### What is TypeScript?

1. **What is TypeScript?**
   - a) A programming language for styling web pages
   - b) A superset of JavaScript that adds static types
   - c) A database query language
   - d) A framework for building mobile apps
   - **Answer: b)**

2. **TypeScript is maintained by which company?**
   - a) Google
   - b) Apple
   - c) Microsoft
   - d) Facebook
   - **Answer: c)**

Benefits of TypeScript

3. **Which of the following is a benefit of using TypeScript?**
   - a) Dynamic typing

  - b) Cross-platform applications

  - c) Early bug detection through static type checking

  - d) Faster execution than JavaScript

  - **Answer: c)**

4. **TypeScript helps in catching errors at what stage?**

  - a) Runtime

  - b) Compile-time

  - c) Deployment

  - d) Testing

  - **Answer: b)**

### Setup the Environment

5. **Which command is used to install TypeScript via npm?**

  - a) `npm install ts`

  - b) `npm install typescript`

  - c) `npm install ts -g`

- d) `npm install typescript -g`
- **Answer: d)**

6. **What file extension is used for TypeScript files?**
- a) .js
- b) .java
- c) .ts
- d) .tsx
- **Answer: c)**

### Basic Data Types

7. **Which of the following is not a basic data type in TypeScript?**
- a) String
- b) Number
- c) Symbol
- d) Character
- **Answer: d)**

8. **What keyword is used to define a variable with a specific type?**

   - a) type
   - b) var
   - c) let
   - d) const
   - **Answer: a)**

### Arrays

9. **How do you define an array of numbers in TypeScript?**

   - a) `let arr: number[];`
   - b) `let arr: Array<number>;`
   - c) Both a and b
   - d) None of the above
   - **Answer: c)**

10. **Which method adds an element to the end of an array in TypeScript?**

    - a) `push()`

- b) `pop()`
- c) `shift()`
- d) `unshift()`
- **Answer: a)**

### Tuples

11. **What is a tuple in TypeScript?**
    - a) A collection of elements of the same type
    - b) A fixed-size collection of elements of mixed types
    - c) A variable-size collection of elements of mixed types
    - d) A method to iterate over arrays
    - **Answer: b)**

12. **How do you declare a tuple in TypeScript?**
    - a) `let tuple: [string, number];`
    - b) `let tuple: {string, number};`
    - c) `let tuple: (string, number);`
    - d) `let tuple: Array<string, number>;`

   - **Answer: a)**

### Enum

13. **What is an enum in TypeScript used for?**
   - a) To define a set of named constants
   - b) To create a new type
   - c) To iterate over arrays
   - d) To perform type casting
   - **Answer: a)**

14. **How do you declare an enum in TypeScript?**
   - a) `enum Direction { North, South, East, West }`
   - b) `enum Direction = { North, South, East, West }`
   - c) `enum Direction [ North, South, East, West ]`
   - d) `enum Direction < North, South, East, West >`
   - **Answer: a)**

### Any and void

15. **What is the `any` type in TypeScript?**

- a) It represents a value of a specific type
- b) It represents a value of any type
- c) It represents a void value
- d) It represents an undefined value
- **Answer: b)**

16. **What is the `void` type in TypeScript used for?**
- a) To represent functions that return a value
- b) To represent functions that do not return a value
- c) To represent any type of value
- d) To represent null values
- **Answer: b)**

null and undefined

17. **What is the difference between `null` and `undefined` in TypeScript?**
- a) `null` is a value assigned by the programmer, `undefined` is assigned by the compiler

- b) `null` means a variable is declared but not initialized, `undefined` means a variable is not declared

- c) `null` is an object, `undefined` is a primitive type

- d) There is no difference

- **Answer: a)**

18. **How do you check if a variable is `null` or `undefined` in TypeScript?**

- a) `if (variable == null)`

- b) `if (variable === null)`

- c) `if (variable == undefined)`

- d) Both a and c

- **Answer: d)**

### Type Inference

19. **What is type inference in TypeScript?**

- a) Manually specifying the type of a variable

- b) Automatically determining the type of a variable based on its value

- c) Casting a variable from one type to another
- d) Specifying the return type of a function
- **Answer: b)**

