**DINESH P**

**1MS24SCS08**

Assignment 1: Named Exports with Aliases

Objective: Practice using named exports and aliases during import.

Question:

Create a module mathOperations.js that exports the following functions using named exports:

add(a, b)

subtract(a, b)

In main.js, import these functions using aliases (e.g., add as sum, subtract as diff) and log the results of sample calculations.

// mathOperations.js

export function add(a, b) {

    return a + b;

}

export function subtract(a, b) {

    return a - b;

}

// main.js

import { add as sum, subtract as diff } from './mathOperations.js';

console.log('Sum of 5 and 3:', sum(5, 3));

console.log('Difference of 5 and 3:', diff(5, 3));

PS C:\Users\ADMIN\FSD> node .\main.js

Sum of 5 and 3: 8

Difference of 5 and 3: 2

PS C:\Users\ADMIN\FSD>

Assignment 2: Dynamic Imports

Objective: Understand how to load modules at runtime using import().

Question:

Create a module stringUtils.js that exports a function capitalize(word).

In main.js, use dynamic import to load stringUtils.js only when the user inputs a word. Display the capitalized word using the imported function.

// main.js

import readline from 'readline';

const rl = readline.createInterface({

    input: process.stdin,

    output: process.stdout

});

rl.question('Enter a word to capitalize: ', async (word) => {

    if (word) {

        const stringUtils = await import('./stringUtils.js');

        console.log('Capitalized:', stringUtils.capitalize(word));

    } else {

        console.log('No word entered.');

    }

    rl.close();

});

// stringUtils.js

export function capitalize(word) {

    if (!word) return '';

    return word.charAt(0).toUpperCase() + word.slice(1).toLowerCase();

}

PS C:\Users\ADMIN\FSD> node .\main.js

Enter a word to capitalize: murugan

Capitalized: Murugan

Assignment 3: Combining Default and Named Exports

Objective: Learn how to combine default and named exports in the same module.

Question:

Create a module user.js that:

Exports a default class User with properties like name and a method getInfo().

Also exports a named function validateEmail(email).

In main.js, import the class and the function, create a User instance, and validate a sample email address using the function.

// main.js

import User, { validateEmail } from './user.js';

// Create a User instance

const user = new User('Murugan');

console.log(user.getInfo());

// Validate a sample email address

const email = 'murugan@example.com';

console.log(`Is "${email}" a valid email?`, validateEmail(email));

// user.js

class User {

    constructor(name) {

        this.name = name;

    }

    getInfo() {

        return `User: ${this.name}`;

    }

}

function validateEmail(email) {

    // Simple email validation regex

    return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);

}

export default User;

export { validateEmail };

PS C:\Users\ADMIN\FSD> node .\main.js

User: Murugan

Is "murugan@example.com" a valid email? true

PS C:\Users\ADMIN\FSD>

**REPL**

The Node.js REPL (Read-Eval-Print Loop) is an interactive command-line environment that allows developers to execute JavaScript code directly. It provides a fast and efficient way to test snippets, debug code, and explore Node.js features without the need to create separate files or run complete applications. This environment is especially useful for quick experimentation, prototyping, and learning.

REPL follows a cycle of:

* **Read**: The REPL reads the user's input, parsing it into JavaScript data structures.
* **Eval**: The parsed input is evaluated.
* **Print**: The result of the evaluation is printed to the console.
* **Loop**: The cycle repeats until explicitly exited.

let list = [];

console.log("Simple Array REPL (JavaScript)");

console.log("Commands: add <item>, remove <item>, print, exit");

function repl() {

const command = prompt(">>").trim();

if (command === 'exit') {

console.log("Exiting REPL. Bye!");

return;

} else if (command.startsWith('add ')) {

const item = command.slice(4);

list.push(item);

console.log(`Added '${item}' to the list.`);

} else if (command.startsWith('remove ')) {

const item = command.slice(7);

const index = list.indexOf(item);

if (index !== -1) {

list.splice(index, 1);

console.log(`Removed '${item}' from the list.`);

} else {

console.log(`'${item}' not found in the list.`);

}

} else if (command === 'print') {

if (list.length === 0) {

console.log("List is empty.");

} else {

console.log("Current list:");

list.forEach((item, index) => {

console.log(`${index}: ${item}`);

});

}

} else {

console.log("Unknown command.");

}

// Repeat the prompt

repl();

}

// Start the REPL loop

repl();

output:

Simple Array REPL (JavaScript)

Commands: add <item>, remove <item>, print, exit

>> hello

Unknown command.

>> add 1

Added '1' to the list.

>> remove 3

'3' not found in the list.

>> print

Current list:

0: 1

>> add apple

Added 'apple' to the list.

>> print

Current list:

0: 1

1: apple

>> remove 1

Removed '1' from the list.

>> print

Current list:

0: apple

>> exit

Exiting REPL. Bye!