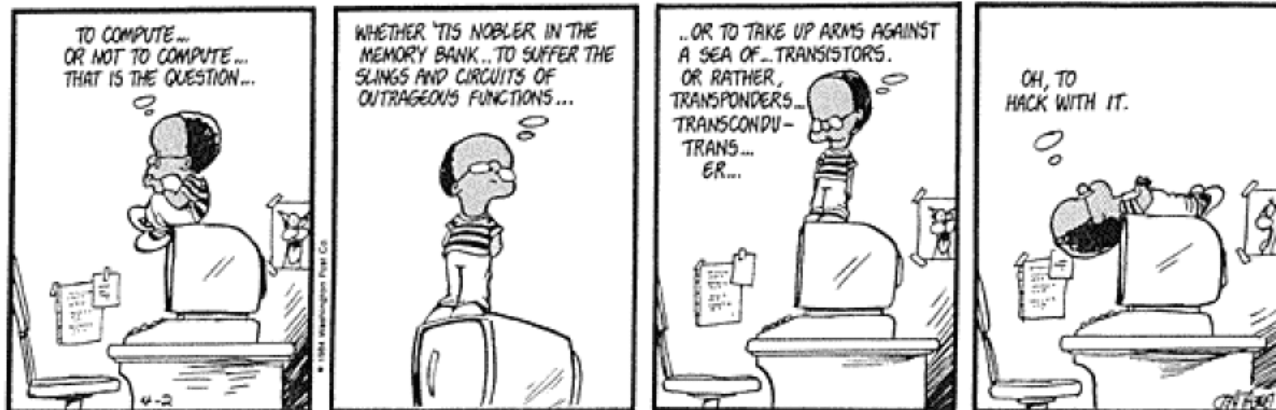


BLOOM COUNTY

by Berke Breathed



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

store Access to Petstore orders

GET

/store/inventory Returns pet inventories by status

POST

/store/order Place an order for a pet

GET

/store/order/{orderId} Find purchase order by ID

DELETE

/store/order/{orderId} Delete purchase order by ID

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)
  })
```




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!



**99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.**

127 little bugs in the code...

Promise Error Handling

