**Modules**

### Import Javascript modules with the require function

In Node.js, the require function can used to import code from another file into the current script.

var moduleA = require( "./module-a.js" ); // The .js extension is optional var moduleA = require( "./module-a" ); // Both ways will produce the same result. // Now the functionality of moduleA can be used console.log(moduleA.someFunctionality)

### Javascript export default

As of ES6, the export default keywords allow for a single variable or function to be exported, then, in another script, it will be straightforward to import the default export.

After using the export default it is possible to import a variable or function without using the require() function.

// module "moduleA.js" export default function cube(x) { return x \* x \* x; } // In main.js import cube from './moduleA.js'; // Now the `cube` function can be used straightforwardly. console.log(cube(3)); // 27

### Using the import keyword in Javascript

As of ES6, the import keyword can be used to import functions, objects or primitives previously exported into the current script.

There are many ways to use the import keyword, for example, you can import all the exports from a script by using the \* selector as follows: import \* from 'module\_name';.

A single function can be imported with curly brackets as follows: import {funcA} as name from 'module\_name';

Or many functions by name: import {funcA, funcB} as name from 'module\_name';

// add.js export const add = (x, y) => { return x + y } // main.js import { add } from './add'; console.log(add(2, 3)); // 5