20. **When does TypeScript infer the type of a variable?**
    - a) When the variable is declared without a type
    - b) When the variable is declared with a type
    - c) When the variable is cast to a different type
    - d) TypeScript does not infer types
    - **Answer: a)**

### Type Casting

21. **What is type casting in TypeScript?**
    - a) Converting a variable from one type to another
    - b) Assigning a type to a variable
    - c) Inferring the type of a variable
    - d) Checking the type of a variable
    - **Answer: a)**

22. **How do you perform type casting in TypeScript?**

   - a) `let value: number = <number>variable;`

   - b) `let value: number = (variable as number);`

   - c) Both a and b

   - d) None of the above

   - **Answer: c)**

### Difference between let and var

23. **What is the main difference between `let` and `var` in TypeScript?**

   - a) `let` is block-scoped, `var` is function-scoped

   - b) `let` is function-scoped, `var` is block-scoped

   - c) `let` is used for constants, `var` is used for variables

   - d) There is no difference

   - **Answer: a)**

24. **Which keyword should you use to declare a variable that will not change its value?**

   - a) var

- b) let

- c) const

- d) type

- **Answer: c)**

### Const declaration

25. **What does the `const` keyword declare in TypeScript?**

- a) A variable that can be reassigned

- b) A variable that cannot be reassigned

- c) A block-scoped variable

- d) A variable with a specific type

- **Answer: b)**

26. **Can the value of a `const` object be changed in TypeScript?**

- a) No

- b) Yes, but the object cannot be reassigned

- c) Yes, the object and its properties can be changed

   - d) No, neither the object nor its properties can be changed

   - **Answer: b)**

### Writing and Using Classes

27. **How do you define a class in TypeScript?**

   - a) `class MyClass { }`

   - b) `class: MyClass { }`

   - c) `type class MyClass { }`

   - d) `class = MyClass { }`

   - **Answer: a)**

28. **Which method in a TypeScript class is called when an object is created?**

   - a) init()

   - b) construct()

   - c) create()

   - d) constructor()

   - **Answer: d)**

### Constructor method

29. **What is a constructor method used for in TypeScript?**
   - a) To define static properties
   - b) To initialize objects of a class
   - c) To create methods in a class
   - d) To declare private variables
   - **Answer: b)**

30. **How do you

 define a constructor in a TypeScript class?**
   - a) `constructor() { }`
   - b) `init() { }`
   - c) `construct() { }`
   - d) `function constructor() { }`
   - **Answer: a)**

### Inheritance of classes

31. **How do you inherit a class in TypeScript?**

   - a) `class Child extends Parent { }`
   - b) `class Child inherits Parent { }`
   - c) `class Child implements Parent { }`
   - d) `class Child inherits from Parent { }`
   - **Answer: a)**

32. **Which keyword is used to call the parent class constructor in TypeScript?**

   - a) this()
   - b) super()
   - c) parent()
   - d) base()
   - **Answer: b)**

### Type Assertion

33. **What is type assertion in TypeScript?**

   - a) Declaring a type for a variable
   - b) Converting a variable from one type to another

- c) Telling the compiler to treat a variable as a different type
- d) Checking the type of a variable
- **Answer: c)**

34. **How do you perform type assertion in TypeScript?**
- a) `let value = <string>variable;`
- b) `let value = variable as string;`
- c) Both a and b
- d) None of the above
- **Answer: c)**

### Abstract class

35. **What is an abstract class in TypeScript?**
- a) A class that cannot be instantiated
- b) A class that has no properties
- c) A class that has no methods
- d) A class that only contains static methods
- **Answer: a)**

