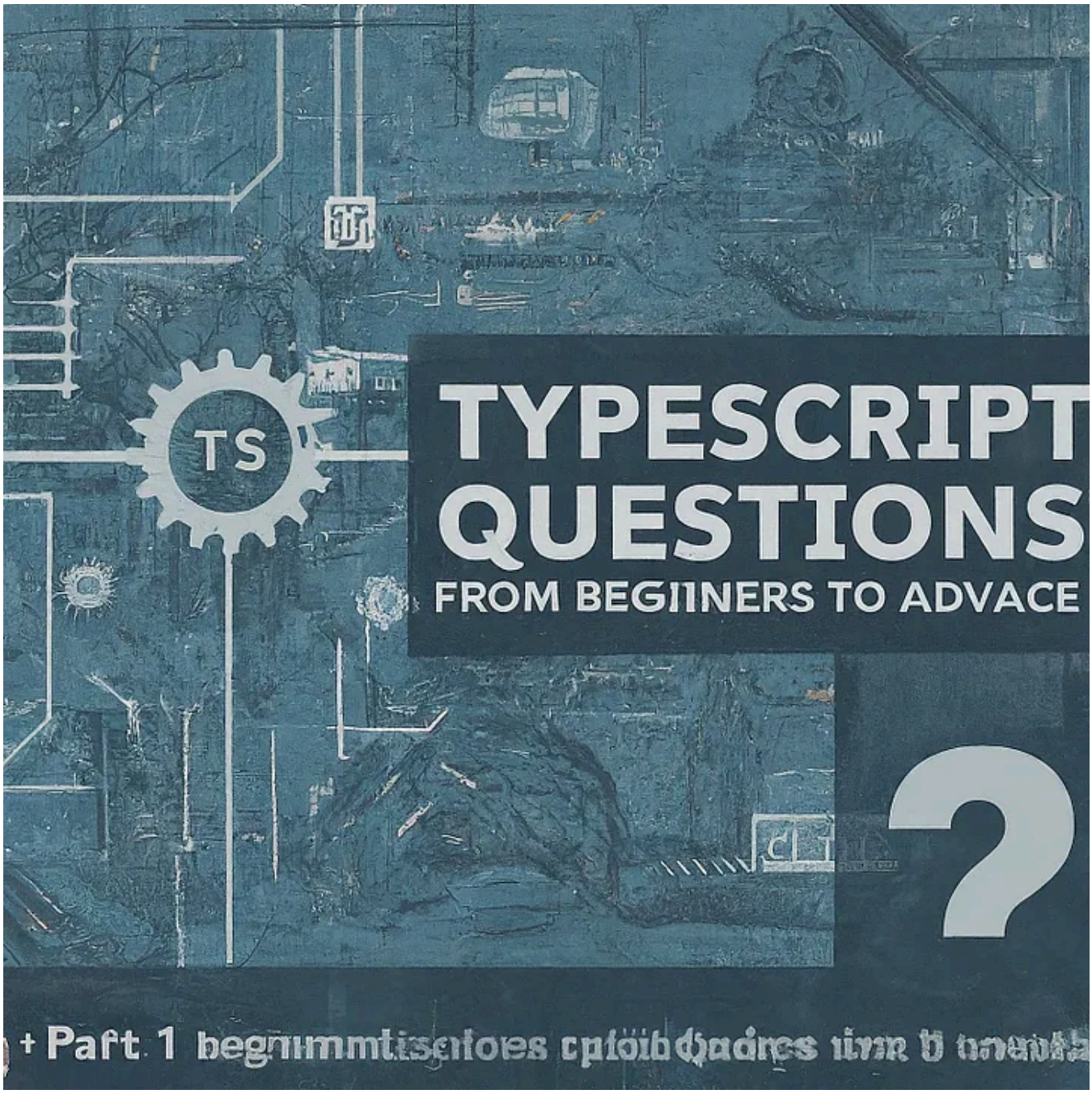
TypeScript Interview Questions: From Beginners to Advanced Part 1

4 min read · May 19, 2024

Pawan Kumar Follow

Listen Share



TypeScript, a superset of JavaScript, offers static typing along with other powerful features to help developers build robust, maintainable, and scalable applications. Whether you're a beginner starting with TypeScript or preparing for an interview, understanding the fundamental concepts is crucial. This article covers essential beginner-level TypeScript interview questions, complete with answers and examples.

1. What is TypeScript and how is it different from JavaScript?

Answer: TypeScript is an open-source programming language developed and maintained by Microsoft. It is a superset of JavaScript, which means it includes all JavaScript features and adds additional features such as static types, classes, and interfaces.

Example:

```
let message: string = 'Hello, TypeScript!';
console.log(message);
```

In this example, the variable `message` is explicitly typed as a `string`.

2. How do you install TypeScript?

Answer: TypeScript can be installed globally using npm (Node Package Manager) with the following command:

```
npm install -g typescript
```

After installation, you can verify it by checking the version:

```
tsc -v
```

3. What are the benefits of using TypeScript?

Answer: The benefits of using TypeScript include:

- Static Typing: Helps catch errors during development.

