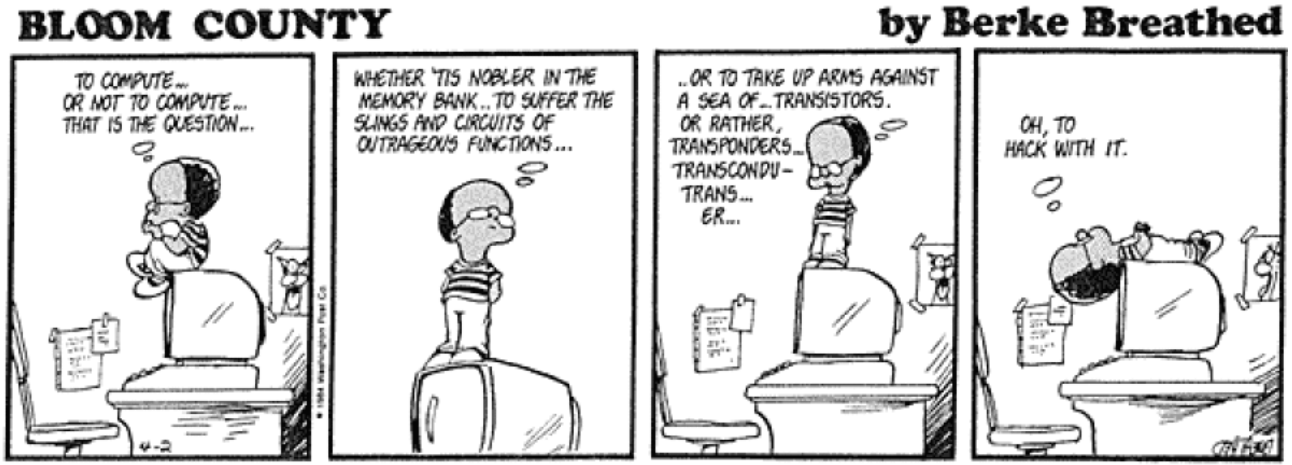

BLOOM COUNTY
by Berke Breathed
TO COMPUTE... OR NOT TO COMPUTE... THAT IS THE QUESTION...
WHETHER 'TIS NOBLER IN THE MEMORY BANK..TO SUFFER THE SLINGS AND CIRCUITS OF OUTRAGEOUS FUNCTIONS...
..OR TO TAKE UP ARMS AGAINST A SEA OF...TRANSISTORS. OR RATHER, TRANSPONDERS... TRANSCONDU- TRANS... ER...
OH, TO HACK WITH IT.


My code never has bugs
It just develops random unexpected features.

