# JavaScript Import Statement

## What is Import in JavaScript?

The import statement in JavaScript is used to bring in modules—or files containing JavaScript code—into another file.

## Syntax:

### 1. Default Import:

**import defaultExport from "module-name";**- Here, defaultExport refers to the default export from the imported module. You can name it anything you like.

### 2. Named Import:

**import { export1, export2 } from "module-name";**- export1 and export2 are specific named exports from the module. You must use the exact names that were used to export them in the original file.

### 3. Namespace Import:

**import \* as name from "module-name";**- Here, all exports from "module-name" are loaded into an object named name.

### 4. Aliasing:

**import { export1 as alias1, export2 as alias2 } from "module-name";**- Here, you rename the named exports export1 and export2 as alias1 and alias2, respectively.

### 5. Mixed Import:

**import defaultExport, { export1, export2 } from "module-name";**- This is a combination of default and named imports.

### 6. Importing Side-Effects Only:

**import "module-name"; // Used in case of CSS files**- This will execute the code in "module-name" but will not import any values.

## Additional Notes:

- Live Bindings: Imported variables are read-only and reflect changes made in the original exporting module.  
- Strict Mode: Files containing import are automatically in strict mode.  
- Dynamic Imports: You can also use dynamic import() as a function to load modules dynamically.  
- Host-Specified Evaluation: The way module-name is resolved can depend on the environment, like the browser or Node.js.  
- Valid Identifiers: All aliasing and variable names must be valid JavaScript identifiers.

## Example:

// math.js  
export const add = (x, y) => x + y;  
export const subtract = (x, y) => x - y;  
export default 'Math Library V1';  
  
// app.js  
import defaultExport, { add, subtract as sub } from './math';  
  
console.log(add(5, 3)); // Output: 8  
console.log(sub(5, 3)); // Output: 2  
console.log(defaultExport); // Output: 'Math Library V1'

# Import - Additional Details

## Placement Restrictions:

1. Top-Level Only: Import declarations can only be at the top level of a module and not nested inside any blocks or functions.  
2. Modules Only: Import declarations are only allowed in files that are treated as modules. This can mean that the file is specified with `type="module"` in an HTML script tag or that it has an `.mjs` extension in Node.js.