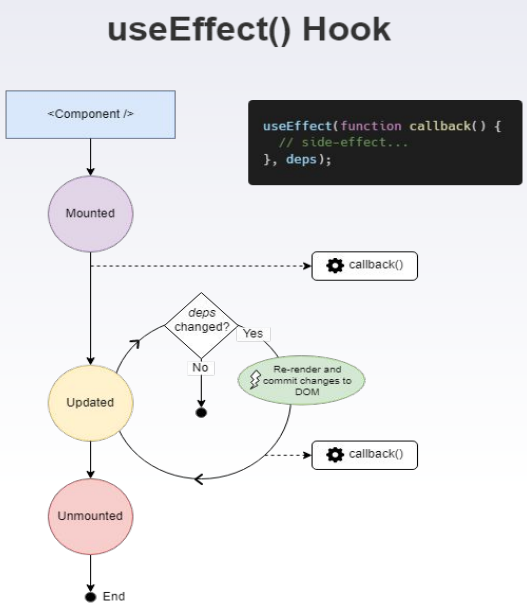
useEffect

Comprende dos argumentos:

useEffect(<función>, <dependencias>)

Ejemplo:

```
useEffect(() => {
    console.log("Esto se ejecuta una sola vez")
}, [])
```



En este caso, la función es una Arrow function y la dependencia es un array vacío.

Cuando el componente

Se monta

```
useEffect(() => {
    console.log("Esto se ejecuta una sola vez")
}, [])
```

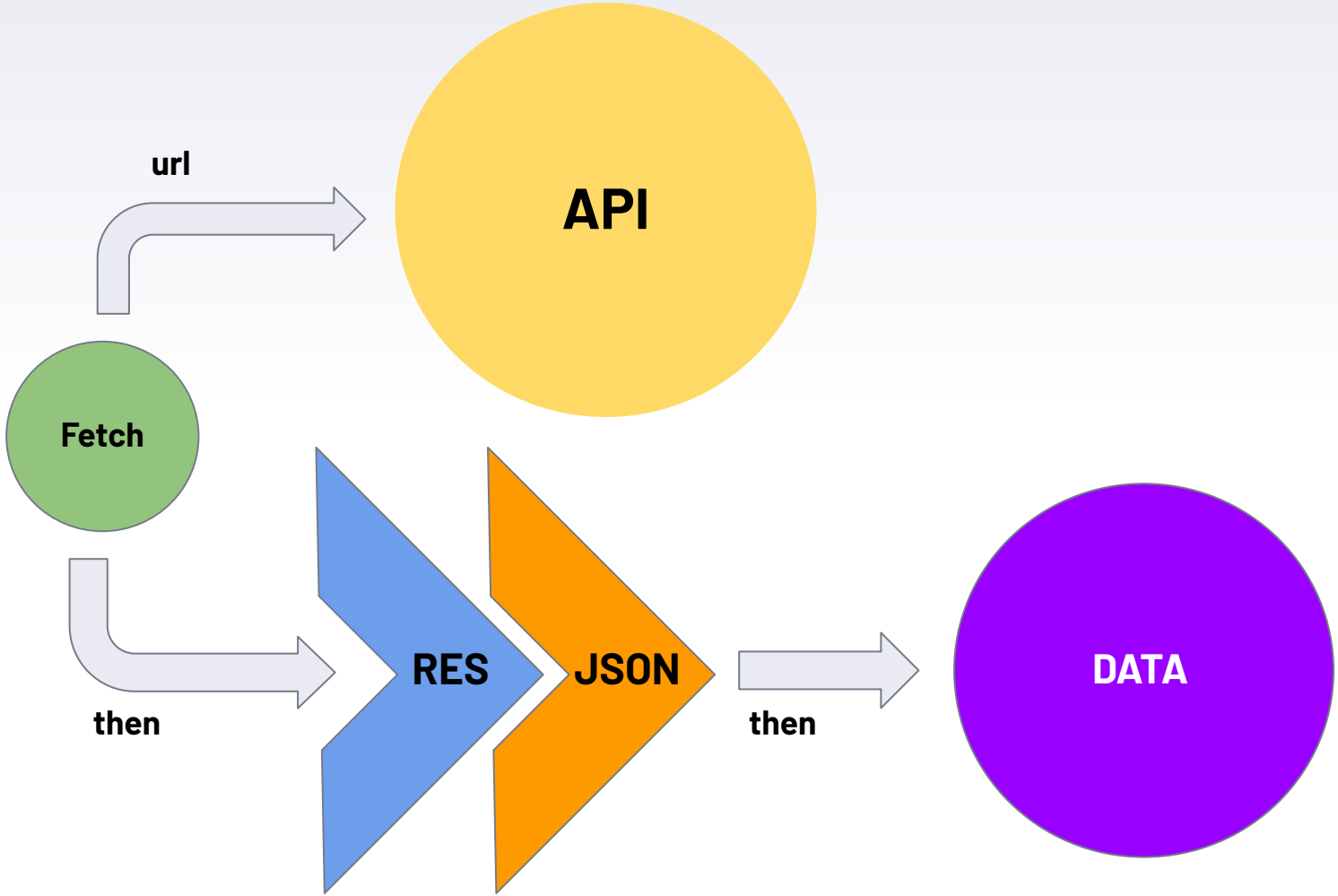
Se actualiza

```
useEffect(() => {
    console.log("Esto se ejecuta cuando name se actualiza")
}, [name])
```