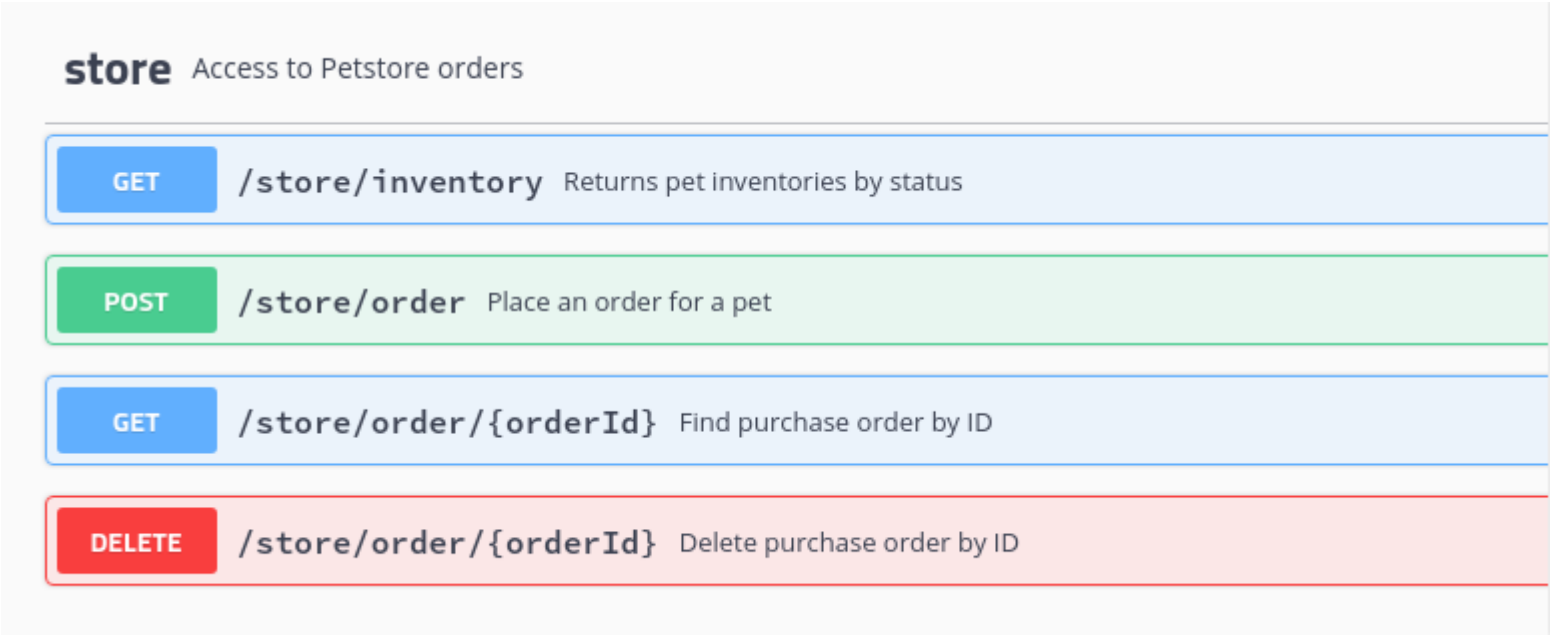
Web Services (POST and Error Handling) in VUE

Objectives

- Make POST, PUT, and DELETE requests using Axios
- Catch and handle errors using Axios
- Explain what Cross-Origin Resource Sharing is and how it works
- Explain the cross-origin request process
- Explain why developers might encounter CORS errors when testing APIs on localhost

POST, PUT, & DELETE

HTTP Methods



Axios Post

```
axios
  .post("https://pokeapi.co/api/v2/pokedex/kanto/")
  .then(response => {
    this.pokemon = response.data;
  })
  .catch(err => {
    console.log(err);
  });
```

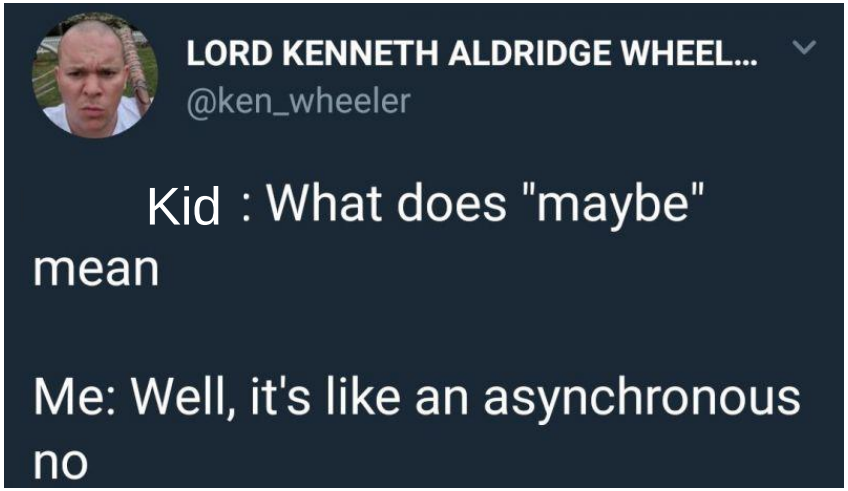
Axios Put

```
let userData = {
  firstName: 'Fredrick',
  lastName: 'Smith'
};

axios.put('/users/23', userData)
  .then((response) => {
    console.log(response);
  });
```

Axios Delete

```
axios.delete(process.env.VUE_APP_BASE_URL + 'api/users/1/', config)
  .then(function (response) {
    console.log(response)
  })
  .catch(function (error) {
    console.log(error)
  })
```


