Async/Await, Fetch, Cross-Origin Resource Sharing

CS 390 – Web Application Development

J. Setpal

September 19, 2022

Table of Contents

Table of Contents

WIWYT – Asnyc/Await

- On Tuesday, Arnav discussed asynchrony through callbacks and promises.

WIWYT – Asnyc/Await

- On Tuesday, Arnav discussed asynchrony through callbacks and promises.
- Callbacks "suck" (read: tedious, unweildy).

WIWYT – Asnyc/Await

- On Tuesday, Arnav discussed asynchrony through callbacks and promises.
- Callbacks "suck" (read: tedious, unweildy).
- Promises are cool!

WIWYT – Asnyc/Await

- On Tuesday, Arnav discussed asynchrony through callbacks and promises.
- Callbacks "suck" (read: tedious, unweildy).
- Promises are cool!
- Async/Await allows us to develop well-structured and maintainable asynchronous code without worrying about callbacks' quirks.

WIWYT – Fetch

- Fetch is a versatile method for making network requests.
- It enables client-server interaction via AJAX (**A**synchronous **J**avaScript **A**nd **X**ML) queries.

WIWYT – Fetch

- Fetch is a versatile method for making network requests.
- It enables client-server interaction via AJAX (**A**synchronous **J**avaScript **A**nd **X**ML) queries.
- Bonus: It's a wonderful implementation of promises!

WIWYT – Cross-Origin Resource Sharing

- Cross-Origin Resource Sharing is a policy enabling different remotes to share resources.

WIWYT – Cross-Origin Resource Sharing

- Cross-Origin Resource Sharing is a policy enabling different remotes to share resources.
- By default, resource sharing is blocked as a security measure.

WIWYT – Cross-Origin Resource Sharing

- Cross-Origin Resource Sharing is a policy enabling different remotes to share resources.
- By default, resource sharing is blocked as a security measure.
- Understanding CORS policy allows us to navigate it cleverly, so as to use legitimate resources **without undermining** existing security measures.

Table of Contents

Async – General Idea, Syntax

Async: Functions defined with the `async` keyword wrap the returned value of the function in a promise.

```
async function foo() {
        // ... code that takes a long time to run
        return true;
}
```

Async – General Idea, Syntax

Async: Functions defined with the `async` keyword wrap the returned value of the function in a promise.

```
async function foo() {
        // ... code that takes a long time to run
        return true;
}
```

We interact with an `async` function exactly like a promise variable. For our above example:

```
foo().then(
        function(result) // .. handle a resolution
        function(error) // .. handle a rejection
);
```

Async – General Idea, Syntax

Async: Functions defined with the `async` keyword wrap the returned value of the function in a promise.

```
async function foo() {
        // ... code that takes a long time to run
        return true;
}
```

We interact with an `async` function exactly like a promise variable. For our above example:

```
foo().then(
        function(result) // .. handle a resolution
        function(error) // .. handle a rejection
);
```

However, we can simplify this even further!

Await – General Idea, Syntax

Await: Works only within an `async` function. Halts execution until the promise is settled.

```
async function foo() {
        // ... code to define a promise
        let value = await promise;
        // ... code to handle the output value
}
```

Await – General Idea, Syntax

Await: Works only within an `async` function. Halts execution until the promise is settled.

```
async function foo() {
        // ... code to define a promise
        let value = await promise;
        // ... code to handle the output value
}
```

The main advantage being, we can call it like so:

```
foo(); // and everything runs perfectly!
```

Table of Contents

Fetch – General Idea, Syntax

Fetch: A versatile method used to interface with data on HTTP(s) remotes. It returns promises, so we can use `await` to directly catch the end-result.

```
let response = await fetch('https://cs390.dev/');
```

:confetti: We just pulled data from a GET request on the course website!

Fetch – General Idea, Syntax

Fetch: A versatile method used to interface with data on HTTP(s) remotes. It returns promises, so we can use `await` to directly catch the end-result.

```
let response = await fetch('https://cs390.dev/');
```

:confetti: We just pulled data from a GET request on the course website!

We can read and perform manipulations on the data using methods within the response data structure:

```
if (response.ok) {
        let val = await response.json();
        // ... code to use json output
} else {
        // ... code to handle error
}
```

Some More Syntax

`fetch` is incredibly dynamic with a **HUGE** API. Here's a few important methods:

- **response.json()** – parse the response as JSON object.
- **response.text()** – parse the response as plaintext.
- **response.blob()** – parse the response as raw binary data.

Some More Syntax

`fetch` is incredibly dynamic with a **HUGE** API. Here's a few important methods:

- **response.json()** – parse the response as JSON object.
- **response.text()** – parse the response as plaintext.
- **response.blob()** – parse the response as raw binary data.

```
let requestOptions = {
      .. // add method, headers, body
};
```

Full Documentation:
`https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API`

Let's Play with Fetch

If you can view this screen, I am making a mistake.

Table of Contents

CORS – General Idea

Cross-Origin Resource Sharing.
Long, fancy name :/
Let's break it down:

CORS – General Idea

Cross-Origin Resource Sharing.
Long, fancy name :/
Let's break it down:

- **Cross-Origin:** Pertaining to varying remote sources.