Se desmonta

```
useEffect(() => {
    return () => {
        console.log("Esto se ejecuta cuando se destruye el componente")
    }
}, [])
```

Fetch()



Como se vería Fetch( ) en el código:

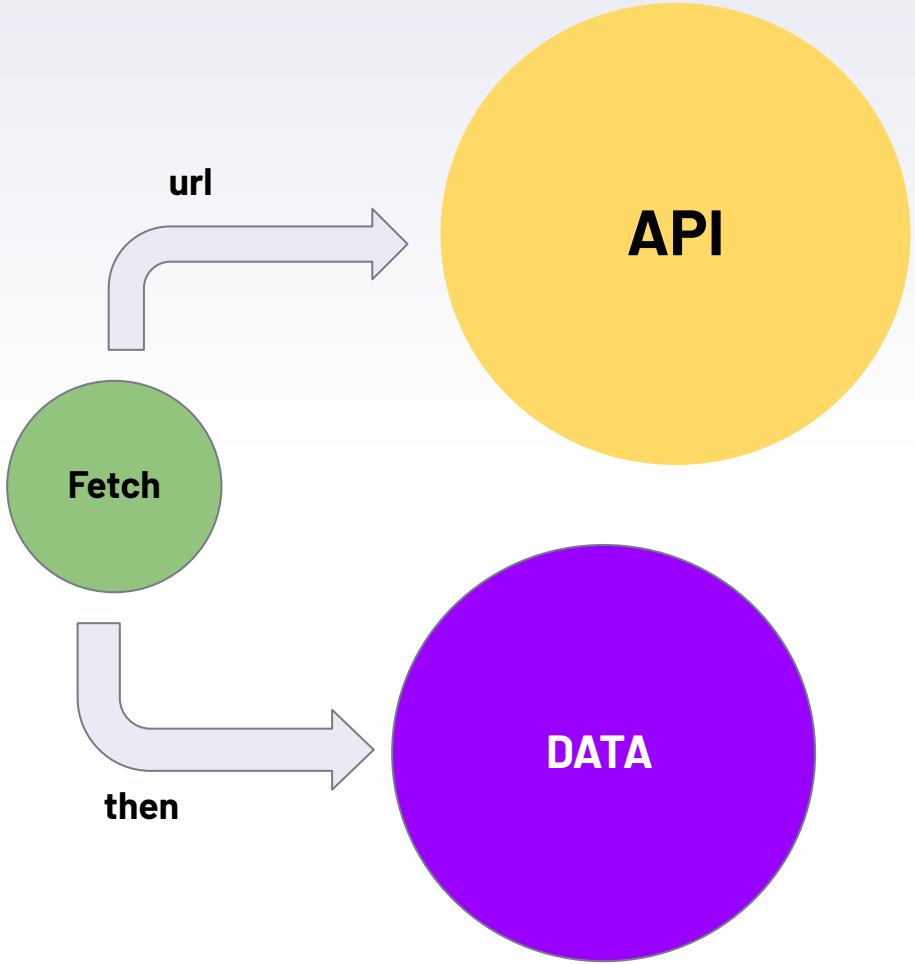
```
fetch(url)
        .then((response) => response.json())
        .then((data) => {
            console.log(data)
            })
```

