**Vue**

**Desestructuración de objetos.**

**Documentación:**

[**https://v3.vuejs.org/guide/introduction.html**](https://v3.vuejs.org/guide/introduction.html)

[**https://vuejs.org/v2/guide/**](https://vuejs.org/v2/guide/)

Para hacer una desestructuración de un objeto podemos usar las **“{}”** en la variable constante e indicar las propiedades que queremos obtener del objeto, sin importar el orden del acomodo de las propiedades.Ejemplo:

const heroes = [{

    id: 1,

    nombre: "Batman"

},{

    id: 2,

    nombre: "Superman"

},{

    id: 3,

    nombre: "Spiderman"

},{

    id: 4,

    nombre: "Ironman"

},{

    id: 5,

    nombre: "Dr Strange"

}];

const {nombre, id} = heroes.find( (heroe) => heroe.id === 1 );

console.log(id); // 1

console.log(nombre); // Batman

Otra forma de desestructurar los objetos es ir navegando sobre sus propiedades con un **“.” (punto).** Ejemplo:

const persona = {

    nombre: "Benjamín",

    apellido: "Ayala",

    edad: 29,

    apodo: "Benyi"

}

console.log(persona.nombre); // Benjamín

console.log(persona.apellido); // Ayala

console.log(persona.edad); // 29

console.log(persona.apodo); // Benyi

Pero la idea de la desestructuración de los objetos es usar solamente las propiedades que necesitamos. Ejemplo:

const { nombre, apellido, apodo } = persona;

console.log(nombre); // Benjamín

console.log(apellido); // Ayala

console.log(apodo); // Benyi

Una de las ventajas que tiene la desestructuración de los objetos, es que si declaramos una variable que no se encuentre dentro del objeto, esta solamente marcara undefined. **La razón de esto es porque básicamente todo lo que no está asignado en JS es undefined, pero una de las funciones interesantes de la desestructuración es que podemos poner un valor por defecto.** Ejemplo:

Ejemplo:

const { nombre, apellido, edad, apodo, altura, peso = '80kg' } = persona;

console.log(nombre); // Benjamín

console.log(apellido); // Ayala

console.log(edad); // 29

console.log(apodo); // Benyi

console.log(altura); // undefined

console.log(peso); // 80kg

**Desestructuración de los objetos desde los argumentos de las funciones.**

Un uso bastante común es también la desestructuración desde los argumentos de las funciones. Que quiere decir esto, que en vez de desestructurar un objeto dentro de una función, se puede desestructurar desde el inicio en los argumentos que se reciben. Ejemplo:

1. **Desestructuración del objeto dentro de la función**

const crearHeroe = ( persona ) => {

    const { nombre, apellido, edad, apodo, altura } = persona;

    return {

        id: Math.floor(Math.random() \* 100),

        nombre: nombre,

        apellido: apellido,

        edad: edad,

        apodo: apodo,

        altura: altura

    };

}

1. **Desestructuración del objeto desde los argumentos de la función.**

const crearHeroe = ( { nombre, apellido, edad, apodo, altura } ) => {

    return {

        id: Math.floor(Math.random() \* 100),

        nombre: nombre,

        apellido: apellido,

        edad: edad,

        apodo: apodo,

        altura: altura

    };

}

**Nota importante.**

Cuando tenemos el caso en el que una propiedad tiene el mismo nombre que el nombre de una variable o constante que tengamos en nuestro código, podemos obviar la relación de la propiedad con el nombre de la variable o también podemos renombrar una propiedad con los **nombrePropiedad:nuevoNombre**. Ejemplo:

const crearHeroe = ( { nombre, apellido:apellidoPaterno, edad, apodo, altura } ) => (

    // Cuando tenemos el caso en el que una propiedad tiene el mismo nombre que el nombre de una variable o constante que tengamos en nuestro código, podemos obviar la relación de la propiedad con el nombre de la variable.

    // Relación de las propiedades con las variables constantes, los nombres serán las llaves y los valores serán lo que se tenga asignado en las variables constantes

    {

        id: Math.floor(Math.random() \* 100),

        nombre,

        apellidoPaterno,

        edad,

        apodo,

        altura

    }

);