- Improved IDE Support: Offers better code completion, navigation, and refactoring.
- Enhanced Code Readability and Maintainability: Types and interfaces make the code more self-documenting.
- Early Bug Detection: Errors are caught at compile-time rather than runtime.
- Compatibility with JavaScript: TypeScript code can be compiled to plain JavaScript, ensuring compatibility with existing JavaScript libraries and frameworks.

4. How do you compile TypeScript code?

Answer: TypeScript code is compiled using the TypeScript compiler, `tsc`. You can compile a TypeScript file with the following command:

```
tsc filename.ts
```

This generates a JavaScript file that can be run in any JavaScript environment.

5. What is a TypeScript configuration file (`tsconfig.json`)?

Answer: `tsconfig.json` is a configuration file for TypeScript projects. It specifies the root files and the compiler options required to compile the project.

Example:

```
{
  "compilerOptions": {
    "target": "ES6",
    "module": "commonjs",
    "outDir": "./dist",
    "strict": true
  },
  "include": ["src/**/*"]
}
```

In this example, TypeScript is configured to compile the code to ES6, use the CommonJS module system, output files to the `dist` directory, enforce strict type-checking, and include all files in the `src` directory.

6. What are basic types in TypeScript?

Answer: TypeScript supports various basic types such as:

- `boolean`
- `number`
- `string`
- `array`
- `tuple`
- `enum`
- `any`
- `void`
- `null` and `undefined`
- `never`

Example:

```
let isDone: boolean = false;
let age: number = 25;
let firstName: string = 'John';
let hobbies: string[] = ['Reading', 'Gaming'];
```

7. How do you define an interface in TypeScript?

Answer: An interface in TypeScript defines the structure of an object, specifying its properties and their types.

Example:

```
interface Person {
  firstName: string;
  lastName: string;
  age: number;
```

```
}

let user: Person = {
  firstName: 'Jane',
  lastName: 'Doe',
  age: 30
};
```

8. What are TypeScript enums and how do you use them?

Answer: Enums in TypeScript allow you to define a set of named constants. Enums can be numeric or string-based.

Example:

```
enum Direction {
  Up,
  Down,
  Left,
  Right
}

let move: Direction = Direction.Up;
```

9. How do you use type assertions in TypeScript?

Answer: Type assertions allow you to override the inferred type of an expression. This is useful when you know more about the type than TypeScript's type checker.

Example:

```
let someValue: any = 'Hello, World!';
let strLength: number = (someValue as string).length;
```

10. What is the difference between `any` and `unknown` types in TypeScript?

Answer:

- `any` disables all type checking for a variable, allowing it to be assigned any type.

- `unknown` is a type-safe counterpart to `any`. It requires a type assertion or check before it can be used as a specific type.

Example:

```
let anyValue: any = 'Hello';
let unknownValue: unknown = 'World';

// No error for `any` type
anyValue = 123;

// Error for `unknown` type without type assertion
// let str: string = unknownValue; // Error

if (typeof unknownValue === 'string') {
  let str: string = unknownValue; // No error
}
```

11. How do you define a function in TypeScript?

Answer: Functions in TypeScript can be defined with type annotations for parameters and return types.

Example:

```
function greet(name: string): string {
  return `Hello, ${name}!`;
}

console.log(greet('Alice'));
```

12. What are optional parameters in TypeScript functions?

Answer: Optional parameters can be specified by using a question mark (`?`) after the parameter name.

Example:

```
function greet(name: string, greeting?: string): string {
  return `${greeting || 'Hello'}, ${name}!`;
```

```
}

console.log(greet('Alice')); // Hello, Alice!
console.log(greet('Bob', 'Hi')); // Hi, Bob!
```

13. What are default parameters in TypeScript?

Answer: Default parameters allow you to initialize parameters with default values if no value or `undefined` is passed.

Example:

```
function greet(name: string, greeting: string = 'Hello'): string {
  return `${greeting}, ${name}!`;
}

console.log(greet('Alice')); // Hello, Alice!
console.log(greet('Bob', 'Hi')); // Hi, Bob!
```

14. How do you use union types in TypeScript?

Answer: Union types allow a variable to be one of several types, using the pipe (|) symbol.

Example:

```
let value: string | number;
value = 'Hello';
value = 123;

function printId(id: number | string) {
  console.log(`ID: ${id}`);
}

printId(101); // ID: 101
printId('202'); // ID: 202
```

15. What is a tuple in TypeScript and how do you define one?

Answer: Tuples are a special type of array where the type of each element is known and fixed.

Example:

```
let tuple: [string, number];
tuple = ['Alice', 30];

console.log(tuple[0]); // Alice
console.log(tuple[1]); // 30
```

Conclusion

TypeScript is a powerful language that enhances JavaScript by adding type safety and other features. Understanding these beginner-level questions is the first step toward mastering TypeScript and preparing for interviews. In the next parts of this series, we will cover intermediate and advanced TypeScript interview questions.

For more advanced questions, be sure to check out the following parts of this series:

- **TypeScript Interview Questions: Intermediate Level Part 2**
- **TypeScript Interview Questions: Advanced Level Part 3**