CORS – General Idea

Cross-Origin Resource Sharing.
Long, fancy name :/
Let's break it down:

- **Cross-Origin:** Pertaining to varying remote sources.
- **Resource:** Since we're dealing with JavaScript, resources here are script files.

CORS – General Idea

Cross-Origin Resource Sharing.
Long, fancy name :/
Let's break it down:

- **Cross-Origin:** Pertaining to varying remote sources.
- **Resource:** Since we're dealing with JavaScript, resources here are script files.
- **Sharing:** From prior context, we're sharing scripts.

CORS – General Idea

Cross-Origin Resource Sharing.
Long, fancy name :/
Let's break it down:

- **Cross-Origin:** Pertaining to varying remote sources.
- **Resource:** Since we're dealing with JavaScript, resources here are script files.
- **Sharing:** From prior context, we're sharing scripts.

Putting it all together, the policy that governs this inter-source script sharing is the CORS policy!

The CORS Workflow

So: **why is it important?**

The CORS Workflow

So: **why is it important?**

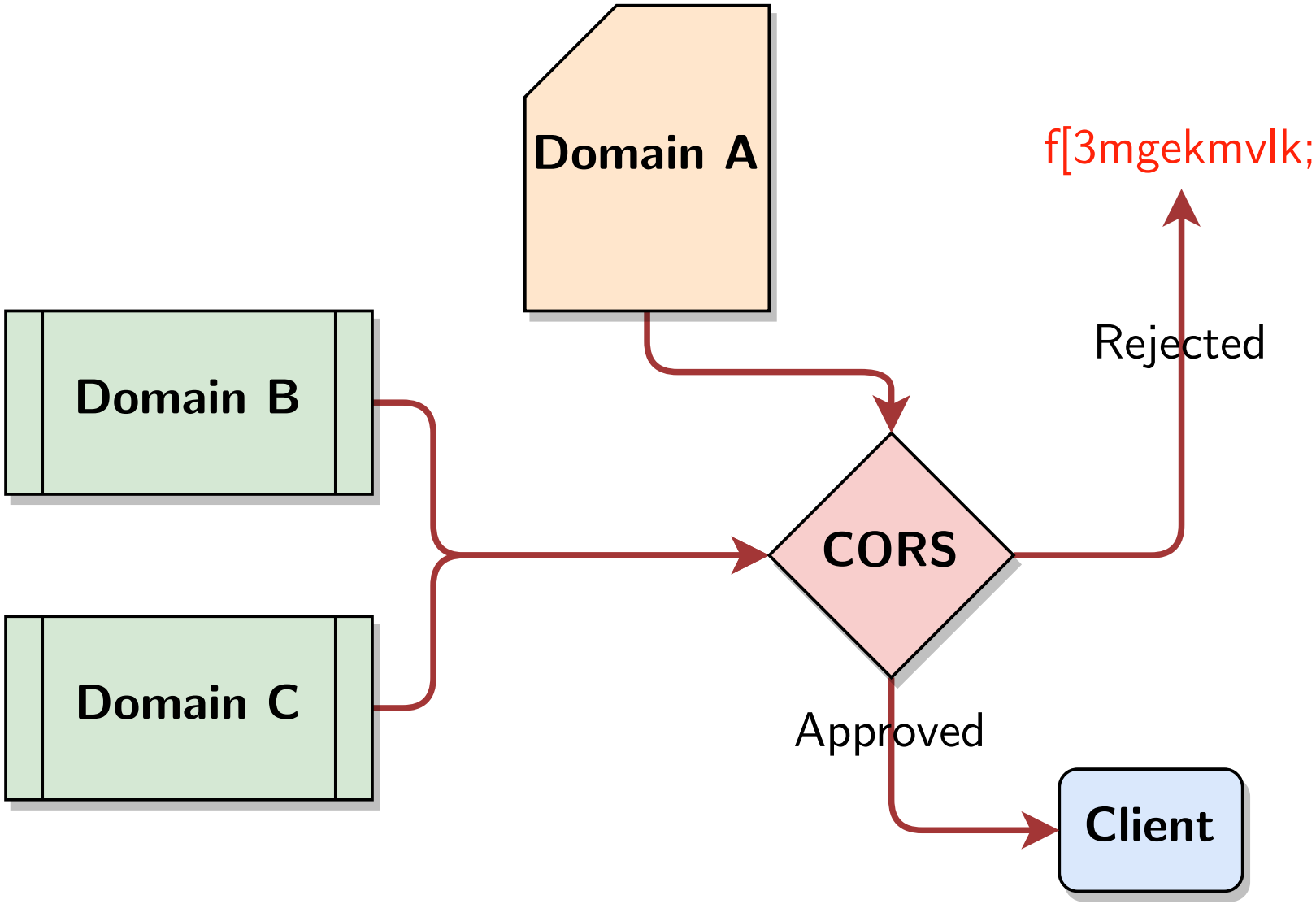

Table of Contents

Homework 4?

Homework 3 was due last Friday. So, homeworks will now be announced Wednesday!

Quiz 8 should be out now, due **Sep 20, 11:59pm**.

Thank you!

Have an awesome rest of your day!

Slides: `https://www.cs390.dev/slides/async-await.pdf`

If anything's incorrect or unclear, please ping: `jsetpal@purdue.edu`
I'll patch it ASAP.