web programming

sharing resources


oli


rec

agenda

1. RESTful
2. Static files
3. Fetch API
4. JSON
5. CORS

1. RESTful

what is REST?

- **RE**presentational **S**tate **T**ransfer
- is an **architectural style** for defining web services
- allows to **access and manipulate web resources** by using a predefined set of **stateless** operations

Resource

One of the "nouns" being served

Method

The "verbs" to act on the resources

REST Request

"Sentence" with a noun (resource) and a verb (method)

For example

GET /users/johnny

-> *Fetch user named johnny*

DELETE /books/1234

-> *Delete book with ID 1234*

HTTP METHODS

name	use
GET	**Get a single or list of resources**
HEAD	
POST	**Create a new resource**
PUT	**Replace a resource (may create a resource if it doesn’t exist)**
PATCH	**Update a resource**
DELETE	**Delete a resource**
CONNECT	
OPTIONS	
TRACE	

HTTP METHODS

name	use
GET	**Get a single or list of resources**
POST	**Create a new resource**
PUT	**Replace a resource (may create a resource if it doesn’t exist)**
PATCH	**Update a resource**
DELETE	**Delete a resource**

Endpoints

- An *endpoint* represents a resource
- The resource should be in **plural** (because it's talking with the collection), e.g. `/users`
- To identify a single entity inside the collection, we send the identity as part of the endpoint, e.g. `/users/johnny`

More examples

name	use
`GET /users`	**Get list of users**
`GET /users/johnny`	**Get the user with the ID johnny**
`PUT /users/johnny`	**Replace the user with ID johnny**
`PATCH /users/johnny`	**Update the user with ID johnny**
`DELETE /users/johnny`	**Delete the user with ID johnny**

2. static files

```
$ cd web
```

```
$ npm i --save-dev serve
```

- serve is a development utility
- allow us to serve static files without having to setup anything

web/package.json

```
{
  …,
  "scripts": {
    "start": “serve”,
    …
  },
  …,
}
```

3. Fetch API

what is Fetch API?

- interface for fetching resources
- provides `Request` and `Response` objects, as well as a global `fetch()` method
- better alternative to `XMLHttpRequest`
- allows us to do AJAX

what is AJAX?

- **A**synchronous **J**avaScript **A**nd **X**ml
- not a technology in itself, but a **combination of technologies**
- allows a web app to make updates to the UI without reloading the browser page
- makes the app faster and more responsive to user interaction

`fetch()`

- easy way to fetch resources asynchronously across the network
- uses Promises (*)

4. JSON

what is JSON?

- **J**ava**S**cript **O**bject **N**otation
- syntax for **serializing** JavaScript values
- **to serialize** is to convert a value to a format that can be stored or transmitted, and later deserialized
- **to deserialize** is to reconstruct the original value from the serialized source

JSON.stringify()

- serializes a JavaScript value into a string:

```
const foo = {
    bar: 42,
};

const serializedFoo = JSON.stringify(foo);

console.log(serializedFoo);
```

JSON.parse()

- deserializes a string into a JavaScript value:

```
const serializedFoo = '{"bar":42}';

const foo = JSON.parse(serializedFoo);

console.log(foo);
```

why JSON?

- useful to transmit data from server to web app
- lightweight*
- built-in support in Fetch API, i.e. `response.json();`

(*): compared to alternatives such as XML

<EXERCISE>

Make a html page to fetch the api

- The web app should have a list of products and users that fetches after clicking a button