I Am Devloper
@iamdevloper
Roses are red
And so are you
Violets are blue
Asynchronous operations
are great
15:53 · 14 Feb 19 · Twitter Web App
1,993 Retweets 6,207 Likes


LORD KENNETH ALDRIDGE WHEEL...
@ken_wheeler
Kid : What does "maybe" mean
Me: Well, it's like an asynchronous no

Axios Verbs

```
getItem(id) {
  return http.get(`/items/${id}`);
},

update(myItem) {
  return http.put(`/items/${myItem.id}`, myItem);
},

create(myItem) {
  return http.post('/items', myItem);
},

delete(myItem) {
  return http.delete(`/items/${myItem.id}`);
}
```

Let’s look at the code!

Promise Error Handling

```
getAllQuestions()
  .then(response => {
    // Data is loaded from the contents of the response body
    // It's typically going to be a JavaScript object or an array of objects
    const questions = response.data;
    this.$store.commit('QUESTIONS_LOADED', questions);
  })
  .catch(error => {
    console.error('An error occurred trying to load questions', error);
  })
  .finally(() => console.log('Finally!'));
```

.catch method

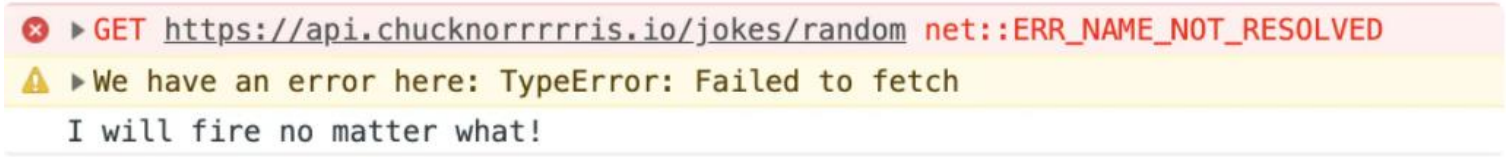
Runs if:

- Server responds with no – 2xx response code (2xx codes are “success” messages)
- Server fails to respond due to error
- Something happened that triggered an error.

.finally method

- Runs ALWAYS

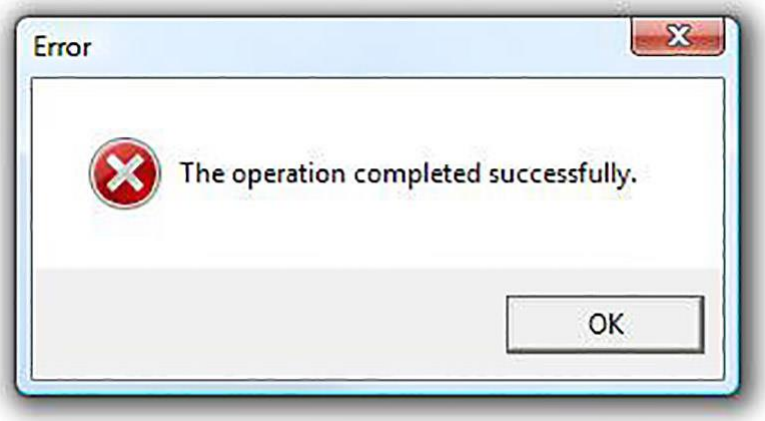
```
try { // let's mess this up a bit
   let chuckJokes =
      fetch(`https://api.chucknorrrrris.io/jokes/random`)
        .then(res => res.json());

      console.log('I have some data for you!');
      document.getElementById("quote").innerHTML = chuckJokes.value;
   }
   catch(error) {
      console.warn(`We have an error here: ${error}`)
   }
   finally {
      console.log('I will fire no matter what!')
   }
}
```



