El concepto de JavaScript Destructuring nos puede resultar útil en el día a día de nuestro trabajo con JavaScript. ¿Qué es y para que sirve el concepto de Destructuring? . Como su nombre indica sirve para desestructurar una estructura determinada. Esto en principio nos puede resultar chocante a los programadores nos gusta tener todo ordenado. Vamos a empezar a entenderlo todo un poco mejor a través de un ejemplo sencillo. Supongamos que tenemos un array con información:

```
var datosPersona=["pepe","perez",20];

console.log(datosPersona[0]);
console.log(datosPersona[1]);
console.log(datosPersona[2]);
```

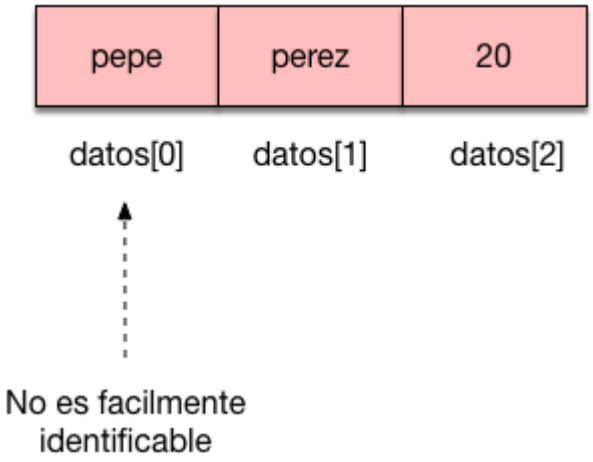
El resultado sale por la consola:

```
pepe
perez
20
```

Esa información es fácilmente imprimible accediendo a cada posición del array , es algo sencillo. ¿Sin embargo es algo claro? . Si por supuesto que lo es … bueno miremos de otra forma.

```
console.log(datosPersona[0]);
console.log(datosPersona[1]);
console.log(datosPersona[2]);
```

¿Sería claro si simplemente tuviéramos los consoles logs?..



La realidad es que no ya que no sabemos con certeza cual es la información guardada en cada posición.

JavaScript Destructuring un enfoque moderno

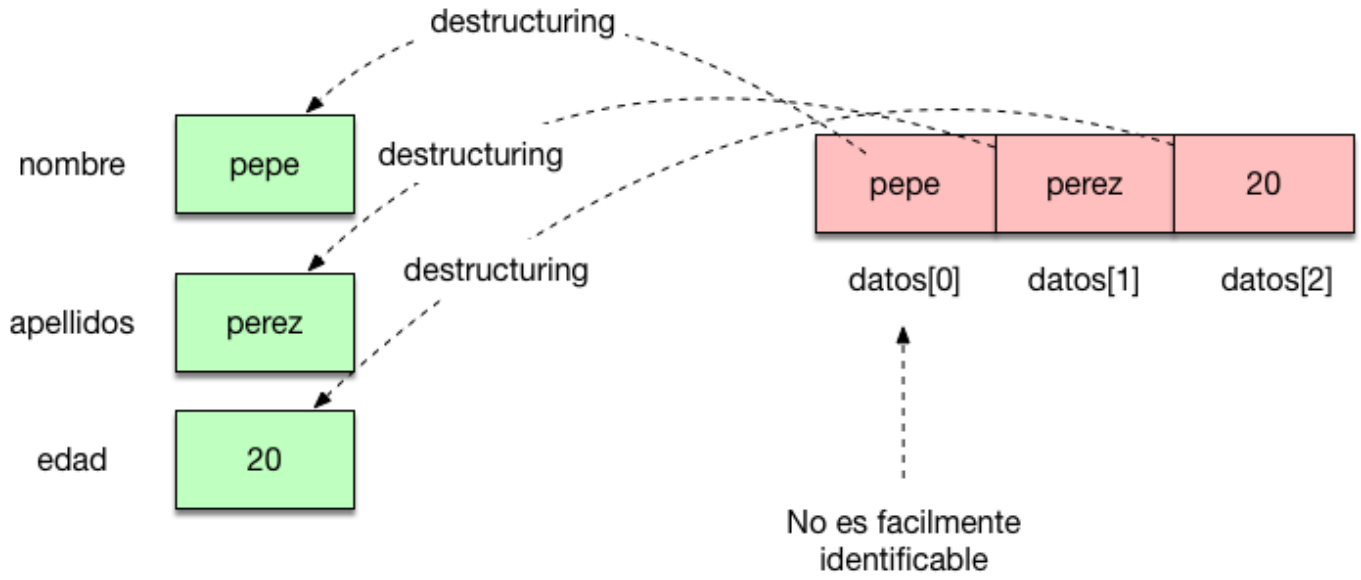
Para evitar este tipo de problemas y poder acceder a la información de forma mas clara podemos usar el concepto de JavaScript Destructuring y asignar la información de la siguiente forma:

```
var [nombre,apellidos,edad]= datosPersona;

console.log(nombre);
console.log(apellidos);
console.log(edad);
```

Esta forma nos permite acceder a los datos del array de forma totalmente desestructurada ,

es como desmontar el array.



