

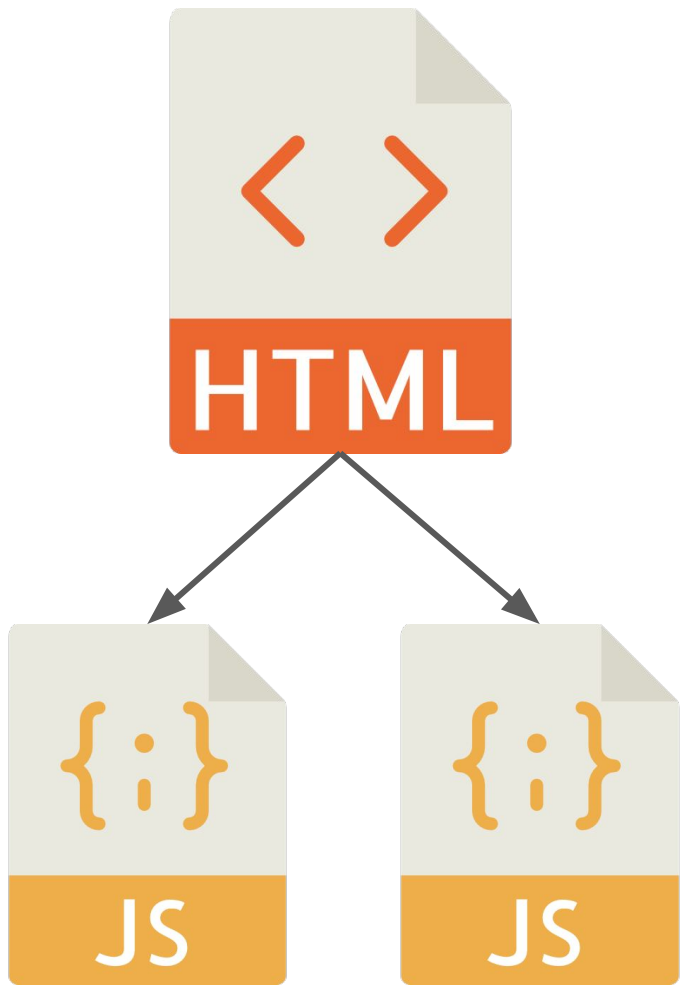
# Gestión de Módulos con JavaScript

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

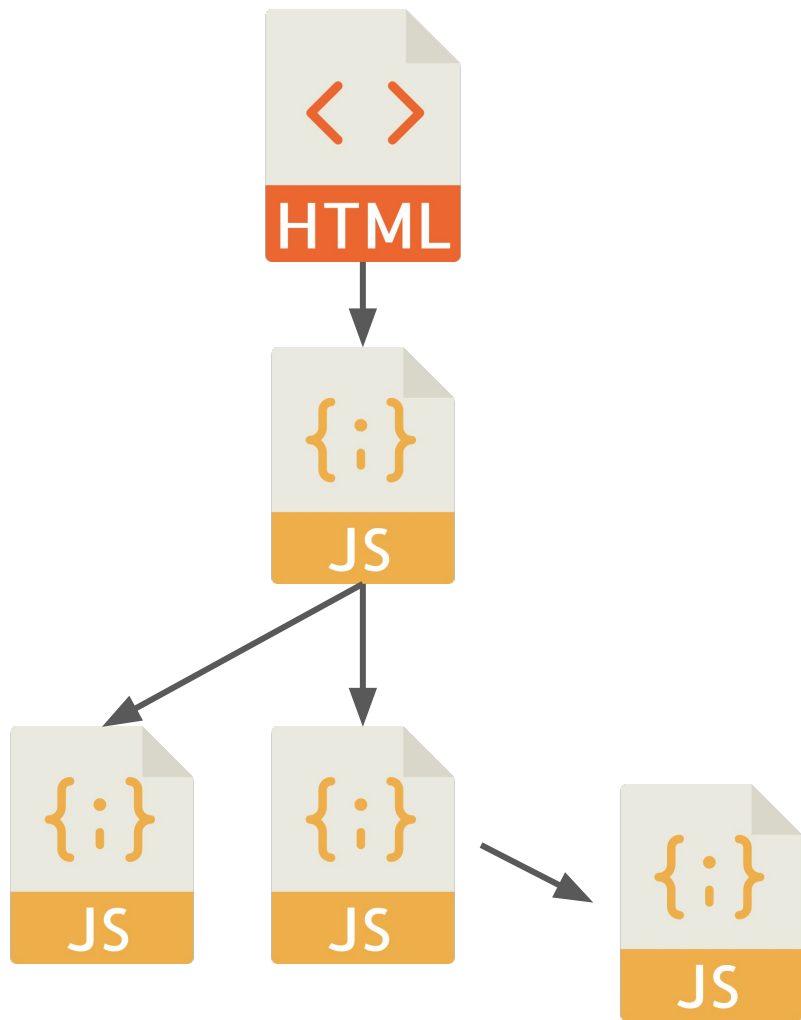
## Objetivos

1. *Conocer la evolución y uso de módulos en JavaScript.*
2. *Comprender la diferencia entre require, import.*
3. *Comprender la diferencia entre CommonJS y EcmaScript Modules (ESM), y cuando usar cada uno.*
4. *Aprender a dividir nuestra lógica de JavaScript en archivos.*



## Un Antecedente sobre Módulos (1)

Los programas JavaScript comenzaron siendo bastante pequeños, inicialmente eran fragmentos aislados que brindaban interactividad a las páginas web donde fuera necesario, por lo que generalmente no se necesitaban grandes scripts.



## Un Antecedente sobre Módulos (2)

Avancemos unos años y ahora tenemos aplicaciones completas que se ejecutan en navegadores con mucho JavaScript.

Por lo tanto, en los últimos años se ha comenzado a pensar en proporcionar mecanismos para dividir programas JavaScript en módulos separados que se puedan importar cuando sea necesario.

The logo consists of the text 'DEV.FX' in a bold, white, sans-serif font. The 'F' is stylized with three small squares to its right. The logo is centered within a dark blue diamond shape.

# DEV.FX

*Actualmente existen varias alternativas para importar módulos de JavaScript, pero vamos a centrarnos en 2:*

- *ES Modules (ES6)*
- *CommonJS*

# CommonJs (CJS) vs ES Modules (ESM)

Hoy en día, de las opciones disponibles, lo más común suele ser utilizar CommonJS o ESM.

