



ACADEMY
OF DIGITAL ARTS
EGYPT



Adobe

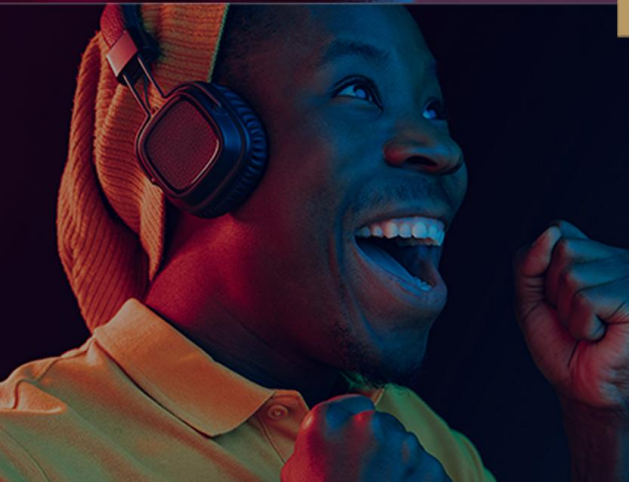


Microsoft

CompTIA.



START
YOUR TECH JOURNEY
WITH ADA



Node.js Session 2 - Student Task

Academy of Digital Arts Egypt - Back-End Development Course

Objective:

By completing this task, you will:


- Set up and use ES modules with import/export syntax in Node.js
- Understand the difference between default and named exports
- Apply ES6+ features: template literals, destructuring, and spread operators
- Create and handle Promises using .then() and .catch() methods
- Write clean asynchronous code using async/await with proper error handling
- Build a complete task management system using modern Node.js patterns



Node.js Session 2 - Student Task

Academy of Digital Arts Egypt - Back-End Development Course

Step 1 – Set Up ES Modules Environment

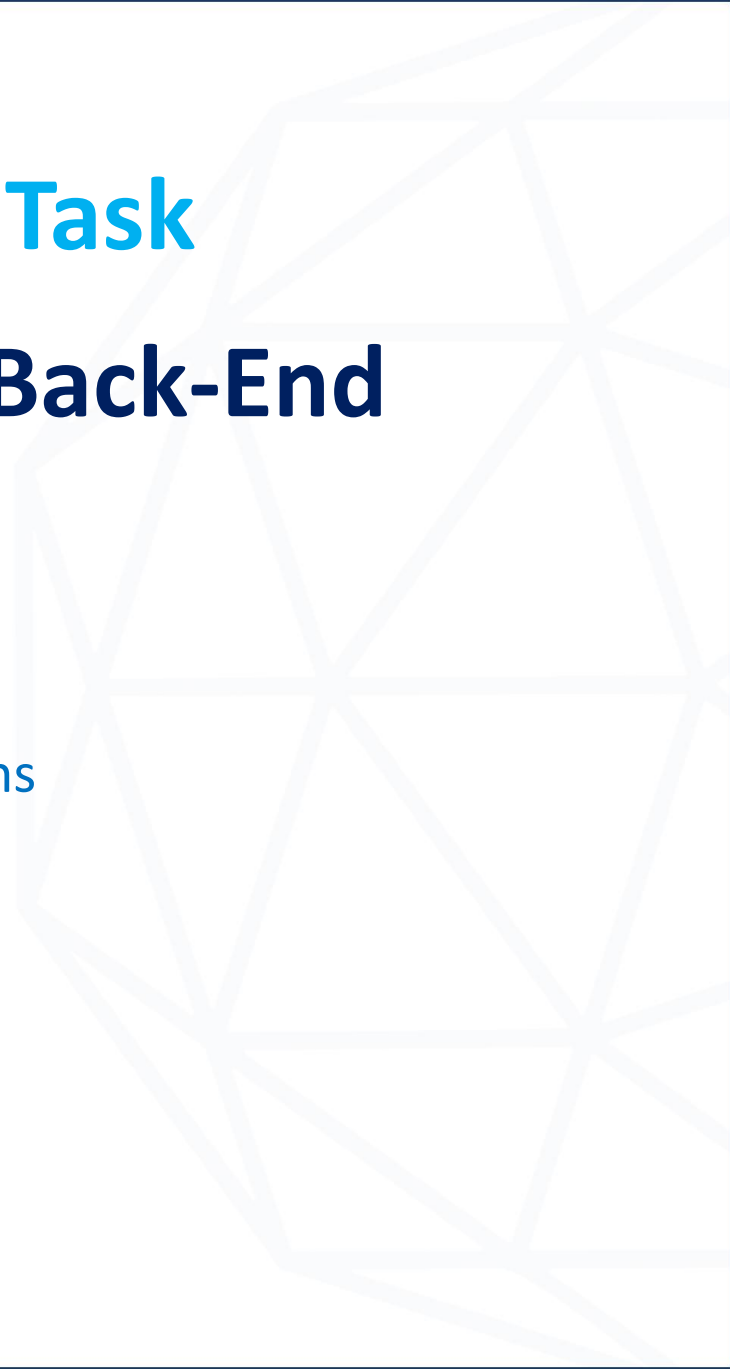
- Create package.json with "type": "module" to enable ES modules
 - Build task.js module with Task class and export it as default
 - Create taskService.js with named exports for task management functions
 - Use import/export syntax instead of require/module.exports
- 



Node.js Session 2 - Student Task

Academy of Digital Arts Egypt - Back-End Development Course

Step 2 – Apply ES6+ Features

- Use template literals for dynamic string formatting in task descriptions
 - Apply destructuring to extract properties from task objects
 - Implement spread operator to merge task data and create copies
 - Use arrow functions and modern JavaScript syntax throughout
- 

Node.js Session 2 - Student Task

Academy of Digital Arts Egypt - Back-End Development Course

Step 3 – Implement Promise-Based Operations


- Create Promise-based functions for task operations (create, read, update, delete)
- Use `setTimeout` to simulate async database operations
- Handle success cases with `.then()` and errors with `.catch()`
- Chain multiple Promise operations together



Node.js Session 2 - Student Task

Academy of Digital Arts Egypt - Back-End Development Course

Step 4 – Convert to Async/Await Pattern

- Rewrite Promise chains using async/await syntax for cleaner code
 - Implement proper error handling with try/catch blocks
 - Create async functions for sequential and parallel task operations
 - Build a main application that demonstrates all concepts working together
- 




Node.js Session 2 - Student Task

Academy of Digital Arts Egypt - Back-End Development Course

Requirements

Tools:

- Node.js environment with ES modules support (Node.js 14+)
 - Text editor (VS Code recommended)
 - Terminal access for running Node.js applications
- 

Node.js Session 2 - Student Task

Academy of Digital Arts Egypt - Back-End Development Course

Reminder

- Add "type": "module" to package.json to enable ES modules
- Use .js extension in import statements: `import './task.js'`
- Default exports: `export default ClassName`, `import ClassName from './file.js'`
- Named exports: `export { function1, function2 }`, `import { function1 } from './file.js'`
- Always use try/catch with async/await for error handling
- Use await only inside async functions
- Template literals use backticks: `Hello ${name}`

THANK YOU

ADAEGY     