fetch('https://vue-http-demo-a1d4c.firebaseio.com/surveys.json')

        .then((response) => {

          if (response.ok) {

            return response.json();

          }

        }).then((data) => {

          // console.log(data); // JavaScript object

          this.isLoading = false;

          const results = [];

          for (const id in data) {

            results.push({

              id: id,

              name: data[id].name,

              rating: data[id].rating,

            });

          }

          this.results = results;

        }).catch((error) => {

          console.log(error);

          this.isLoading = false;

          this.error = 'Failed to fetch data - please try again later.';

        });

    },

  },

The tag HTML will be like below.

<p v-if="isLoading">Loading ...</p>

      <p v-else-if="!isLoading && error">{{ error }}</p>

      <p v-else-if="!isLoading && (!results|| results.length === 0)">No stored experiences found. Start adding some survey results first.</p>

      <ul v-else>

        <survey-result

          v-for="result in results"

          :key="result.id"

          :name="result.name"

          :rating="result.rating"

        ></survey-result>

      </ul>

Chapter 159 –Handling Error Responses

1. Handling when posting data and the result is error. If the file is not .json

this.error = null;

fetch('https://vue-http-demo-a1d4c.firebaseio.com/surveys.json',

{

method: 'POST',

headers: {

          'Content-Type': 'application/json'

},

body: JSON.stringify({

            name: this.enteredName,

            rating: this.chosenRating,

}),

}).catch(error => {

console.log(error);

this.error = 'Something went wrong - try again later !';

});

1. Handling when posting data but the format is not Json, but JavaScript.

Success code 200 / 201. This error code is 400 / 500. Add then before catch.

this.error = null;

fetch('https://vue-http-demo-a1d4c.firebaseio.com/surveys.json',

{

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({

name: this.enteredName,

rating: this.chosenRating,

}),

}).then(response => {

if (response.ok) {

} else {

throw new Error('Could not save data !');

}

}).catch(error => {

console.log(error);

this.error = error.message;

});

this.enteredName = '';

this.chosenRating = null;

},

Chapter 160 – Module Summary

1. Learn HTTP request both to send data and to get data. How to work with response data, how to extract it, and how to show it.
2. You learn how to show loading text or a loading spinner.
3. You learn how to handle error in case something goes wrong.
4. Communicate with backend.