En ecosistemas donde predomina la utilización de NodeJS, es más frecuente encontrarse usando **CommonJS**, mientras que en sistemas más modernos, de navegador o, por ejemplo, Deno, es más habitual utilizar el enfoque de ESM.

## CJS

CommonJS

```
const elem = require('module');  
module.exports = {  
};
```



node  
supported



sync



bare  
imports



browser  
compatible



from  
url/cdn



tree  
shaking



html  
supported



build  
oriented



dynamic  
require



deno  
supported

## ESM

ES Modules

```
import { elem } from './module.js';  
export const elem = {  
};
```



node  
supported



async



bare  
imports



browser  
compatible



from  
url/cdn



tree  
shaking



html  
supported



skypack  
compatible



dynamic  
imports



deno  
supported

# CommonJS vs ESM: Algunas Diferencias

- NodeJS soporta tradicionalmente la **sintaxis require de CommonJS**, y aunque cada vez soporta mejor ESM, aún el soporte no es completo y tiene una amplia comunidad con paquetes utilizando CommonJS a través de NPM.
- CommonJS sólo permite cargar módulos de forma **síncrona**, mientras que ESM permite carga **síncrona y asíncrona**.
- Los **require** de CommonJS no son compatibles en el navegador de forma **directa**, mientras que los import de ESM si lo son si se indica el atributo **<script type="module">** en los scripts que los utilicen.
- CommonJS no permite cargar directamente desde una URL o CDN un **módulo**, mientras que con ESM puedes **hacerlo sin problemas** y funciona directamente desde un navegador.

