TypeScript Export and Import Guide

In TypeScript (and JavaScript ES Modules), the 'export' and 'import' keywords are used to share code between files. They help organize large projects by splitting code into smaller, reusable modules.

Example 1: Exporting and Importing a Function

```
// mathUtils.ts
export function add(a: number, b: number): number {
  return a + b;
}

export function subtract(a: number, b: number): number {
  return a - b;
}

// app.ts
import { add, subtract } from "./mathUtils";
console.log(add(10, 5));      // 15
console.log(subtract(10, 5)); // 5
```

Example 2: Exporting a Class

```
// Person.ts
export class Person {
  constructor(public name: string, public age: number) {}

  greet(): void {
    console.log(`Hello, I'm ${this.name}, ${this.age} years old.`);
  }
}

// main.ts
import { Person } from "./Person";
const p1 = new Person("Alice", 25);
p1.greet();
```

Example 3: Default Export

```
// logger.ts
export default function logMessage(message: string): void {
  console.log(`[LOG]: ${message}`);
}

// main.ts
import logMessage from "./logger";
logMessage("Application started successfully.");
```

Example 4: Mixing Default and Named Exports

```
// config.ts
export const API_URL = "https://api.example.com";
export function connect(): void {
  console.log("Connected to API");
}
export default { timeout: 5000, retries: 3 };

// main.ts
import config, { API_URL, connect } from "./config";
console.log("URL:", API_URL);
connect();
console.log("Default Config:", config);
```

Example 5: Re-exporting from Another Module

```
// shapes/Circle.ts
export class Circle {
  constructor(public radius: number) {}
  area() {
    return Math.PI * this.radius ** 2;
  }
}

// shapes/Square.ts
export class Square {
  constructor(public side: number) {}
  area() {
    return this.side * this.side;
  }
}

// shapes/index.ts
export * from "./Circle";
export * from "./Square";

// main.ts
import { Circle, Square } from "./shapes";
const c = new Circle(5);
const s = new Square(4);
console.log("Circle area:", c.area());
console.log("Square area:", s.area());
```

Example 6: Renaming Imports/Exports using 'as'

```
// math.ts
export function add(a: number, b: number) {
  return a + b;
}
export function multiply(a: number, b: number) {
  return a * b;
}

// main.ts
import { add as sum, multiply as product } from "./math";
console.log(sum(3, 4));     // 7
console.log(product(3, 4)); // 12
```

Example 7: Importing Everything from a Module

```
// tools.ts
export function greet() {
  console.log("Hello!");
}
export function bye() {
  console.log("Goodbye!");
}

// main.ts
import * as Tools from "./tools";
Tools.greet();
Tools.bye();
```

Summary

- 'export' makes a variable, class, or function available outside its file. - 'export default' is used when a file exports one main thing. - 'import' brings in exported code from another module. - You can import everything, rename imports, or re-export modules for better organization.