**Destructuring**

Para explicar que es la asignación por destructuración, pensemos en el siguiente escenario. Imaginemos que tenemos la información de un alumno incluyendo las calificaciones en 3 asignaturas (matemática, informática y física), representado por un objeto, y necesitamos mostrar cierta información basada en esos datos.

Se vería algo así:

**const estudiante = {**

**nombre: 'Ada Lovelace',**

**edad: 16,**

**calificaciones: {**

**matematica: 10,**

**informatica: 9,**

**fisica: 8**

**}**

**};**

**function mostrarResumen (alumno) {**

**console.log('Hola, ' + alumno.nombre);**

**console.log('Calificación en Matemática: ' + (alumno.calificaciones.matematica || 0));**

**console.log('Calificación en Informática: ' + (alumno.calificaciones.informatica || 0));**

**console.log('Calificación en Física: ' + (alumno.calificaciones.fisica || 0));**

**}**

**mostrarResumen(estudiante);**

**// Hola, Ada Lovelace**

**// Calificación en Matemática: 10**

**// Calificación en Informática: 9**

**// Calificación en Física: 8**

Con este código obtenemos el resultado esperado. Sin embargo, tiene algunos puntos que se pueden mejorar. Uno de los puntos es que podríamos fácilmente tener un error de tipeo y escribir **alumno.calificacion.matematica** lo que resultaría en un error. Otro problema es que si la propiedad **calificaciones** tuviera más objetos adentro, la cadena de propiedades sería muy larga y propensa a errores.

Destructuring va a permitirnos hacer hacer lo mismo de una forma expresiva y compacta.

**Entonces… ¿Qué es Destructuring?**

**Destructuring** es simplemente partir una estructura compleja en partes más simples. En JS estas estructuras complejas son generalmente objetos o arrays.

Con la sintáxis de **destructuring**, vamos a poder extraer fragmentos más chicos de arrays/objetos. Se puede utilizar para la declaración o asignación de variables.

Con **destructuring**, la función del ejemplo anterior se vería de la siguiente forma:

**function mostrarResumen ( { nombre, calificaciones: { matematica = 0, informatica= 0, fisica= 0 }} ) {**

**console.log('Hola, ' + nombre);**

**console.log('Calificación en Matemática: ' + matematica);**

**console.log('Calificación en Informática: ' + informatica);**

**console.log('Calificación en Física: ' + fisica);**

**}**

**Object Destructuring**

Lo que vimos en los anteriores ejemplos es una forma de destructuración de objetos. Vamos a ver qué podemos hacer con **object destructuring**.

**const estudiante = {**

**nombre: 'Ada',**

**apellido: 'Lovelace',**

**pais: 'UK'**

**};**

**// Object Destructuring**

**const { nombre, apellido, pais } = estudiante;**

**console.log(nombre, apellido, pais); // Ada Lovelace UK**

Usamos **object destructuring** para asignar valores a 3 variables: nombre, apellido y pais, usando los valores de sus correspondientes claves en el objeto del estudiante. Es la forma más básica de utilizar **destructuring**.

También podemos utilizar valores por defecto. Cuando tratamos de asignar una variable correspondiente a una clave que no existe en el objeto, el resultado que vamos a tener es **undefined**. Podemos utilizar valores por defecto para asignar ese valor en lugar de undefined cuando la propiedad no se encuentra presente en el objeto.

**const estudiante = {**

**nombre: 'Ada',**

**apellido: 'Lovelace'**

**};**

**// Object Destructuring**

**const { nombre, apellido, pais = 'AR' } = estudiante;**

**console.log(`Hola soy ${nombre} ${apellido}, y soy de ${pais}`); // Hola soy Ada Lovelace, y soy de AR**

En este ejemplo le dimos el valor por defecto de **AR** a la variable **pais**. Como la propiedad no se encuentra en el objeto, **AR** se asigna a la variable **pais** en lugar de undefined.

Hasta ahora utilizamos nombres de variables que coinciden con el nombre de las propiedades del objeto. Podemos asignar los valores a variables con diferentes nombres utilizando la sintaxis **[clave]: [nombreVariable]**. Incluso es posible asignarle un valor por defecto de la siguiente forma **[clave]: [nombreVariable] = [valorDefault]**.

**const estudiante = {**

**nombre: 'Ada',**

**apellido: 'Lovelace',**

**pais: 'UK'**

**};**

**// Object Destructuring**

**const { nombre: firstname, apellido: lastname, pais: country = 'AR' } = estudiante;**

**console.log(`Hola soy ${firstname} ${lastname}, y soy de ${country}`); // Hola soy Ada Lovelace, y soy de UK**

En este ejemplo creamos 3 variables locales (firstname, lastname, country), que se corresponden con las propiedades nombre, apellido y pais del objeto estudiante. Notemos como le asignamos un valor por default a country, para evitar que sea undefined en caso que no exista esa propiedad en el objeto.