# First Steps with TypeScript – Question & Answers

Question:

1. What is the main benefit of using TypeScript over JavaScript?
2. What is the purpose of the ‘*interface’* keyword in TypeScript?
3. How do you declare a variable with a specific type in TypeScript?
4. What is the difference between the ‘*any’* and ‘*unknown’* types in TypeScript?
5. What is the purpose of the tsconfig.json file in TypeScript project?

Answers:

1. The main benefit of using TypeScript over JavaScript is **static typing,** a feature of TypeScript where the data types of variables, function parameters, and return values are known and checked at compile time. Therefore, the number of errors at runtime reduces.  
   Whereas in JavaScript types are determined at runtime which can cause more runtime errors.
2. The **‘*interface*’** keyword in TypeScript is used to define custom data types. Interfaces allow you to create a blueprint for an object, defining the shape and structure of that object, including the properties and methods it should have.  
   For example:  
   interface Point {  
    x: number;

y: number;

}

1. Declaring a variable with a specific type in TypeScript is done by using type annotation syntax. This is done by adding a colon (:) after the variable name, followed by the type.  
   For example:   
   **let** idNumber : number = 12542;  
   The meaning of declaring a variable with a specific type is the variable will not be able to be assigned later with a different type.

1. The any and unknown types in TypeScript have the following key differences:
   1. **Safety:  
      any:**  
      This type provides the least amount of type safety.   
       A variable of type any can be assigned any value, and you can call any method on it without TypeScript throwing an error.

**unknown:**   
This type provides more type of safety than any. Variables of type unknown cannot be used until they are narrowed down to a more specific type.

* 1. **Type Inference:  
     any:**

When a variable is not explicitly typed, TypeScript will often default to inferring the any type, especially when the type cannot be determined.

**unknown:**

TypeScript will not automatically infer the unknown type. You must explicitly declare a variable as unknown if you want to use this type.

* 1. **Type Checking:  
     any:**

TypeScript will not perform any type checking on variables of type ‘any’. They can be treated as if they have no type at all.

**unknown:**

TypeScript will perform type checks on unknown variables before allowing you to use them. You must narrow the type down to a more specific type before you can safely access its properties or call its methods.

* 1. **Type Safety:  
     any:**

Using ‘any’ can lead to type-unsafe code, as it allows you to bypass TypeScript's type system entirely.

**unknown:**

Using ‘unknown’ encourages type safety, as you must explicitly narrow the type before using the variable.

1. The tsconfig.json file is a configuration file for TypeScript projects that specifies how the TypeScript compiler should process your code. It serves several important purposes:
   1. **Compiler Options**:  
      - Defines how TypeScript should compile your code.  
      - Sets the JavaScript version to target.  
      - Controls type-checking strictness.   
      - Specifies output directory for compiled files.
   2. **Project Structure**:  
      - Determines which files should be included/excluded in

compilation.  
- Defines the root directory of your source files.  
- Specifies file patterns to ignore.

* 1. **Module Resolution**:  
     - Controls how TypeScript resolves module imports.  
     - Sets up path aliases for imports.  
     - Configures module resolution strategy.
  2. **Type Checking Rules**:  
     - Enables/disables specific type checking features.  
     - Controls how strict the compiler should be.  
     - Sets rules for null/undefined handling.
  3. **Project References**:  
     - Defines relationships between different TypeScript projects.  
     - Enables building dependent projects in the correct order.

Without this file, these options should be defined manually when running the TypeScript compiler, which would be impractical for larger projects and could lead to inconsistencies across your development team.