1. **Which of the following are true about TypeScript?**

   - a) TypeScript is a superset of JavaScript.

   - b) TypeScript can run directly in a browser.

   - c) TypeScript supports static typing.

   - d) TypeScript doesn't support ES6+ features.

   Answer: a, c

2. **Which of the following are valid TypeScript types?**

   - a) `string`

   - b) `number`

   - c) `any`

   - d) `data`

   Answer: a, b, c

3. **What can be used to create a type alias in TypeScript?**

   - a) `interface`

   - b) `type`

   - c) `enum`

   - d) `class`

   Answer: a, b

4. **Which of the following keywords can be used to declare variables in TypeScript?**

   - a) `let`

   - b) `var`

   - c) `const`

   - d) `define`

   Answer: a, b, c

5. **Which of the following are correct about interfaces in TypeScript?**

- a) Interfaces can extend other interfaces.

- b) Interfaces can implement classes.

- c) Interfaces can define optional properties.

- d) Interfaces can enforce the shape of objects.

Answer: a, c, d

6. **Which of the following are valid ways to define a function in TypeScript?**

- a) `function myFunc(): void {}`

- b) `const myFunc = (): void => {}`

- c) `let myFunc = function(): void {}`

- d) `function myFunc: void() {}`

Answer: a, b, c

7. **Which of the following can be used to enforce immutability in TypeScript?**

- a) `readonly` keyword

- b) `const` keyword

- c) `Object.freeze()`

- d) `private` keyword

Answer: a, b, c

8. **Which of the following are true about TypeScript generics?**

- a) Generics allow the creation of reusable components.

- b) Generics provide a way to define functions that work with multiple types.

- c) Generics enforce strict typing at runtime.

- d) Generics can be used with classes, interfaces, and functions.

Answer: a, b, d

9. **Which of the following statements are correct about type assertions in TypeScript?**

- a) Type assertions are used to tell the compiler to treat a value as a specific type.
- b) Type assertions change the runtime type of a variable.
- c) Type assertions can be performed using `as` syntax.
- d) Type assertions are the same as type casting in other languages.

Answer: a, c

10. **Which of the following are TypeScript access modifiers?**

- a) `private`
- b) `public`
- c) `protected`
- d) `sealed`

Answer: a, b, c

11. **Which of the following can be used to define a tuple in TypeScript?**

- a) `[string, number]`
- b) `Array<string, number>`
- c) `Tuple<string, number>`
- d) `(string, number)`

Answer: a

12. **Which of the following are correct about TypeScript enums?**

- a) Enums can have string or numeric values.
- b) Enums can be iterated over using `for...of`.
- c) Enums can have computed members.
- d) Enums are zero-based by default.

Answer: a, c, d

13. **Which of the following can be used to combine types in TypeScript?**

- a) `union types`
- b) `intersection types`
- c) `type guards`
- d) `interfaces`

Answer: a, b, d

14. **Which of the following are valid ways to define an array in TypeScript?**

- a) `let arr: number[] = [1, 2, 3];`
- b) `let arr: Array<number> = [1, 2, 3];`
- c) `let arr: [number] = [1, 2, 3];`
- d) `let arr = new Array<number>(1, 2, 3);`

Answer: a, b, d

15. **Which of the following are true about TypeScript decorators?**

- a) Decorators can modify the behavior of classes, methods, and properties.
- b) Decorators are applied at compile time.
- c) Decorators are an experimental feature in TypeScript.
- d) Decorators can only be used with classes.

Answer: a, c

16. **Which of the following are valid use cases for the `never` type in TypeScript?**

- a) A function that always throws an error.
- b) A variable that can never be assigned a value.
- c) A function that never returns.

  - d) A function that returns `null`.

  Answer: a, c

17. **Which of the following are correct about TypeScript modules?**
  - a) Modules in TypeScript allow code to be split into separate files.
  - b) Modules can export classes, functions, and variables.
  - c) TypeScript modules are the same as JavaScript ES modules.
  - d) Modules in TypeScript are only supported in Node.js.

  Answer: a, b, c

18. **Which of the following can be used to create a mapped type in TypeScript?**
  - a) `keyof`
  - b) `in`
  - c) `extends`
  - d) `type`

  Answer: a, b, d

19. **Which of the following are correct about optional chaining (`?.`) in TypeScript?**
  - a) It can be used to access deeply nested properties without throwing an error.
  - b) It works with method calls as well as property accesses.
  - c) It returns `undefined` if the object is `null` or `undefined`.
  - d) It can be used to assign default values.

  Answer: a, b, c

20. **Which of the following statements are true about TypeScript's `type` keyword?**
  - a) It is used to create new types.
  - b) It can be used to create union and intersection types.

- c) It can create types based on existing types.
- d) It is used to declare variables.

Answer: a, b, c

21. **Which of the following are valid ways to define a `readonly` property in TypeScript?**
   - a) Use the `readonly` keyword in the property declaration.
   - b) Use the `readonly` modifier in the constructor parameter.
   - c) Use the `const` keyword when declaring the property.
   - d) Use the `Object.freeze()` method.