Services

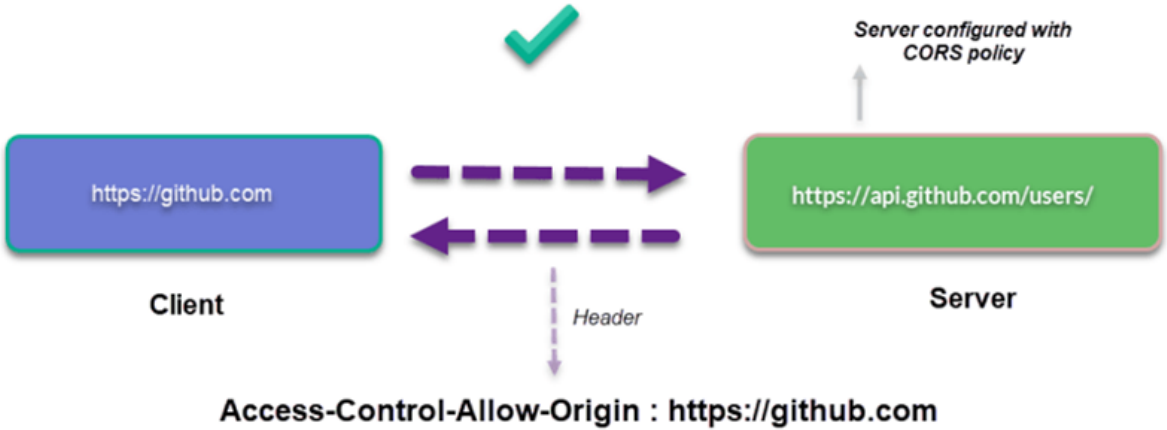
```
import axios from 'axios';

// Create our Axios instance used to communicate with the server
const http = axios.create({
  baseURL: 'https://some.url.net'
});

export default { // This object is what other files will import via the import keyword

    getAllItems() {
      return http.get('/items'); // This is added to the end of baseURL specified above
    },

    getItem(id) {
      return http.get(`/items/${id}`);
    },

    update(myItem) {
      return http.put(`/items/${myItem.id}`, myItem);
    },

    create(myItem) {
      return http.post('/items', myItem);
    },

    delete(myItem) {
      return http.delete(`/items/${myItem.id}`);
    }

  };
```

Let’s Code!

