Fetch & Axios

CS568 – Web Application Development I

Computer Science Department

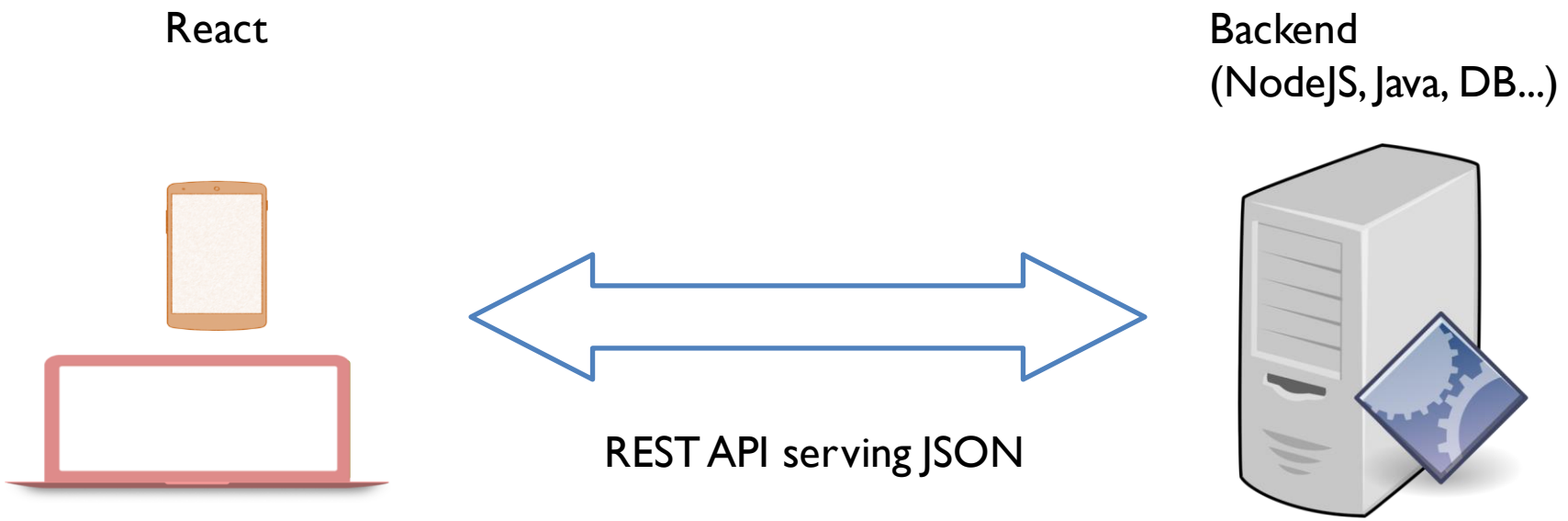
Maharishi International University

Maharishi International University - Fairfield, Iowa

Content

- Modern Web Architecture
- Fetch
- Axios

Modern WEB Architecture



REST

- REST: Representation State Transfer
- Simple HTTP client/server mechanism to exchange data
- Everything – the UNIVERSE is available through a URI
- Utilizes HTTP: GET/POST/PUT/DELETE operations

HTTP Methods for RESTful APIs

HTTP METHOD	CRUD	ENTIRE COLLECTION (E.G. /USERS)	SPECIFIC ITEM (E.G. /USERS/123)
POST	Create	201 (Created), 'Location' header with link to /users/{id} containing new ID.	Avoid using POST on single resource
GET	Read	200 (OK), list of users. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single user. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method not allowed), unless you want to update every resource in the entire collection of resource.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
PATCH	Partial Update/Modify	405 (Method not allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method not allowed), unless you want to delete the whole collection — use with caution.	200 (OK). 404 (Not Found), if ID not found or invalid.

Fetch

- The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses.
- fetch() is a global and asynchronous function
- Syntax:
  - `let promise = fetch(url, [options]);`
  - `url` – the URL to access.
  - `options` – optional parameters: method, headers etc.
  - Without `options`, this is a simple `GET` request, downloading the contents of the url.

Using fetch() - GET

- `let promise = fetch('http://localhost:3000/products')`
- The browser starts the request right away and returns a promise that the calling code should use to get the result.
- Getting a response is usually a two-stage process.
  - First, the promise, returned by fetch, resolves with an object of the built-in Response class as soon as the server responds with headers.
    - `promise.then(response => console.log(response.ok, response.status));`
  - Second, to get the response body, we need to use an additional method call.
    - response.text() – read the response and return as text,
    - response.json() – parse the response as JSON,
    - …
    - `promise.then(response => response.json())`
    - `.then(result => console.log(result));`

Using fetch() – POST with options

```
fetch('http://localhost:3000/products', {
      method: 'POST',
      body: JSON.stringify({
         title: 'cs1000',
         description: 'This is  a Demo',
         price: 100,
      }),
      headers: {
         'Content-type': 'application/json; charset=UTF-8',
      },
   })
   .then((response) => response.json())
   .then((json) => console.log(json));
```

Using Async & Await

```
async function fetchProductById(id) {
   let response = await fetch(`http://localhost:3000/products/${id}`);
   if (response.ok) {
      let json = await response.json();
      console.log(json);
   } else {
      console.log("HTTP-Error: " + response.status);
   }
}

fetchProductById(1);

// Remember  to use try/catch in case of network problem
```

Axios

Axios is a popular and third-party library that is used in JS apps to call a backend API like fetch.

Need to install before usage: npm install axios

Axios methods

This is on the client-side that you are invoking the back-end REST API.

- axios.get(URL, config)
- axios.post(URL, data, config)
- axios.put(URL, data, config)
- axios.delete(URL, config)

Axios methods

- The Axios methods have 3 parameters:
  1. URL (you can set a default URL)
  2. Data (optional) – It is available for POST, PUT, PATCH. It is not available in GET, DELETE.
  3. Options (optional). You can pass query parameters and headers to the server. In header, you pass authorization tokens if the API is secure.

Query params/URL params vs Body

Query params and body have one thing in common. Pass data to server that is required for processing the request

Query params/URL params	Body
Used for HTTP requests	Use for **POST/PUT** requests
Developers use query params in the back-end when the input is not complex. For example, you only need text inputs. Query params **don't support** complex object like **nested JSON**.	**You can pass** complex object such **nested JSON**.
URL is too verbose. That doesn't look clean.	URL is so much clean. Just domain and resource.
There is a size limit.	You can pass more data than query params.
Data is passed explicitly in the URL.	Data is passed implicitly in the body.
Values are string	Values are JSON object

Query parameters

GET /products?sort=true&origin=USA

```
async function getData(){
try {
let res = await axios.get('http://localhost:5001/products?sorted=true');
console.log(res.data);
} catch (error) {
console.log('Cannot  get data');
}
}
```

URL Parameters

DELETE /products/{id}

```
async function deleteData(){
try {
let res = await axios.delete('http://localhost:5001/products/2');
console.log(res.data);
} catch (error) {
console.log('Cannot get data');
}
}
```

Request Body

POST /products

```
async function postData(){
try {
let obj = {name: "iPhone", price: 500}
let res = await axios.post('http://localhost:5001/products', obj);
console.log(res.data);
} catch (error) {
console.log('Cannot get data');
}
}
```

Axios Global Configuration

With this global configuration, you don’t have to provide the domain name or Authorization token every time you write code for Axios operations. These are automatically injected to all requests.

```
axios.defaults.baseURL = 'http://localhost:5001';
axios.defaults.headers.common['Authorization'] = "Bearer token";
...
axios.post('/products', obj);
```

Summary

- Modern Web Architecture
- Fetch
- Axios