Axios: a library for Client-Server Communication

Prof. Fabio Ciravegna

Dipartimento di Informatica

Università di Torino

fabio.ciravegna@unito.it

UNIVERSITÀ DI TORINO

Axios

Rather than using Fetch for the browser and Fetch on the server, we will use Axios on both

https://axios-http.com/docs/intro

- Axios is a promise-based HTTP Client for node.js and the browser
  - It is isomorphic (= it can run in the browser and nodejs with the same codebase).
    - On the server-side it uses the native node.js http module
    - on the client (browser) it uses XMLHttpRequests (i.e. Ajax)
  - Makes XMLHttpRequests from the browser
  - Make http requests from node.js
  - Supports the Promise API
  - Intercepts request and response
  - Transforms request and response data
  - Cancels requests
  - Automatically transforms TO/FROM JSON data
- Client side support for protecting against XSRF
  - Cross-site request forgery, also known as one-click attack or session riding and abbreviated as CSRF (sometimes pronounced sea-surf[1]) or XSRF, is a type of malicious exploit of a website where unauthorized commands are submitted from a user that the web application trusts https://en.wikipedia.org/wiki/Cross-site_request_forgery

```
to add Axios to the nodes server:
                    npm install axios
(or add the line:
    "axios": "^0.26.0",
to your package.json file
(see today's lab class)
```

To use Axios on the client (browser) side: ad

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></s
```

to your html/EJS file!!

UNIVERSITÀ DI TORINO

Performing a Get

```
axios.get('/user', {
    params: {
      ID: 12345
    }
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  })
  .then(function () {
    // always executed
  });
```

These params will be received in node.js's body

Code to execute in case of success

Code to execute in case of error

Code to execute in both cases at the end

Performing a POST

```
axios.post('/user', {
    firstName: 'Fred',
    lastName: 'Flintstone'
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

These will be received in node.js's body (no need of the term "p

Code to execute in case of success

Code to execute in case of error

UNIVERSITÀ DI TORINO

Multiple parallel requests -> Promise.all

```
function getUserAccount() {
  return axios.get('/user/12345');
}

function getUserPermissions() {
  return axios.get('/user/12345/permissions');
}

Promise.all([getUserAccount(), getUserPermissions()])
  .then(function (results) {
    const acct = results[0];
    const perm = results[1];
  });
```

Define multiple functions returning an axic

Promise.all them (use an array)

This will return a sorted array of results

A catch branch should be defined to cope with errors

UNIVERSITÀ DI TORINO

Response parameter for the .then branch

```
{
  // `data` is the response that was provided by the server
  data: {},

  // `status` is the HTTP status code from the server response
  status: 200,

  // `statusText` is the HTTP status message from the server response
  // As of HTTP/2 status text is blank or unsupported.
  // (HTTP/2 RFC: https://www.rfc-editor.org/rfc/rfc7540#section-8.1.2.4)
  statusText: 'OK',

  // `headers` the HTTP headers that the server responded with
  // All header names are lower cased and can be accessed using the bracket notation.
  // Example: `response.headers['content-type']`
  headers: {},

  // `config` is the config that was provided to `axios` for the request
  config: {},

  // `request` is the request that generated this response
  // It is the last ClientRequest instance in node.js (in redirects)
  // and an XMLHttpRequest instance in the browser
  request: {}
}
```

The data

The code (always check in case

When using then, you will receive the response as follows:

```
axios.get('/user/12345')
  .then(function (response) {
    console.log(response.data);
    console.log(response.status);
    console.log(response.statusText);
    console.log(response.headers);
    console.log(response.config);
  });
```

UNIVERSITÀ DI TORINO

Response for the .catch branch

The .then branch is not shown here

```
axios.get('/user/12345')
  .catch(function (error) {
    if (error.response) {
      // The request was made and the server responded with a status code
      // that falls out of the range of 2xx
      console.log(error.response.data);
      console.log(error.response.status);
      console.log(error.response.headers);
    } else if (error.request) {
      // The request was made but no response was received
      // `error.request` is an instance of XMLHttpRequest in the browser and an instance of
      // http.ClientRequest in node.js
      console.log(error.request);
    } else {
      // Something happened in setting up the request that triggered an Error
      console.log('Error', error.message);
    }
    console.log(error.config);
  });
```

An example: the client

```
function sendAxiosQuery(url, data) {
    axios.post(url , data)
        .then (function (dataR) {
            // do something with the data
        })
        .catch( function (response) {
            alert (response.toJSON());
        })
}
```

the data is a javascript object tha
stringified automatically by axios

An example: Server to Server Comm in NodeJS

insert this code in a route

never use the parameters without checking that they are valid (I have not done it here)

```
.post(function  (req, res) {
        axios.post('http://localhost:3000/add',
              {name: req.body.name, surname: req.body.surname})
              .then(received =>
                  res.json(received.data))
              .catch(err => {
                  res.setHeader('Content-Type', 'application/json');
                  res.status(505).json(err)
              })
});
```

the result is in the data field

Always check for errors

last time we did the same with the fetch library

UNIVERSITÀ DI TORINO

What you should know

- You should be able to perform an Ajax request using Axios
- You can use Axios to replace the fetch library we used in the last lab
  - much easier!!