36. **How do you define an abstract class in TypeScript?**

  - a) `abstract class MyClass { }`

  - b) `class abstract MyClass { }`

  - c) `abstract MyClass { }`

  - d) `class MyClass abstract { }`

  - **Answer: a)**

### Interface Declaration and Initialization with an object

37. **What is an interface in TypeScript?**

  - a) A blueprint for classes

  - b) A way to define multiple constructors

  - c) A way to declare a class without methods

  - d) A type assertion method

  - **Answer: a)**

38. **How do you declare an interface in TypeScript?**

- a) `interface MyInterface { }`

- b) `type interface MyInterface { }`

- c) `interface: MyInterface { }`

- d) `interface = MyInterface { }`

- **Answer: a)**

### When to Use Generic Functions

39. **What is a generic function in TypeScript?**

- a) A function that can operate on any data type

- b) A function that returns a generic type

- c) A function that can have multiple parameters

- d) A function that can return any value

- **Answer: a)**

40. **When should you use generic functions in TypeScript?**

- a) When you want to write reusable code that works with any data type

- b) When you need a function to return multiple values

- c) When you want to enforce type checking on function parameters

- d) When you need a function to operate on specific data types

- **Answer: a)**

### Additional Questions

41. **Which keyword is used to define a constant variable in TypeScript?**

- a) var

- b) let

- c) const

- d) constant

- **Answer: c)**

42. **What is the default value of an uninitialized variable in TypeScript?**

- a) null

- b) undefined

- c) 0

- d) ""

- **Answer: b)**

43. **Which of the following is a correct way to define a function in TypeScript?**

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

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

- c) `function add(a: number, b: number): string { return a + b; }`

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

- **Answer: a)**

44. **How do you define an optional parameter in a TypeScript function?**

- a) `function greet(name?: string) { }`

- b) `function greet(name: string?) { }`

- c) `function greet(?name: string) { }`

- d) `function greet(name: string|undefined) { }`

- **Answer: a)**

45. **Which operator is used to spread an array in TypeScript?**

- a) `...`
- b) `*`
- c) `&`
- d) `@`
- **Answer: a)**

46. **Which of the following is a TypeScript type guard?**
   - a) `typeof`
   - b) `instanceof`
   - c) Both a and b
   - d) None of the above
   - **Answer: c)**

47. **How do you declare a read-only property in a TypeScript class?**
   - a) `readonly`
   - b) `const`
   - c) `static`
   - d) `final`
   - **Answer: a)**

48. **What does the `protected` keyword do in TypeScript?**

   - a) It makes a class property accessible only within the class

   - b) It makes a class property accessible only within the class and its subclasses

   - c) It makes a class property accessible from anywhere

   - d) It makes a class property immutable

   - **Answer: b)**

49. **How do you specify a default value for a function parameter in TypeScript?**

   - a) `function greet(name: string = "Guest") { }`

   - b) `function greet(name = "Guest": string) { }`

   - c) `function greet(name: string) = "Guest" { }`

   - d) `function greet(name: string) { if (!name) name = "Guest"; }`

   - **Answer: a)**

50. **What is the correct way to define a union type in TypeScript?**

  - a) `let value: number|string;`

  - b) `let value: number|or|string;`

  - c) `let value: number or string;`

  - d) `let value: (number|string);`

  - **Answer: a)**

### Continue with more questions:

51. **How do you create a new instance of a class in TypeScript?**

  - a) `let obj = MyClass();`

  - b) `let obj = new MyClass();`

  - c) `let obj = create MyClass();`

  - d) `let obj = MyClass.new();`

  - **Answer: b)**

52. **What is the purpose of the `extends` keyword in TypeScript?**

  - a) To implement an interface

- b) To inherit from another class
- c) To declare a constant
- d) To define a new type
- **Answer: b)**

53. **How do you declare a method in a TypeScript interface?**
    - a) `method(): void;`
    - b) `function method(): void;`
    - c) `method = function(): void;`
    - d) `method: void;`
    - **Answer: a)**

54. **Which of the following keywords is used to define a variable that cannot be reassigned in TypeScript?**
    - a) const
    - b) let
    - c) var
    - d) static
    - **Answer: a)**