```
1 getAllQuestions()  
2   .then(response => {  
3     // Data is loaded from the contents of the response body  
4     // It's typically going to be a JavaScript object or an array of objects  
5     const questions = response.data;  
6     this.$store.commit('QUESTIONS_LOADED', questions);  
7   })  
8   .catch(error => {  
9     console.error('An error occurred trying to load questions', error);  
10  })  
11  .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.chucknorrrrrris.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!`)
}
}
```

✖ ▶ GET <https://api.chucknorrrrrris.io/jokes/random> net::ERR_NAME_NOT_RESOLVED

⚠ ▶ We have an error here: TypeError: Failed to fetch

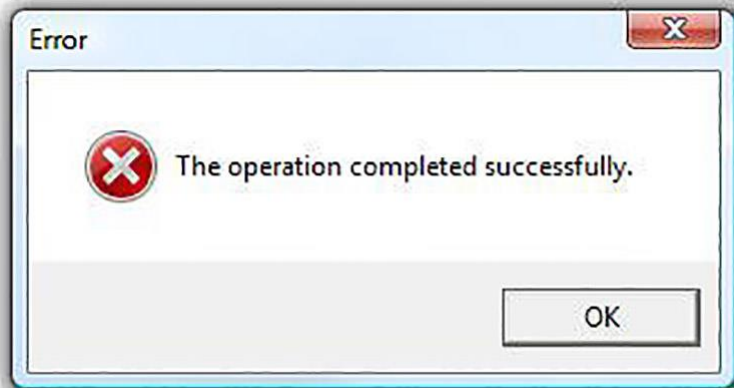
I will fire no matter what!

Services

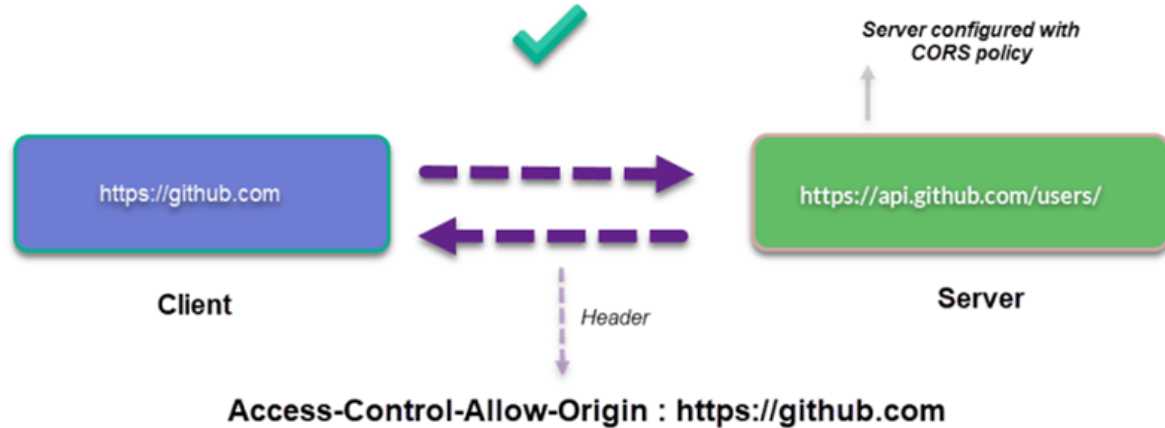
```
1 import axios from 'axios';
2
3 // Create our Axios instance used to communicate with the server
4 const http = axios.create({