- Colocamos la **url** dentro de fetch()
- Luego lo transformamos la **respuesta** en un **json**
- Finalmente, usamos ese **json** y lo traemos como "**data**" para después poder utilizarla en nuestro código.

```
    useEffect(() => {
        fetch(url)
        .then((response) => response.json())
        .then((data) => {
            console.log(data.results)
            })
    }, [])
```

Es importante utilizar el fetch dentro de **useEffect** para que realicé ésta acción una vez. Sino, corremos el riesgo de que haga varias peticiones a la api, lo cual afectaría al renderizado y a la performance de nuestra app.

Axios



Como se vería Axios( ) en el código:

```
axios.get(url)
    .then (response => {
        console.log(response);
    })
```

- Colocamos la **url** dentro de axios.get()
- Luego, ya podemos acceder a la respuesta y poder guardarla en un estado.
- En este caso, nos estamos ahorrando de pasar la información a un json

---

Recordemos que para utilizar axios, tenemos que instalar la librería en nuestro proyecto con el siguiente comando:

```
npm install axios
```

También es importante mencionar que al igual que fetch, es recomendable utilizar esta función dentro de un useEffect para que la acción se realice una sola vez.

Métodos de Axios

```
axios.request(config)

axios.get(url[, config])

axios.delete(url[, config])

axios.head(url[, config])

axios.options(url[, config])

axios.post(url[, data[, config]])

axios.put(url[, data[, config]])

axios.patch(url[, data[, config]])
```

Axios comprende de varios métodos. Todos los métodos pueden recibir el argumento de una **configuración**, en donde se especifican los headers y las credenciales. También reciben una **url**, el cual será la endpoint de la API al que queremos acceder. Y los métodos **post**, **put** y **patch** reciben el argumento de **data** el cual contendrá la información que vamos a mandar a la API.

Vamos a trabajar con estos últimos métodos, más adelante. Por ahora solo nos vamos a quedar con el método de **axios.get( )**

Manejo de errores

```
fetch(url)
        .then((response) => response.json())
        .then((data) => {
            console.log(data)
            })
        .catch(err => console.log(err))
```

```
axios.get(url)
    .then (response => {
        console.log(response);
    })
    .catch(err => console.log(err))
```

Para saber si hubo algún error con nuestra petición, en ambos casos utilizamos la palabra reservada **.catch()**, en donde podemos colocar una función que nos devolverá un error. Generalmente, lo que se hace en la función es un console.log() para tener un detalle del error sucedido.

Async/Await

Async/Await

Async/Await viene a ser otra forma de resolver las promesas.
La palabra reservada **async** hace que una función sea considerada asincrona, y de esa forma, esta función pasa a ser una promesa.
Mientras que, **await** es la forma de como resolver esta promesa.

```
async function getUser() {
 let response = await fetch(url);
 let userData = await response.json();
 return userData.name; // no es necesario await en el return
}
```

Su implementación es colocar async antes de declarar la función y await antes de realizar cada acción que implica la resolución de una promesa.

