***Code Migration Exercise***

Converting a JavaScript project to TypeScript involves several steps, from setting up TypeScript in your project to gradually adding type annotations and resolving type errors. Here’s a thorough step-by-step guide to help you through this process:

### **Step 1: Setting Up TypeScript**

**Install TypeScript**: Start by installing TypeScript and the necessary types for your project. Use npm to install TypeScript:  
  
npm install --save-dev typescript

**Initialize TypeScript Configuration**: Create a tsconfig.json file to configure TypeScript:  
  
npx tsc --init

**Configure tsconfig.json**: Open the tsconfig.json file and configure it according to your project needs. A basic configuration might look like this:  
  
{

"compilerOptions": {

"target": "ES6",

"module": "commonjs",

"strict": true,

"esModuleInterop": true,

"skipLibCheck": true,

"forceConsistentCasingInFileNames": true,

"outDir": "./dist",

"rootDir": "./src"

},

"include": ["src/\*\*/\*"],

"exclude": ["node\_modules", "dist"]

}

### **Step 2: Converting JavaScript Files to TypeScript**

**Rename JavaScript Files**: Change the file extension of your JavaScript files from .js to .ts. For example, rename index.js to index.ts.

**Start with Simple Type Annotations**: Begin by adding type annotations to your files. TypeScript can infer many types, but explicit annotations help with clarity and type safety.

### **Step 3: Adding Type Annotations**

**Function Parameters and Return Types**: Add type annotations to function parameters and return types.  
  
// Before (JavaScript)

function greet(name) {

return `Hello, ${name}!`;

}

// After (TypeScript)

function greet(name: string): string {

return `Hello, ${name}!`;

}

**Variable Types**: Specify types for variables.

// Before (JavaScript)

let age = 30;

// After (TypeScript)

let age: number = 30;

### **Step 4: Handling Existing Code Patterns**

**Any Type**: If you encounter complex types or third-party libraries without types, use any as a temporary measure.  
  
let user: any = getUser();

**Third-Party Libraries**: Install type definitions for third-party libraries. For example, for Express, you can install types as follows:  
  
npm install --save-dev @types/express

### **Step 5: Resolving Conversion Issues**

**Fix Type Errors**: As you add types, TypeScript might show errors. Address these errors by refining your type annotations and ensuring compatibility.

**Type Assertion**: Use type assertions when you know more about a type than TypeScript.

let input = document.querySelector('input') as HTMLInputElement;

**Union and Intersection Types**: Use union (|) and intersection (&) types to handle variables that can have multiple types.  
  
let value: string | number;

value = "Hello";

value = 42;

### **Step 6: Building and Testing**

**Compile TypeScript**: Compile your TypeScript code to JavaScript:  
  
npx tsc

**Run Tests**: Ensure your tests pass. If you use a testing framework like Jest, install its TypeScript types:  
  
npm install --save-dev ts-jest @types/jest

### **Step 7: Continuous Integration and Linting**

**Integrate with CI/CD**: Update your CI/CD pipeline to run TypeScript compilation and tests.

**Linting**: Use a linter to enforce code style. ESLint with TypeScript support can be set up as follows:  
  
npm install --save-dev eslint @typescript-eslint/parser @typescript-eslint/eslint-plugin

Configure .eslintrc.json:

{

"parser": "@typescript-eslint/parser",

"plugins": ["@typescript-eslint"],

"extends": ["plugin:@typescript-eslint/recommended"]

}

### **Example Project Walkthrough**

Let's walk through a simple project conversion:

#### **Original JavaScript Project**

src/index.js:  
  
const express = require('express');

const app = express();

app.get('/', (req, res) => {

res.send('Hello, World!');

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

#### **Step-by-Step Conversion**

1. **Rename File**: Rename src/index.js to src/index.ts.

**Install Types**:  
  
npm install --save-dev @types/express

1. **Update Code with Types**:  
     
   import express, { Request, Response } from 'express';

const app = express();

app.get('/', (req: Request, res: Response) => {

res.send('Hello, World!');

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

1. **Compile and Test**:  
   bash  
   Copy code  
   npx tsc

node dist/index.js

### **Conclusion**

Migrating a JavaScript project to TypeScript enhances code quality and maintainability. By following these steps, you can systematically convert your project, ensuring a smooth transition with minimal disruption.