CURSO DESAROLLO WEB CON MEAN

WEB FULL STACK DEVELOPER

Germán Caballero Rodríguez
germanux@gmail.com

Desarrollo con

INDICE

ES6: Módulos

Los separación en módulos es un patrón usual en la mayoría de las aplicaciones de javascript.

Estos nos ayudan a resolver distintos problemas, entre los cuales destacan:

- Evitar colisiones de variables por nombres repetidos.
- Separar la lógica de la aplicación en partes específicas y definidas.
- Mantener nuestras variables lejos del scope global.
- Aumentar la comprensibilidad y reusabilidad del código.

ES6: Módulos

Los exports de un modulo son las variables, objetos o funciones que el modulo expone para que otros módulos puedan consumir.

```
// modulo1.js
module.exports = "Hola soy un modulo";

// modulo2.js
module.exports = function suma(a, b) {
  return a + b;
}
```

```
// modulo1.js
export default "Hola soy un modulo";

// modulo2.js
export default function suma(a, b) {
  return a + b;
}
```

ES6: Módulos

En CommonJS también teníamos exports con nombres, o específicos, los cuales exportábamos con module.exports.cosa.

```
// modulo.js
module.exports.SECRETO = "Me gustan los robots";

module.exports.opciones = {
  color: 'rojo',
  imagen: false,
};

module.exports.suma = function(a, b) {
  return a + b;
}
```

ES6: Módulos

En ES6 también podemos hacer esto usando export seguido de una definición estándar de lo que vamos a exportar.

```
// modulo.js
export const SECRETO = "Me gustan los robots";

export var opciones = {
  color: 'rojo',
  imagen: false,
};

export function suma(a, b) {
  return a + b;
}
```

ES6: Módulos

Lo interesante es que el nombre que le des será el mismo usara para importar esa parte del módulo. Este ultimo ejemplo con exports nombrados es equivalente a exportarlos todos dentro de un objeto:

```
// modulo.js
const SECRETO = "Me gustan los robots";

var opciones = {
  color: 'rojo',
  imagen: false,
};

function suma(a, b) {
  return a + b;
}

export default {
  SECRETO,
  opciones,
  suma,
};
```

ES6: Módulos

Imports

import es como la contraparte de export. export hace variables o funciones de un módulo accesible por otros módulos e import se encarga de tomar un modulo y darnos acceso a esas variables y funciones que el módulo expone.

ES6: Módulos

Esto mismo puede ser replicado usando ES6 modules y la sintaxis es sencilla. Si en CommonJS es var miModulo = require('./archivoDelModulo'), en ES6 lo escribimos como import miModulo from './archivoDelModulo.

```
//suma.js
export default function suma(a, b) {
  return a + b;
};

// modulo.js
import suma from './suma';

suma(1, 2); // 3
```

ES6: Módulos

Imports con nombre.

podemos importar sólo una parte del modulo.

```
import express from 'express';

var router = express.Router();
...
```

podemos hacerlo mejor usando imports por nombre

```
import { Router } from 'express;

var router = Router();
```

ES6: Módulos

Este concepto de obtener una sola propiedad de un objeto es **desestructuración.** Por lo que podemos usarlo tanto para exports nombrados como para objetos exportados.

```
// numero.js
export const numero = 2;

// nombre.js
export default {
  nombre: 'pepe',
};

// modulo.js
import { numero } from './numero';
import { nombre } from './nombre';

console.log(numero); // 2
console.log(nombre); // "pepe"
```

ES6: Módulos

Algo importante a tomar en consideración es que, debido a la naturaleza de la desestructuración y javascript, si tratamos de importar mediante imports con nombre una variable que no existe o escribimos mal el nombre de la propiedad, JS no tirará un error, sino que nos devolverá undefined.

```
// numero.js
export const numero = 2;

// modulo.js
import { numerow } from './numero';

console.log(numerow); // undefined
```

ES6: Módulos

Si bien funciona hacer un import normal (sin nombre), este sólo nos regresa lo que contiene export default.

Si tenemos un modulo que importa un default y otra variable; o un modulo con sólo exports con nombre no nos regresará nada.

```
// numero.js
export const numero = 2;

// modulo.js
import { numerow } from './numero';

console.log(numerow); // undefined
```

ES6: Módulos

Para estos casos, podemos obtener todos los exports de un módulo usando import * as.

```
// modulo1.js
export default 'Hola';
export const numero = 42;
export const nombre = 'pepe';

// modulo2js
import * as modulo1 from './modulo1';

console.log(modulo1.default); // "Hola"
console.log(modulo1.numero); // 42
console.log(modulo1.nombre); // "pepe"
```

ES6: Módulos

Lo que exportemos con export default será accesible mediante la propiedad default.

https://datyayu.xyz/es6-modulos/