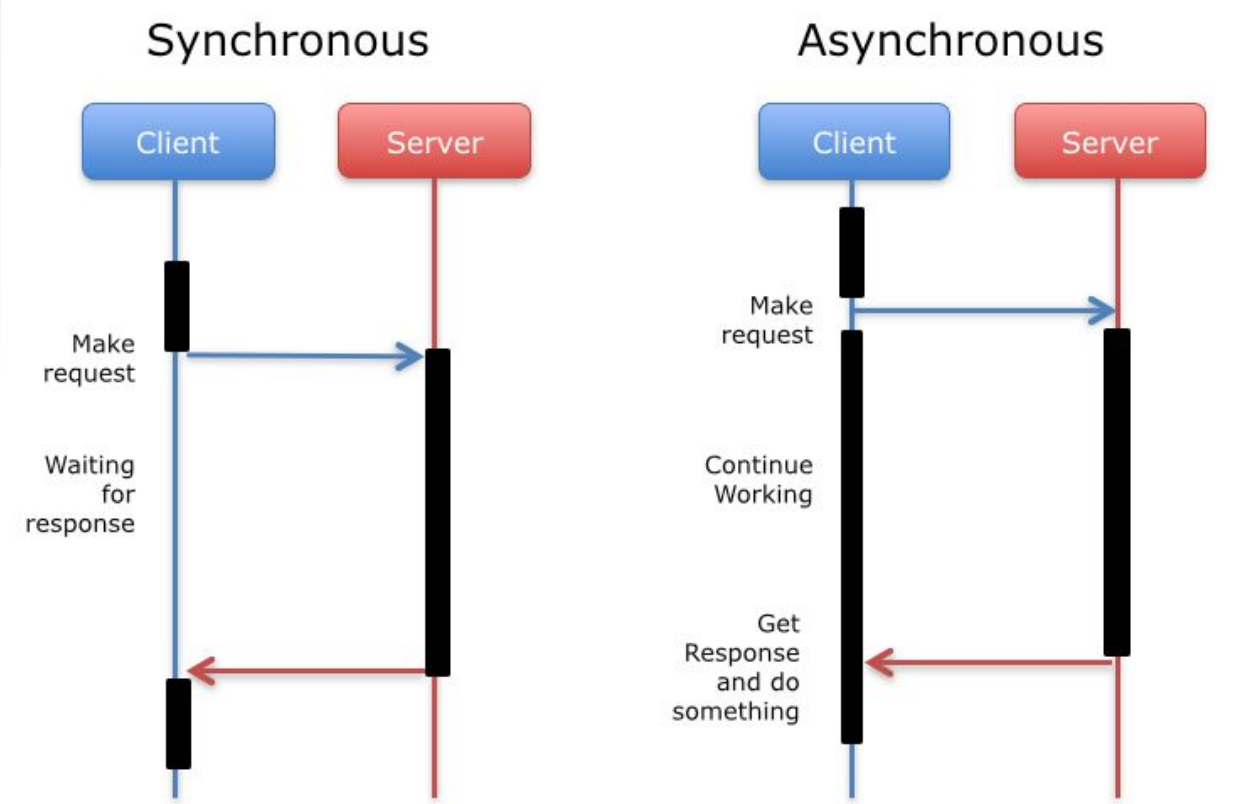
.then()

```
    useEffect(() => {
        const fetchData = () =>  {
            fetch(url)
            .then((response) => response.json())
            .then((data) => console.log(data))
            .catch(err => console.log(err))
        }
        fetchData();
    }, [])
```

Async/Await

```
    useEffect(() => {
        const fetchData = async () =>  {
            const response = await fetch(url)
            const data = await response.json()
            return console.log(data)
        }
        fetchData();
    }, [])
```

Async/Await


Synchronous
Client
Server
Make request
Waiting for response
Asynchronous
Client
Server
Make request
Continue Working
Get Response and do something

Consigna de hoy

- **Hacer un llamado de la api utilizando PokeApi (u otra api que sea accesible) y renderizar los datos que puedo extraer.**
- **Para esto necesitaremos useEffect y Fetch o Axios**