# Gestión de Módulos con ES Modules (ES6)

**DEV.FX**  
DESARROLLAMOS(PERSONAS);

dev



## Module Cheatsheet

### Name Export

```
export const name = 'value'
```

### Name Import

```
import { name } from '...'
```

### Default Export

```
export default 'value'
```

### Default Import

```
import anyName from '...'
```

### Rename Export

```
export { name as newName }
```

### Name Import

```
import { newName } from '...'
```

### Export List + Rename

```
export {  
  name1,  
  name2 as newName2  
}
```

### Import List + Rename

```
import {  
  name1 as newName1,  
  newName2  
} from '...'
```

📸 samanthaming

🌐 samanthaming.com

🐦 samantha\_ming

Referencia: [Samantha Ming \(s.f.\). Module CheatSheet](#)

## Formas de Exportar e Importar en ESM

1. Podemos exportar e importar un elemento por su nombre (name export/import).
2. Podemos usar un export default con alguna función o valor para que sea anónima y poder llamarla como queramos al momento de importar.
3. Podemos renombrar alguna variable al momento de exportar para importarla posteriormente con dicho nombre.
4. Podemos exportar una lista de elementos e importarlas de igual forma, esto como un objeto.

DEV.FX

# Resumen de uso Export/Import con ESM

## Exports/Imports (ES6)

### Default

A.js

```
export default 42
```

B.js

```
import A from './A'
```

### Named

A.js

```
export const A = 50
```

B.js

```
import { A } from './A'
```

### Starred

A.js

```
export default 42
```

```
export const A = 50
```

B.js

```
import * as Mix from './A'
```

## Ejemplo en código

```
// module.js
export const data = 42;
export const method = () => console.log("Hello");

// index.js
import { data, method } from "./module.js";
```

JS



# DEV.F

## IMPORTANTE

Si queremos usar ESM en conjunto con HTML, deberemos usar en nuestro archivo HTML el atributo:

**`<script type="module">`**

```
<script src="main.js" type="module">
```



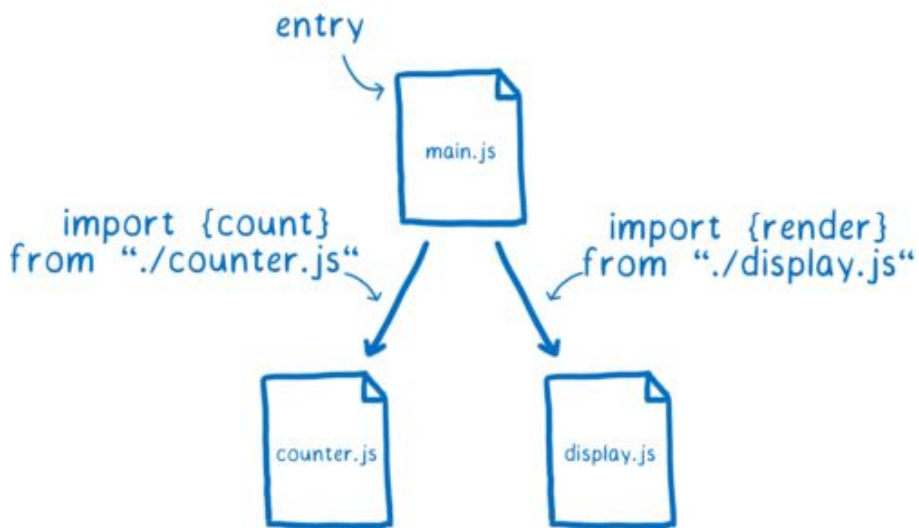
Este se usa para indicar cuándo se está apuntando a un módulo. En caso contrario obtendremos un error en la consola.

# Gestión de Módulos con CommonJS

**DEV.FX**  
DESARROLLAMOS(PERSONAS);

dev

# CommonJS



# Usando módulos con CommonJS

logger.js

```
const error = 'ERROR';
const warning = 'WARNING';
const info = 'INFO';

function log(message, level = info) {
  console.log(`${level}: ${message}`);
}

module.exports.log = log;
module.exports.error = error;
module.exports.info = info;
module.exports.warning = warning;
```

Exportar:  
module.exports

app.js

```
const {
  log,
  error,
  info,
  warning
} = require('./logger');

log('Node.js module demo 1');
log('Node.js module demo 2', warning);
```

Importar:  
require

## En Resumen...

CommonJS

`require`

`module.exports`

ES6

`import`

`export`



# Recapitulando...

## Require vs Import in JavaScript



BY CHAMEERA DULANGA

### FEATURES

Syntax

Used In

Asynchronous

Conditional Usage

Selective Load

Node Support

TypeScript Support

### REQUIRE

```
const x = require()
```

CommonJS  
Modules

✗

✓

✗

✓

✓

### IMPORT

```
import x from "./"
```

ES Modules

✓

✗

✓

V 13+

✓