55. **How do you specify that a property in an interface is optional?**

  - a) `property?: type;`

  - b) `property: type?;`

  - c) `property?: type?;`

  - d) `optional property: type;`

  - **Answer: a)**

56. **Which TypeScript feature allows you to create a variable that can hold values of multiple types?**

  - a) Union types

  - b) Generics

  - c) Enums

  - d) Type assertions

  - **Answer: a)**

57. **What is the output of the following TypeScript code? `let x: number = 10; console.log(typeof x);`**

  - a) "number"

  - b) "string"

- c) "boolean"

- d) "object"

- **Answer: a)**

58. **What is the correct syntax for defining a function with a return type in TypeScript?**

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

- b) `function add(a: number, b: number) -> number { return a + b; }`

- c) `function add(a: number, b: number) => number

{ return a + b; }`

- d) `function add(a: number, b: number): number -> { return a + b; }`

- **Answer: a)**

59. **Which TypeScript feature provides a way to create more reusable and flexible functions?**

- a) Generics

- b) Interfaces

- c) Enums
- d) Abstract classes
- **Answer: a)**

60. **What is the default access modifier for properties and methods in a TypeScript class?**
    - a) public
    - b) private
    - c) protected
    - d) readonly
    - **Answer: a)**

61. **How do you define a static method in a TypeScript class?**
    - a) `static methodName() { }`
    - b) `methodName static() { }`
    - c) `function methodName() static { }`
    - d) `methodName() static { }`
    - **Answer: a)**

62. **How do you create an instance of an abstract class in TypeScript?**

   - a) `let obj = new AbstractClass();`

   - b) `let obj = create AbstractClass();`

   - c) Abstract classes cannot be instantiated directly

   - d) `let obj = AbstractClass.new();`

   - **Answer: c)**

63. **Which keyword is used to indicate that a class implements an interface in TypeScript?**

   - a) implements

   - b) extends

   - c) uses

   - d) supports

   - **Answer: a)**

64. **How do you define a property in an interface in TypeScript?**

   - a) `property: type;`

   - b) `property(type);`

   - c) `property -> type;`

   - d) `property = type;`

   - **Answer: a)**

65. **What does the `readonly` keyword do in TypeScript?**

   - a) It makes a property immutable after initialization

   - b) It makes a property private

   - c) It makes a property accessible only within the class

   - d) It makes a property write-only

   - **Answer: a)**

66. **Which of the following is a correct way to declare a tuple type in TypeScript?**

   - a) `let tuple: [string, number];`

   - b) `let tuple: {string, number};`

   - c) `let tuple: (string, number);`

   - d) `let tuple: Array<string, number>;`

   - **Answer: a)**

67. **How do you specify that a function parameter can be of more than one type in TypeScript?**

  - a) `parameter: type1 | type2`

  - b) `parameter: type1 & type2`

  - c) `parameter: type1 or type2`

  - d) `parameter: (type1, type2)`

  - **Answer: a)**

68. **What does the `void` keyword indicate when used as a return type for a function in TypeScript?**

  - a) The function does not return a value

  - b) The function returns a number

  - c) The function returns a string

  - d) The function returns null

  - **Answer: a)**

69. **Which of the following is the correct syntax for declaring a type alias in TypeScript?**

  - a) `type MyType = { name: string, age: number };`

  - b) `alias MyType = { name: string, age: number };`

  - c) `typedef MyType = { name: string, age: number };`

- d) `define MyType = { name: string, age: number };`
- **Answer: a)**

70. **How do you declare a method in a TypeScript class that does not return a value?**
- a) `methodName(): void { }`
- b) `methodName(): number { }`
- c) `methodName(): string { }`
- d) `methodName(): null { }`
- **Answer: a)**

