Desestructuración en javascript

Laura Isabel López Naves

¿Qué es la desestructuración?

- La desestructuración es una nueva caracterísca incluida en ES6 que permite desempaquetar valores de arrays o propiedades de objetos en distintas variables.
- El objetivo es escribir código más conciso y claro.

```
// Dada la siguiente tabla:
const array = ['José', 'Pérez', 'García', 'C/ Mayor,1'];
```

```
// Podemos escribir:
const nombre = array[0];
const apellido1 = array[1];
console.log(nombre); // José
console.log(apellido1); // Pérez
```

```
// O utilizando desestructuración:
const [nombre, apellido1] = array;
console.log(nombre); // José
console.log(apellido1); // Pérez
```

Nota: la sentencia “const[nombre, edad]=…” crea las variables “nombre” y “edad” si no existen.

Desestructurando arrays

- Es posible desempaquetar solo ciertos campos y obviar otros:

```
const array = ['José', 'Pérez', 'García', 'C/ Mayor,1'];

const [nombre,,, direccion] = array;
console.log(nombre); // José
console.log(direccion); // C/ Mayor,1
```

- O utilizar parámetros “rest” para recoger el resto de los datos en otro array:

```
const array = ['José', 'Pérez', 983344343, 9834343];

const [nombre,, ...telefonos] = array;
console.log(nombre); // José
console.log(telefonos); // [983344343, 9834343];
```

Desestructurando arrays

- Se pueden utilizar valores por defecto (por ejemplo con “apellido”):

```
const array = ['José'];
const [nombre, apellido = 'Rodríguez'] = array;
console.log(nombre); // José
console.log(apellido); // Rodríguez
```

- Se pueden inicializar varias variables utilizando desestructuración:

  (Si las variables ya existen no se debe utilizar “const”)


```
// Con variables existentes:
let a = 0;
let b = 0;
let c = 0;
[a, b, c] = [1, 2, 3];
console.log(a, b, c); // 1 2 3

// Creando nuevas varibles:
const [x, y, z] = [4, 5, 6];
console.log(x, y, z); // 4 5 6
```

- También es posible intercambiar el valor de variables mediante desestructuración:

```
let a = 3;
let b = 6;
[a, b] = [b, a];
console.log(a); // 6
console.log(b); // 3
```

Desestructurando objetos

- En el siguiente código extraemos atributos de un objeto en variables:

```
const persona = {
  nombre: 'Sarah',
  pais: 'Nigeria',
  trabajo: 'Desarrollador'
};

const nombre = persona.nombre;
const pais = persona.pais;
const trabajo = persona.trabajo;

console.log(nombre); // "Sarah"
console.log(pais); // "Nigeria"
console.log(trabajo); // "Desarrollador"
```

- El código equivalente utilizando desestructuración es:

```
const persona = {
  nombre: 'Sarah',
  pais: 'Nigeria',
  trabajo: 'Desarrollador'
};

const { nombre, pais, trabajo } = persona;

console.log(nombre); // "Sarah"
console.log(pais); // "Nigeria"
console.log(trabajo); // "Desarrollador"
```

Desestructurando objetos

- Si se desea hacer desestructuración en variables existentes se debe evitar el uso de const/let e incluir la sentencia entre paréntesis. Ejemplo:

```
const persona = {
  nombre: 'Sarah',
  trabajo: 'Desarrollador'
};
let nombre, trabajo;
({ nombre, trabajo } = persona);
console.log(nombre); // Sarah
console.log(trabajo); // Desarrollador
```

- También es posible utilizar valores por defecto. Ejemplo:

```
const persona = {
  nombre: 'Sarah',
  trabajo: 'Desarrollador'
};

const { nombre = 'miNombre', amiga = 'Annie' } = persona;

console.log(nombre); // Sarah
console.log(amiga); // Annie
```

Desestructurando objetos

- También es posible cambiar el nombre de las variables. Ejemplo:

```
const persona = {
  nombre: 'Sara',
  pais: 'Italia'
};

const { nombre: firstName, pais: country = 'España' } = persona;

console.log(firstName); // Sara
console.log(country); // Italia
console.log(nombre); // <- Variable no definida
```

- El objeto tiene los atributos “nombre” y “país” y se desestructuran en “firstName” y “country”.
- La variable “nombre” no se crea.
- “España” es el valor por defecto del atributo “país”.

Desestructuración anidada.

```
const persona = {
  nombre: 'Sarah',
  lugar: {
    pais: 'Nigeria',
    ciudad: 'Lagos'
  },
  amigas: ['Annie', 'Becky']
};

const { nombre: foo, lugar: { pais: bar, ciudad: x } } = persona;
console.log(foo, bar, x); // Sarah Nigeria Lagos
const { lugar: { ciudad } } = persona;
console.log(ciudad); // Lagos
const { amigas: [primera] } = persona;
console.log(primera); // Annie
```

Desestructuración en la llamada a funciones

```
const persona = {
  nombre: 'Pedro',
  domicilio: 'C/ Mayor, 1',
  pais: 'España'
};
function visualizarDatos1 (obj) {
  console.log(obj);
}
function visualizarDatos2 ({ nombre, domicilio }) {
  console.log(nombre + ' --- ' + domicilio);
}
visualizarDatos1(persona);
visualizarDatos2(persona);
```

VisualizaDatos1 recibe como parámetro el objeto completo pero VisualizarDatos2 realiza una desestructuración en la invocación y solo recibe las propiedades “nombre” y “domicilio”.

Basta con incluir entre llaves los nombres de las propiedades que se desean desestructurar en variables dentro de la función.

Ejercicio4 (ej4.js)

```
// Dado el siguiente objeto:
const persona = {
  nombre: 'Pepe',
  apellidos: ['García', 'Peréz'],
  edad: 23
};
// Desestructurar "nombre" y "edad" en las variables ej1 y ej2:
// ...
// Dato el siguiente array:
const arr = ['lunes', 'martes', 'miércoles', 'jueves', 'viernes'];
// Desestucturar el primer y tercer elmento en las variables "lu" y "mi":
// ...
// Desestructurar los apellidos de persona en las variables "apellido1" y "apellido2"
// ...
// Crear una función denominada "verDatos" que admita como parámetro el objeto persona y,
// realizando desestructuración, cree y visualice las variables "nombre" y "edad"
// con los valores por defecto "Luis" y 20 respectivamente. Probar con diferentes objetos.
// ...
```