5. CORS

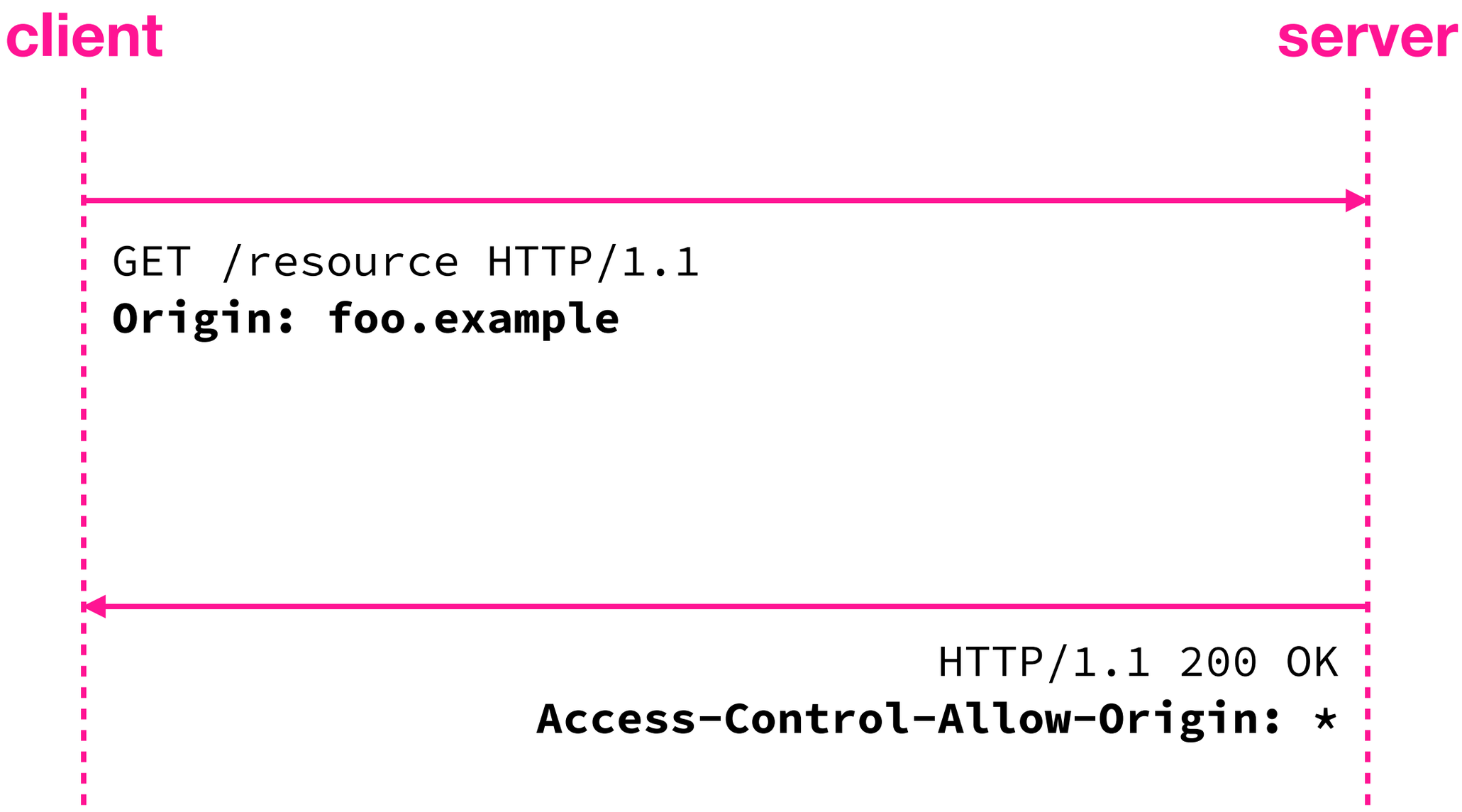
same-origin policy

- critical security mechanism
- implemented by browsers
- restricts a web app to **only access resources from the same origin the application was loaded from**
- origin = protocol + host + port

what is CORS?

- **C**ross-**O**rigin **R**esource **S**haring
- allows AJAX requests to skip the same-origin policy and **access resources from different origins**
- uses **HTTP headers**

example


client
server
GET /resource HTTP/1.1
Origin: foo.example
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *

enabling CORS in Hapi

```
const server = Hapi.server({
    port: 3000,
    host: 'localhost'
});

server.route({
    method: 'GET',
    path:’/',
    cors: {
        origin: [‘localhost’],
    },
    handler: (request, h) => {
        return { myAttribute: ‘This is an example’ };
    }
});

await server.start();
```

</EXERCISE>

Make a html page to fetch the api

thanks!