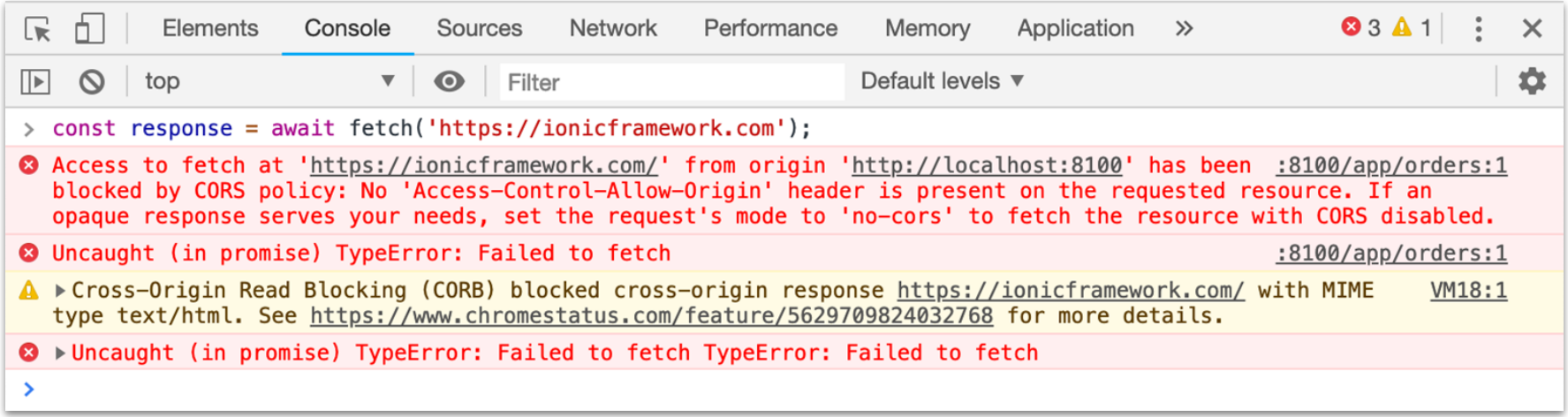


Your code can't crash if you never run it

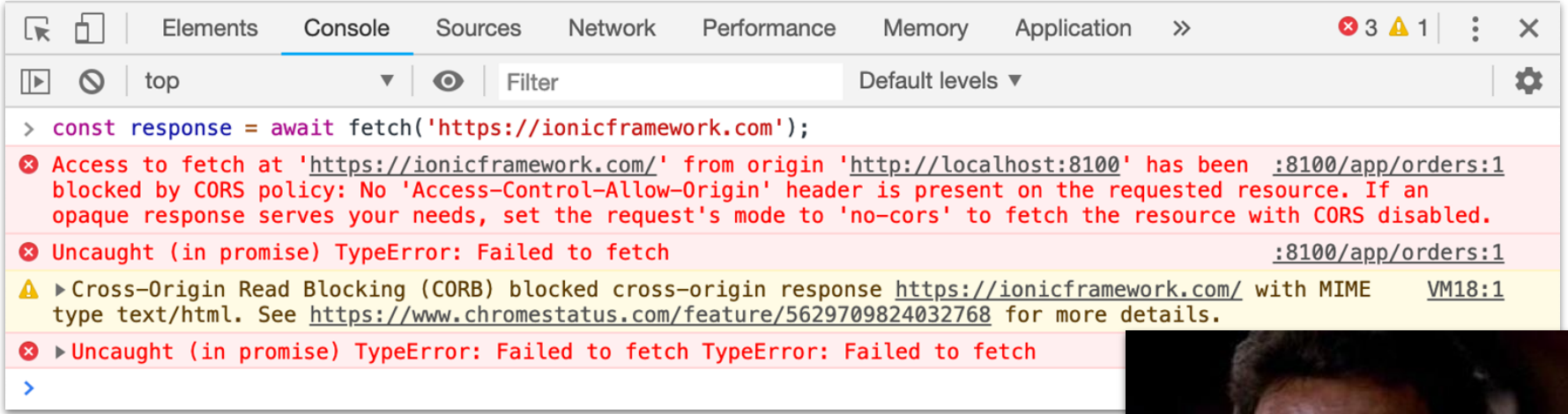
But Wait, There's More!



BUT WAIT, THERE'S MORE!



BUT WAIT, THERE'S MORE!

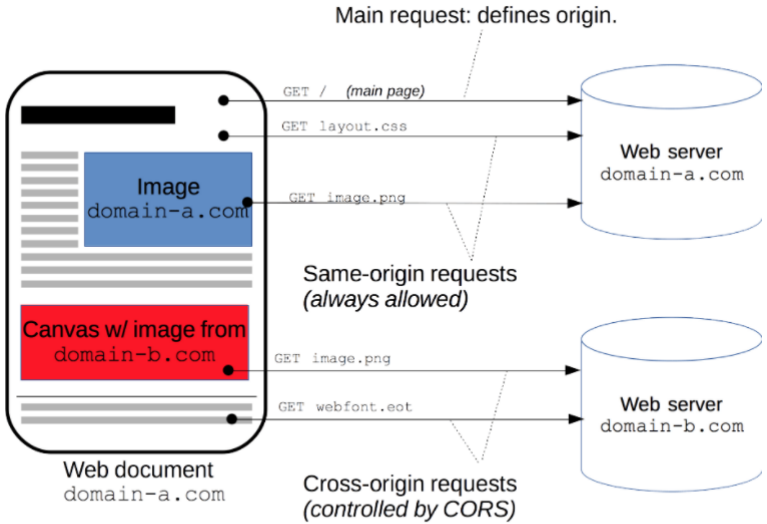


Literally any HTTP error

CORS errors

Handling CORS on Java Backend

```
3  import javax.validation.Valid;
23
24 @RestController
25 @CrossOrigin
26 public class AuthenticationController {
27
28     private final TokenProvider tokenProvider;
29     private final AuthenticationManagerBuilder authenticationManagerBuilder;
30     private UserDAO userDAO;
31
```



The CORS mechanism supports secure cross-origin requests and data transfers between browsers and servers. Modern browsers use CORS in APIs such as `XMLHttpRequest` or Fetch to mitigate the risks of cross-origin HTTP requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/

Objectives

- Make POST, PUT, and DELETE requests using Axios
- Catch and handle errors using Axios
- Explain what Cross-Origin Resource Sharing is and how it works
- Explain the cross-origin request process
- Explain why developers might encounter CORS errors when testing APIs on localhost