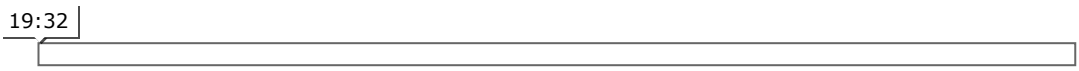
Watch: 😴 **Código menos verboso con rest, spread y destructuring**



Tienes el código de este vídeo en GitHub.

Destructuring

El destructuring consiste en asignar valores de objetos o arrays a distintas variables de forma concisa. Por ejemplo:

```
createUser(form, function (response) {
  if (!response.success) {
    handleFormError();
    return;
  }

  log("User created", response.data);
  handleFormSuccess(form, response.data);
});
```

Podemos usar destructuring en el parámetro que recibe la función para usar directamente `success` y `data`:

```
createUser(form, function ({ success, data }) {
  if (!success) {
    handleFormError();
    return;
  }

  log("User created", data);
  handleFormSuccess(form, data);
});
```

También podemos cambiar el nombre de la variable de la siguiente forma:

```
createUser(form, function ({ success, data: newUser }) {
  if (!success) {
    handleFormError();
    return;
  }

  log("User created", newUser);
  handleFormSuccess(form, newUser);
});
```

Spread

El operador spread `...` nos permite *desempaquetar* varias propiedades de un objeto o valores de un array, permitiendo asignarlas a un nuevo objeto o array.

```
var date = new Date();
log("User created", {
  date: date,
  ...newUser,
});
```

Object shorthand

El object shorthand nos permite asignar la key de un objeto al valor de una variable de la siguiente forma si ambas comparten el mismo nombre:

```
var date = new Date();
  log("User created", {
    date,
    ...newUser,
  });
```

Rest

El operador rest tiene la misma sintaxis que el spread, pero hace justamente lo contrario: agrupa varios valores dentro de un mismo array. Por ejemplo, podemos agrupar los parámetros de una función en un array:

```
addInterests(user, "coding", "css", "cooking");

function addInterests(user, ...interests) {
  user.interests = [...user.interests, ...interests];
}
```

⚠️ Import & Export syntax

A pesar de que puede parecerlo, no podemos usar destructuring al importar un módulo que exporte un objeto. Si queremos hacerlo, debemos hacerlo una vez el módulo ya está importado:

```
// utils.js
export default { countChars, iterateNodes };

// main.js
import utils from "./utils";
```

```
var { countChars, iterateNodes } = utils;
```

Si vemos un import que nos recuerda a la desestructuración de un objeto, es porque ese módulo usa *named exports* en lugar de `export default`:

```
// utils.js

export function countChars(str) { }
export function iterateNodes(array, callback) { }

// main.js
import { countChars, iterateNodes } from "./utils";
```

Enlaces relacionados:

- MDN: Spread & Rest syntax
- MDN: Destructuring assignment
- MDN: Import statement