71. **How do you define an enum with custom numeric values in TypeScript?**
- a) `enum Colors { Red = 1, Green = 2, Blue = 3 }`
- b) `enum Colors { Red: 1, Green: 2, Blue: 3 }`
- c) `enum Colors { Red(1), Green(2), Blue(3) }`
- d) `enum Colors = { Red: 1, Green: 2, Blue: 3 }`
- **Answer: a)**

72. **How do you use a generic function in TypeScript?**

- a) `function identity<T>(arg: T): T { return arg; }`
- b) `function identity<T>(arg: T) -> T { return arg; }`
- c) `function identity<T>(arg: T): T { return arg; }`
- d) `function identity(arg: T): T { return arg; }`
- **Answer: a)**

73. **Which of the following is the correct way to define a type assertion in TypeScript?**
    - a) `let value = <string>variable;`
    - b) `let value = variable as string;`
    - c) Both a and b
    - d) None of the above
    - **Answer: c)**

74. **How do you declare a class property as private in TypeScript?**
    - a) `private propertyName: type;`
    - b) `propertyName: private type;`
    - c) `propertyName private: type;`
    - d) `propertyName: type private;`

- **Answer: a)**

75. **What is the correct way to define an interface that extends another interface in TypeScript?**
    - a) `interface Child extends Parent { }`
    - b) `interface Child implements Parent { }`
    - c) `interface Child inherits Parent { }`
    - d) `interface Child uses Parent { }`
    - **Answer: a)**

76. **How do you define a function type in a TypeScript interface?**
    - a) `method: (param: type) => returnType;`
    - b) `method(param: type) -> returnType;`
    - c) `method: function(param: type): returnType;`
    - d) `method(param: type) => returnType;`
    - **Answer: a)**

77. **What is the purpose of the `super` keyword in TypeScript?**
    - a) To call the constructor of the base class

- b) To call a method from the base class
- c) Both a and b
- d) None of the above
- **Answer: c)**

78. **How do you declare an array of strings in TypeScript?**
    - a) `let arr: string[];`
    - b) `let arr: Array<string>;`
    - c) Both a and b
    - d) None of the above
    - **Answer: c)**

79. **What is the correct way to define a method in a TypeScript class?**
    - a) `methodName(): returnType { }`
    - b) `function methodName(): returnType { }`
    - c) `methodName(): returnType -> { }`
    - d) `methodName(): returnType = { }`
    - **Answer: a)**

80. **How do you define an interface with optional properties in TypeScript?**

   - a) \`interface MyInterface { property?: type; }\`

   - b) \`interface MyInterface { property: type?; }\`

   - c) \`interface MyInterface { property: type; optional; }\`

   - d) \`interface MyInterface { optional property: type; }\`

   - **Answer: a)**

81. **How do you declare a method in a TypeScript interface?**

   - a) \`method(): returnType;\`

   - b) \`method() -> returnType;\`

   - c) \`method(): returnType ->;\`

   - d) \`method() => returnType;\`

   - **Answer: a)**

82. **What is the correct way to define a readonly property in a TypeScript interface?**

   - a) \`readonly propertyName: type;\`

   - b) \`propertyName: readonly type;\`

- c) `propertyName: type readonly;`
- d) `readonly propertyName type;`
- **Answer: a)**

83. **How do you define a method in a TypeScript class that returns a value?**
    - a) `methodName(): returnType { }`
    - b) `methodName() -> returnType { }`
    - c) `methodName(): returnType -> { }`
    - d) `methodName(): returnType = { }`
    - **Answer: a)**

84. **Which of the following is the correct way to define an interface in TypeScript?**
    - a) `interface MyInterface { }`
    - b) `interface: MyInterface { }`
    - c) `type interface MyInterface { }`
    - d) `interface = MyInterface { }`

    - **Answer: a)**

85. **How do you define a function with an optional parameter in TypeScript?**

- a) `function greet(name?: string) { }`
- b) `function greet(name: string?) { }`
- c) `function greet(?name: string) { }`
- d) `function greet(name: string | undefined) { }`
- **Answer: a)**

