Learning Material: Code Reusability

Introduction to Code Reusability

Code reusability is one of the foundational principles in modern programming. It refers to writing code in a way that allows it to be used multiple times without rewriting it. Reusable code reduces development time, improves maintainability, and ensures consistency across a program or even multiple programs.

Code reusability is achieved mainly through **functions** and **modules**, which help organize code into logical, reusable components.

What Is Code Reusability?

Code reusability means creating code components that can perform specific tasks and be used repeatedly in different parts of a program.

Benefits of Code Reusability:

- **Efficiency:** Write once, use many times.
- **Maintainability:** Fixing a bug in one place updates all usages.
- **Cleaner Code:** Reduces duplication and clutter.
- **Scalability:** Easier to extend or modify systems.

---

3. Functions as Reusable Code Blocks

A **function** is a named block of code that performs a specific task.

Why Use Functions?

- They break large programs into manageable parts.
- Improve readability.
- Allow repeated execution of the same logic.
- Reduce errors because logic is centralized.

Structure of a Function

```
function functionName(parameters) {
    // code to execute
    return value;
}
```

Example: Simple Greeting Function

```
function greet(name) {
    return "Hello, " + name + "!";
}

print(greet("Ana"));
print(greet("Mark"));
```

The function is reusable because it works for any input name.

Types of Functions Supporting Reusability

- **Built-in Functions:** Provided by the programming language (e.g., `print()`, `len()`)
- **User-Defined Functions:** Created by programmers
- **Recursive Functions:** Call themselves for repeated tasks
- **Higher-order Functions:** Accept other functions as arguments

---

4. Modules for Organizing Reusable Code

A **module** is a file containing functions, classes, or variables that can be imported into other programs.

Modules allow programmers to organize code logically and reuse components across multiple files or projects.

Why Use Modules?

- Separate code into logical parts
- Reduce file size and complexity

- Share reusable code across applications
- Encourage teamwork: multiple developers can work on different modules

---

Example of a Module

math_utils.js (Module file)

```
function add(a, b) {
    return a + b;
}

function multiply(a, b) {
    return a * b;
}

export { add, multiply };
```

main.js (Using the module)

```
import { add, multiply } from './math_utils.js';

console.log(add(3, 4));
console.log(multiply(5, 2));
```

This allows reuse of math functions across multiple files.

Writing Reusable Functions and Modules

A reusable component should follow these principles:

Single Responsibility

Each function or module should do only **one thing well**.

Parameterization

Make functions flexible by using parameters:

```
function area(width, height) {
    return width * height;
}
```

Avoid Hardcoding

Hardcoded values reduce reusability.

Documentation & Naming

Good names and comments make code easier to reuse and understand.

Minimize Side Effects

Reusable functions should not change external variables unexpectedly.

Real-World Examples of Code Reusability

1. Libraries

Collections of reusable functions—like math libraries, UI libraries, and networking libraries.

2. Frameworks

Large reusable templates for building applications (e.g., React, Django).

3. APIs

Reusable endpoints that different apps can call.

4. Utility Modules

Files containing commonly used helpers such as random number generators, date formatting, or string processors.

Advantages in Software Development

1. Faster Development

Reusing tested code accelerates building applications.

2. Reduced Errors

Using the same logic in multiple places avoids inconsistencies.

3. Better Collaboration

Teams can develop modules independently.

4. Easier Maintenance

Updating a function updates all modules that use it.