HTTP Requests/Ajax Calls

Lesson 8

Lesson Objectives

At the end of this module you will be able to:

- HTTP Requests in React
- Introduction of Axios package
- HTTP GET Request, fetching & transforming data
- HTTP POST, DELETE, UPDATE
- Handing Errors
- Adding/Removing Interceptors
- Creating/Using Axios intances

HTTP Requests in React

One of the most important things a developer should learn is how to (properly) debug an application.

This educates you to easily find out the cause of errors in code, but also teaches internal working of the language thus preventing future errors.

Console window is one of *the* most important tools at your disposal.

It helps as to se the state of application which we generally print using Console.log

It also shows the majority of error messages in browser caused by your app.

Disadvantages of Console.log :

when used with an object/array as a parameter can be deceiving.

```
const myObj = { name: ‘Bala‘, age:23 }
console.log(myObj);
myObj.name = 'John';
myObj.age = 22;
```

Introduction of Axios package

Data Fetching can be done by many packages like **Axios, Apollo, Relay Modern, Request, Super agent**

Axios : Promise based HTTP client for the browser and node.js.

Axios is similar to Fetch Api but has more features listed below

Features :

Make XMLHttpRequests from the browser
Make http requests from node.js
Supports the Promise API
Intercept request and response
Transform request and response data
Cancel requests
Automatic transforms for JSON data
Client side support for protecting against XSRF

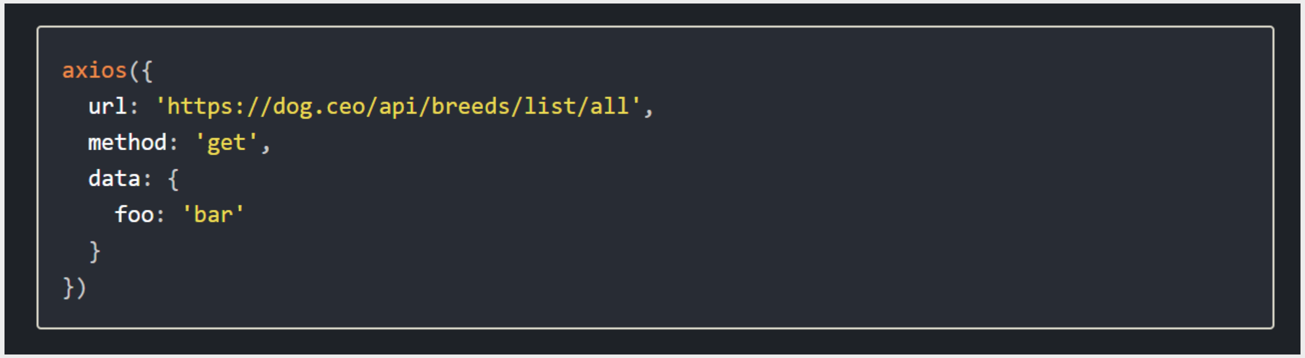
Installing axios:

We can install axios using

```
npm install axios
```

HTTP GET Request, fetching & transforming data

We can start an HTTP request from the axios object for get as shown below



```
axios({
  url: 'https://dog.ceo/api/breeds/list/all',
  method: 'get',
  data: {
    foo: 'bar'
  }
})
```

The above syntax can be made simpler also by below code snippet

```
try{
axios.get('https://api.github.com/users/maecapozzi')
}
catch (error) {
        console.error(error)
 }
```

One more important step here is we have to import axios into the project as shown below

const axios = require('axios')

HTTP POST, Delete & Update

We can start an HTTP request from the axios object for get as shown below

axios.post('https://site.com/')

An object containing the POST parameters is the second argument:

axios.post('https://site.com/', { foo: 'bar' })

For Delete request:

```
handleSubmit = event =>
{ event.preventDefault();
axios.delete(`https://jsonplaceholder.typicode.com/users/${this.state.id}`)
.then(res => { console.log(res); console.log(res.data); }) }
```

Other methods:

axios.delete(url[, config])
axios.head(url[, config])
axios.options(url[, config])
axios.put(url[, data[, config]])
axios.patch(url[, data[, config]])

Transforming Data

Axios allows you to provide functions to transform the outgoing or incoming data, in the form of two configuration options **ransformRequest** and **transformResponse**.

Both properties are arrays, allowing you to chain multiple functions that the data will be passed through.

const options = {
transformRequest: [(data, headers) =>
{ // do something with data return data; }] }

Functions can be added to transformResponse in the same way, but are called only with the response data, and before the response is passed to any chained then()callbacks.

Demo

react-axios-get-demo
react-axios-demo-crud

Handling errors

- Errors can be easily handled by using catch methods in axios http calls

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

- So in the catch , we can give our own customized message for user to understand things easily

Demo

React-debug-logical-errors

Adding/Removing Interceptors

- Transforms is used for modifying outgoing and incoming data, whereas Axios also allows you to add functions called interceptors.
- Like transforms, these functions can be attached to fire when a request is made, or when a response is received.

```
// Add a request interceptor
axios.interceptors.request.use((config) =>
 { // Do something before request is sent return config; },
(error) => { // Do something with request error return Promise.reject(error);
 }); // Add a response interceptor

axios.interceptors.response.use((response) => { // Do something with
response data return response; }, (error) => { // Do something with response
error return Promise.reject(error); });
```

- When creating interceptors, you can also choose to provide an error handler function that allows you to catch any errors
- Request interceptors can be used to do things such as retrieve a token from local storage and send with all requests.

Removing Interceptors

- Errors can be easily handled by using catch methods in axios http calls

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

If you may need to remove an interceptor later you can.

```
const myInterceptor = axios.interceptors.request.use(function () {/*...*/}); axios.i=
nterceptors.request.eject(myInterceptor);
```

Creating/Using Axios intances

Creating an instance

You can create a new instance of axios with a custom config.

axios.create([config])

const instance = axios.create({ baseURL: 'https://some-domain.com/api/', timeout: 1000, headers: {'X-Custom-Header': 'foobar'} });

Summary

- Understanding React Error Messages
- Handling Logical Errors,
- Debugging React apps using google developer tools and React DevTool
- Understanding Error Boundaries