Sin embargo ganamos de una forma clara en legibilidad y es realmente sencillo construirlo. El resultado es idéntico:

```
pepe
perez
20
```

No solo eso sino que podemos solicitar que la desestructuración sea parcial y solo obtener algunos de los valores .

```
var [nombre,apellidos]= datosPersona;

console.log(nombre);
console.log(apellidos);
```

El resultado se muestra en consola:

```
pepe
perez
```

Ademas podemos a la hora de desestructurar añadir nuevos elementos que nos sean necesarios:

```
var [nombre,apellidos,edad,categoria="estudiante"]= datosPersona;

console.log(nombre);

console.log(apellidos);

console.log(edad);

console.log(categoria);
```

Veamos el resultado en la consola:

```
pepe
perez
20
estudiante
```

El concepto de Javascript destructuring soporta muchas combinaciones diferentes. Por ejemplo se podría complementar con rest parameters y generar dos estructuras.

```
var [nombre,...restoElementos]= datosPersona;

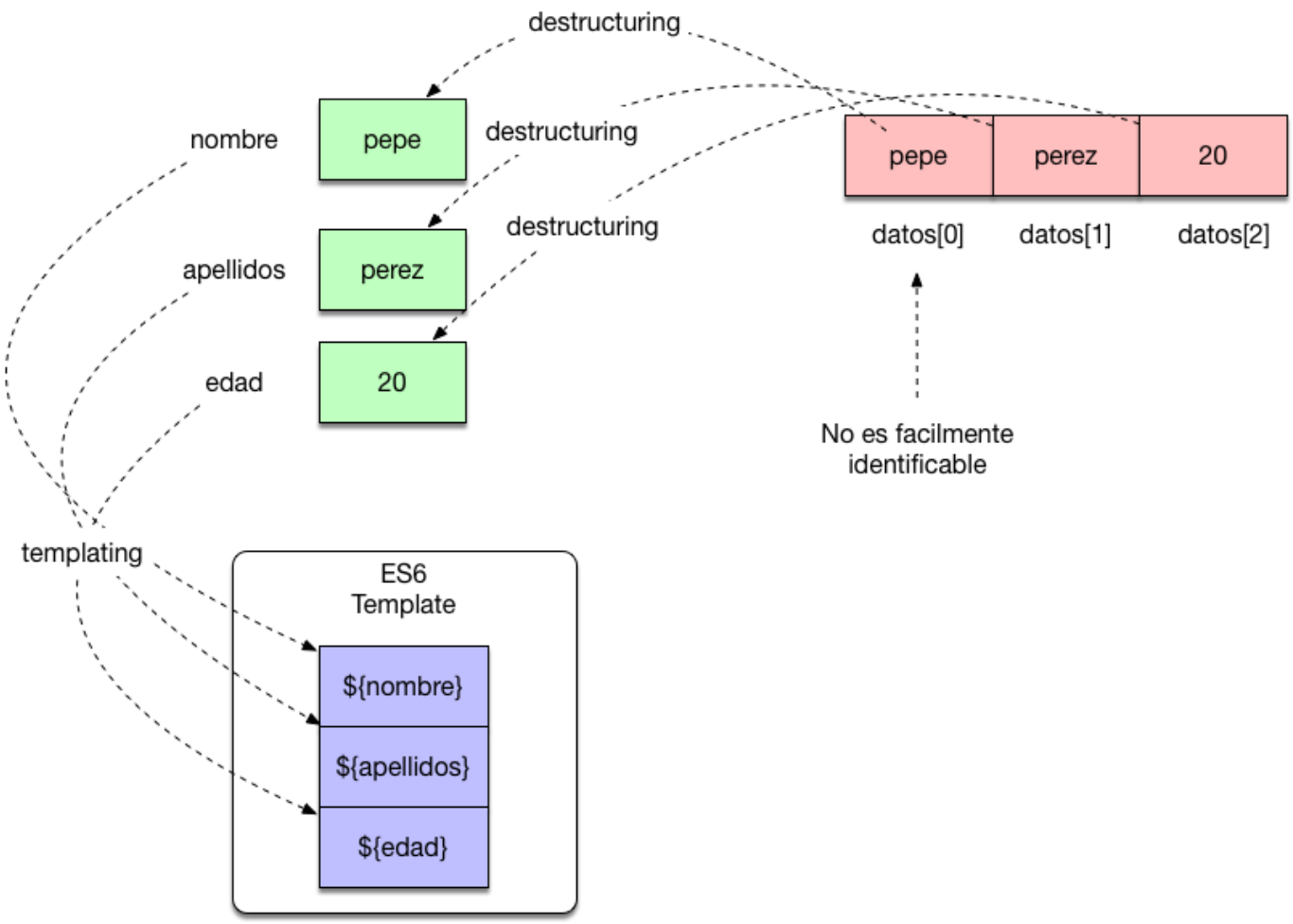
console.log(nombre);
console.log(restoElementos);
```

Nos muestra el nombre y un array por consola:

```
pepe
[ 'perez', 20 ]
```

JavaScript Template Strings

Una de las mayores ventajas del concepto de destructuring es cuando lo combinamos con otras de las características de JavaScript ES6 como son los JavaScript Template Strings



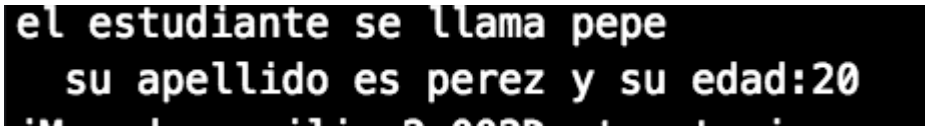
Vamos a verlo que es muy sencillo:

```
var [nombre,apellidos,edad]= datosPersona;

console.log(`el estudiante se llama ${nombre}
  su apellido es ${apellidos} y su edad:${edad}`);
```

En este caso lo que hacemos es usar javascript destructuring para simplificar el

funcionamiento de una plantilla de ES6 y que todo sea más legible.



Aprendamos a usar el concepto de JavaScript Destructuring que nos ayudará a simplificar nuestro código:

1. JavaScript ES6 fetch API
2. Spread vs Rest Parameters en ES6
3. Introduccion a Babel.js y JavaScript ES6
4. JavaScript Template for loop y como usarlo
5. Curso Ajax JSON Promesas y buenas prácticas
6. ES6 y JavaScript