86. **Which of the following is a correct way to define an enum in TypeScript?**

- a) `enum Colors { Red, Green, Blue }`
- b) `enum Colors = { Red, Green, Blue }`
- c) `enum Colors [ Red, Green, Blue ]`
- d) `enum Colors < Red, Green, Blue >`
- **Answer: a)**

87. **How do you create a new instance of a class in TypeScript?**

- a) `let obj = new MyClass();`
- b) `let obj = MyClass();`

    - c) `let obj = create MyClass();`

    - d) `let obj = MyClass.new();`

    - **Answer: a)**

88. **How do you specify a default value for a function parameter in TypeScript?**

    - a) `function greet(name: string = "Guest") { }`

    - b) `function greet(name = "Guest": string) { }`

    - c) `function greet(name: string) = "Guest" { }`

    - d) `function greet(name: string) { if (!name) name = "Guest"; }`

    - **Answer: a)**

89. **Which keyword is used to define a variable that cannot be reassigned in TypeScript?**

    - a) const

    - b) let

    - c) var

    - d) static

    - **Answer: a)**

90. **How do you define a generic function in TypeScript?**

- a) `function identity<T>(arg: T): T { return arg; }`

- b) `function identity<T>(arg: T) -> T { return arg; }`

- c) `function identity<T>(arg: T) => T { return arg; }`

- d) `function identity(arg: T): T { return arg; }`

- **Answer: a)**

91. **How do you define a method in a TypeScript interface?**

- a) `method(): returnType;`

- b) `method() -> returnType;`

- c) `method(): returnType ->;`

- d) `method() => returnType;`

- **Answer: a)**

92. **How do you define a method in a TypeScript class?**

- a) `methodName(): returnType { }`

- b) `function methodName(): returnType { }`

- c) `methodName(): returnType -> { }`
- d) `methodName(): returnType = { }`
- **Answer: a)**

93. **What is the correct way to define a readonly property in a TypeScript class?**
    - a) `readonly propertyName: type;`
    - b) `propertyName: readonly type;`
    - c) `propertyName: type readonly;`
    - d) `readonly propertyName type;`
    - **Answer: a)**

94. **How do you define an interface in TypeScript?**
    - a) `interface MyInterface { }`
    - b) `interface: MyInterface { }`
    - c) `type interface MyInterface { }`
    - d) `interface = MyInterface { }`
    - **Answer: a)**

95. **Which of the following is the correct way to define a union type in TypeScript?**

- a) `let value: number|string;`
- b) `let value: number|or|string;`
- c) `let value: number or string;`
- d) `let value: (number|string);`
- **Answer: a)**

96. **How do you define a tuple type in TypeScript?**

- a) `let tuple: [string, number];`
- b) `let tuple: {string, number};`
- c) `let tuple: (string, number);`
- d) `let tuple: Array<string, number>;`
- **Answer: a)**

97. **What is the correct way to define an abstract class in TypeScript?**

- a) `abstract class MyClass { }`
- b) `class abstract MyClass { }`
- c) `abstract MyClass { }`

- d) `class MyClass abstract { }`
- **Answer: a)**

98. **How do you create a new instance of an abstract class in TypeScript?**
    - a) Abstract classes cannot be instantiated directly
    - b) `let obj = new AbstractClass();`
    - c) `let obj = create AbstractClass();`
    - d) `let obj = AbstractClass.new();`
    - **Answer: a)**

99. **How do you declare a method in a TypeScript class that returns a value?**
    - a) `methodName(): returnType { }`
    - b) `methodName() -> returnType { }`
    - c) `methodName(): returnType -> { }`
    - d) `methodName(): returnType = { }`
    - **Answer: a)**

100. **What is the correct way to define a generic function in TypeScript?**

- a) `function identity<T>(arg: T): T { return arg; }`

- b) `function identity<T>(arg: T) -> T { return arg; }`

- c) `function identity<T>(arg: T) => T { return arg; }`

- d) `function identity(arg: T): T { return arg; }`

- **Answer: a)**