**Axios Crash Course (Traversy Media)**

**HTTP Library/Client to make requests to your own backend or a third party API, to fetch data. Similar to fetch API built into the browser, but more powerful.**

[**https://github.com/axios/axios**](https://github.com/axios/axios)

**<script src=”https://cdnjs.cloudflare.com/ajax/libs/axios/0.19.0/axios.min.js”></script>**

**<button id=”get”>Get</button>**

**Document.getElementById(“get”).addEventListener(‘click’, getTodos);**

**GET.**

**function getTodos(){**

**//Se le puede pasar un objeto.**

**axios({**

**method: ‘get’,**

**url: ‘https://jsonplaceholder.typicode.com/todos’, //funciona como un fake rest API.**

**Params: {\_limit: 5}**

**}) //Esto devuelve una promesa**

**.then(res => console.log(res));**

**.catch(err => console.error(err));**

**}**

**Esto se puede acortar de la siguiente manera.**

**axios**

**.get(‘{URL}’, params: {\_limit: 5, timeout: 5000}) //Se puede setear los parámetros: el limite de registros que se devuelven(filtro provisto por la api) y el timeout, el tiempo máximo para que se complete la request antes de que se frene(en milisegundos)**

**.then(res => showOutput(res)) en este caso el primer res es el parámetro de la función.**

**.catch(err => console.error(err));**

**POST.**

Function addTodo(){

axios({

method: ‘post’,

url: {URL}

data: {title: ‘New Todo’.

Completed: false}

}) .then(res => showOutput(res))

.catch(err => console.error(err))

}

**function addTodo() {**

**axios.post(‘{URL’, data: {title: ‘New todo’, completed: false))**

**.then(res => showOutput(res))**

**.catch(err => console.error(err))**

**}**

**La diferencia entre put y patch es que put reemplaza el objeto completo a partir de la data que enviamos. Patch actualiza la parte de la data que enviamos.**

**Simultaneous requests.**

**axios.all([**

**axios.get(`${URL}/todos `)**

**axios.get(`${URL}/posts`) //Solicito los todos y los posts**

**])**

**.then(res => {**

**Console.log(res[0]);**

**Console.log(res[1]);**

**})**

**.catch(err => {**

**Console.error(err[0])**

**Console.error([err[1])k**

**})**

**Toma un array de requests. Una vez se completan todas las requests y las promesas, obtenemos la response.**

**La segunda parte se puede limpiar con axios.spread() Donde se asigna cada respuesta a una variable en el orden en que se ejecuto la petición.**

**.then(axios.spread(todos, posts) => showOutput(posts))**

**Agregar headers: Se puede pasar un objeto como tercer parámetro. (URL, Data, Headers)**

**Function customHeaders() {**

**Const config = {**

**“Content-Type” : “application/json”,**

**“Authorization”: “Token aisdh3903uiasjkd”**

**};**

**Axios.post(‘{URL},**

**{title: “New Todo”},**

**config)**

**.then()**

**.catch()**

**}**

**GLOBALES**

**Utilizando los globales, podes mandar un valor de header en todas las requests. Principalmente es útil a la hora de trabajar con tokens, no queres agregarlo en cada request.**

**Axios.defaults.headers.common[‘Authorization’]= ‘Token iojefja093’;**

**Se obtiene el token del server al loguearse, se guarda en el localstorage y se guarda en un global.**

**Error handling**

**Hasta ahora solo hicimos console.error a los errores. Supongamos que se obtiene un 404 en nuestra app de react. Vamos a querer mostrar una pagina 404 o hacer algo al respecto.**

**Function errorHandling() {**

**Axios.get({URL})**

**.then()**

**.catch(error => {**

**If(error.response){**

**//server responded with a status other than 200.**

**Console.log(error.response.data)**

**Console.log(error.response.status)**

**Console.log(error.response.headers)**

**}**

**})**

**}**