DigitalHouse>

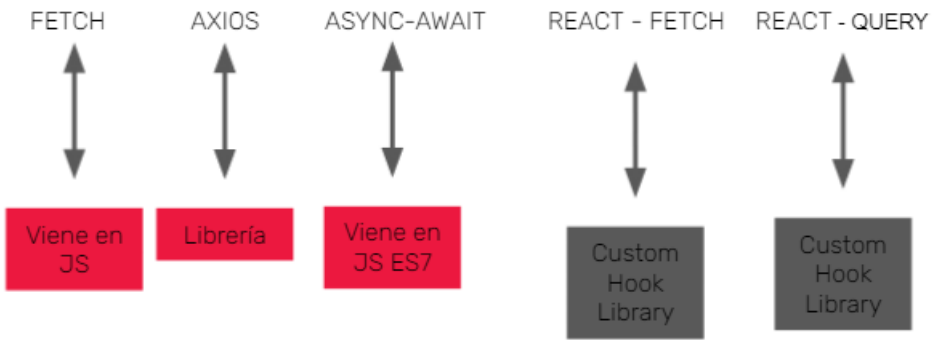
Certified Tech Developer

The Ultimate Degree

Front End III

Llamados a API

Conozcamos las distintas maneras para hacer un llamado a una API. Cada una cumple con su objetivo de devolver la data que necesitamos con la diferencia de su estructura. Vamos a repasar los recuadros rosas. Los recuadros grises son usados con hooks, pero estos los veremos más adelante en otras clases.



Fetch

La primera manera de hacer llamadas es la más utilizada y fácil de acceder: La fetch API es una herramienta empleada en los navegadores para hacer pedidos HTTP usando promesas en JS. Simplemente, debemos hacer nuestro pedido ejecutando un componentDidMount para obtener la data que nos llega. Como toda promesa, debemos asegurarnos de no tener un error con el catch. Finalmente, cambiamos el loader y usamos el render().

```
import React, { Component } from 'react'

export default class AxiosComponent extends Component {
   constructor() {
```

```
        super();
        this.state = {
            data: null,
            loading: true,
            error: null
        }};
    componentDidMount() {
        fetch('https://randomuser.me/api/')
            .then((response) => response.json())
            .then((data) => console.log("Fetch Data:",data))
            .catch((error) => {
                this.setState({error:error})
            })
            .finally(() => this.setState({loading: false}))
    }
    render() {
        if(this.state.error) return "Error!";
        return this.state.loading ?  <p>Loading...</p> : <p>USANDO FETCH</p>
    }
}
```

Axios

Otra manera de hacer llamadas es usando una librería llamada axios. Esta librería nos ahorra tener que obtener el JSON para luego tener la data. Es decir, nos ahorra el primer callback donde obtenemos una response. Entonces, la sintaxis queda reducida.

```
import React, { Component } from 'react';
import axios from 'axios';

export default class FetchComponent extends Component {
    constructor() {
        super();
        this.state = {
            data: null,
            loading: true,
            error: null
        }};
```

```
    componentDidMount() {
        axios('https://randomuser.me/api/')
            .then((response) => console.log("Axios Data:", response.data))
            .catch((error) => this.setState({error:error}))
            .finally(() => this.setState({loading: false}))
    };
    render() {
        if(this.state.error) return "Error!";
        return this.state.loading ?  <p>Loading...</p> : <p>USANDO AXIOS</p>
    }}
```

Async await y try catch

En este caso, implementamos un try catch donde realizamos el pedido a la API. Terminamos nuevamente utilizando el finally para cambiar nuestro estado loading con el objetivo de devolver la respuesta o el error.

```
import React, { Component } from 'react'
import axios from 'axios';

export default class AxiosComponent extends Component {
    constructor() {
        super();
        this.state = {
            data: null,
            loading: true,
            error: null
        }};
    componentDidMount() {this.getData()}

    getData = async () => {
        try {
            const response = await axios('https://randomuser.me/api/');
            console.log("Async Await Data en Try Catch:", response.data);
        } catch (error) {
            this.setState({error:error});
        } finally {
            this.setState({loading: false});
```

```
        }
    };

    render() {
        if(this.state.error) return `Error: ${this.state.error.message}`;
        return this.state.loading ?  <p>Loading...</p> : <p>USANDO ASYNC
AWAIT y TRY CATCH</p>
    }}
```

Por último, podés acceder al repositorio en caso de que quieras copiar la sintaxis.

¡Hasta la próxima!