5   baseURL: 'https://some.url.net'
6 });
7
8 export default { // This object is what other files will import via the import keyword
9
10   getAllItems() {
11     return http.get('/items'); // This is added to the end of baseURL specified above
12   },
13
14   getItem(id) {
15     return http.get(`/items/${id}`);
16   },
17
18   update(myItem) {
19     return http.put(`/items/${myItem.id}`, myItem);
20   },
21
22   create(myItem) {
23     return http.post('/items', myItem);
24   },
25
26   delete(myItem) {
27     return http.delete(`/items/${myItem.id}`);
28   }
29
30 };
```

Let's Code!

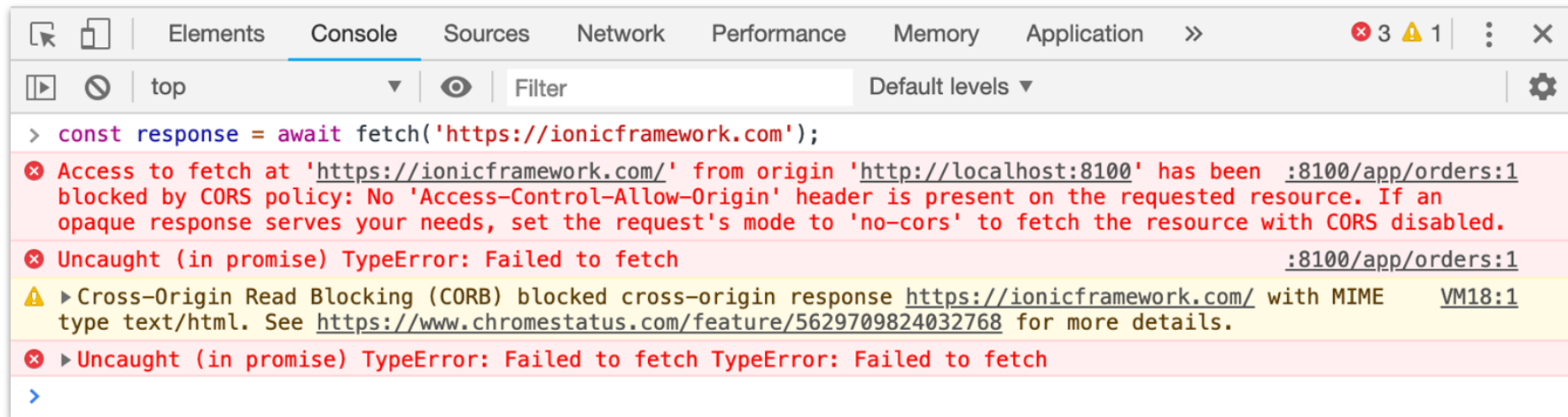
Your code can't crash if
you never run it



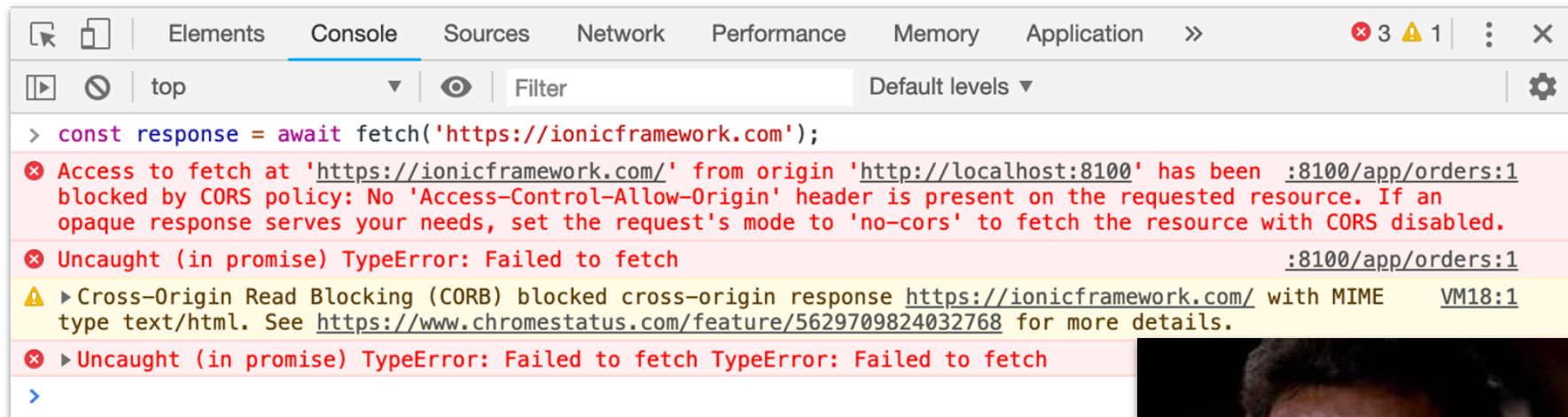
But Wait, There's More!



BUT WAIT, THERE'S MORE!



BUT WAIT, THERE'S MORE!






Literally
any HTTP
error

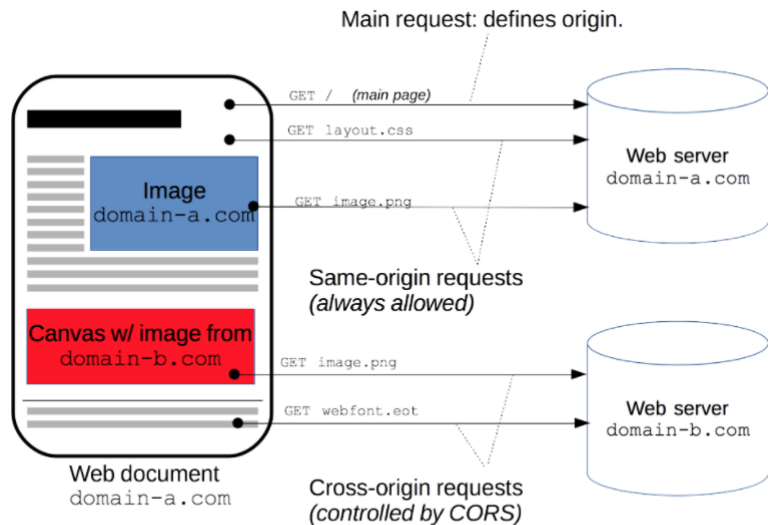


CORS
errors

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