   Answer: a, b

22. **Which of the following are true about TypeScript's type guards?**
   - a) Type guards allow narrowing down the type of a variable.
   - b) `typeof` and `instanceof` are commonly used for type guards.
   - c) Type guards are executed at runtime.
   - d) Type guards are only used with primitive types.

   Answer: a, b, c

23. **Which of the following are correct about the `unknown` type in TypeScript?**
   - a) The `unknown` type is a safer alternative to `any`.
   - b) Values of type `unknown` can be assigned to any type.
   - c) Before using a value of type `unknown`, you need to perform a type check.
   - d) The `unknown` type is the default type for variables in TypeScript.

   Answer: a, c

24. **Which of the following are valid use cases for TypeScript's `Partial` utility type?**
   - a) Making all properties of an interface optional.

- b) Making all properties of a type required.

- c) Creating a new type with some properties of another type.

- d) Creating a subset of an existing type.

Answer: a, d

25. **Which of the following are valid ways to define a custom type guard in TypeScript?**

- a) Use a function that returns a boolean.

- b) Use a function that returns `value is Type`.

- c) Use a function that returns `type`.

- d) Use a function that returns `asserts value is Type`.

Answer: b, d

26. **Which of the following are true about abstract classes in TypeScript?**

- a) Abstract classes cannot be instantiated.

- b) Abstract classes can have both abstract and non-abstract methods.

- c) Abstract methods must be implemented by subclasses.

- d) Abstract classes can implement interfaces.

Answer: a, b, c, d

27. **Which of the following are correct about the `readonly` modifier in TypeScript?**

- a) It can be applied to properties.

-

b) It can be applied to parameters.

- c) It can be applied to variables.

- d) It can be used with `const`.

Answer: a, b

28. **Which of the following are valid ways to declare an optional parameter in a TypeScript function?**

- a) `function myFunc(param?: string): void {}`
- b) `function myFunc(param: string | undefined): void {}`
- c) `function myFunc(param: string = ''): void {}`
- d) `function myFunc(param: string | null): void {}`

Answer: a, b, c

29. **Which of the following are valid for TypeScript namespaces?**

- a) Namespaces are used to organize code in TypeScript.
- b) Namespaces can contain classes, interfaces, and functions.
- c) Namespaces provide a way to encapsulate and export code.
- d) Namespaces are the same as modules.

Answer: a, b, c

30. **Which of the following can be used with `keyof` in TypeScript?**

- a) `keyof` can be used to get the keys of a type.
- b) `keyof` can be used to create mapped types.
- c) `keyof` can be used to create union types.
- d) `keyof` can be used to access values of an object.

Answer: a, b, c

31. **Which of the following are valid for the `Record` utility type in TypeScript?**

- a) It creates an object type with keys of type `K` and values of type `T`.
- b) It can be used to create a map-like structure.
- c) It can be used to define a type with dynamic keys.
- d) It can enforce a specific shape for an object.

Answer: a, b, c

32. **Which of the following are true about the `extends` keyword in TypeScript?**

 - a) It can be used to create subclasses.

 - b) It can be used to create generic constraints.

 - c) It can be used to implement interfaces.

 - d) It can be used to combine types.

 Answer: a, b

33. **Which of the following are correct about TypeScript's `this` keyword?**

 - a) `this` refers to the current object in a method.

 - b) `this` can be explicitly typed in functions.

 - c) `this` behaves the same as in JavaScript.

 - d) `this` can be used in arrow functions to refer to the enclosing context.

 Answer: a, b, c, d

34. **Which of the following are valid ways to create a tuple with optional elements in TypeScript?**

 - a) `[string?, number?]`

 - b) `[string?, number]`

 - c) `[string, number?]`

 - d) `[(string | undefined)?, number?]`

 Answer: b, c

35. **Which of the following are true about the `Pick` utility type in TypeScript?**

 - a) `Pick` creates a new type by selecting specific keys from an existing type.

 - b) `Pick` can be used with both interfaces and type aliases.

 - c) `Pick` can remove keys from a type.

- d) `Pick` can enforce required properties.

Answer: a, b

36. **Which of the following are valid ways to define a discriminated union in TypeScript?**
   - a) Use a union type with a common literal property.
   - b) Use an interface with multiple possible shapes.
   - c) Use a `switch` statement to check the discriminant.
   - d) Use a `type` keyword with multiple types.

   Answer: a, c

37. **Which of the following are correct about TypeScript's `Partial` utility type?**
   - a) It makes all properties of a type optional.
   - b) It can be used with both interfaces and type aliases.
   - c) It can add new properties to a type.
   - d) It can make all properties of a type required.

   Answer: a, b

38. **Which of the following are valid ways to define an index signature in TypeScript?**
   - a) `{ [key: string]: any }`
   - b) `{ [key: number]: string }`
   - c) `{ key: string[] }`
   - d) `{ [key: symbol]: any }`

   Answer: a, b, d

39. **Which of the following are true about TypeScript's `Exclude` utility type?**
   - a) `Exclude` removes types from a union type.
   - b) `Exclude` can be used to filter out specific types.

- c) `Exclude` can add new types to a union type.
- d) `Exclude` works with both primitive and complex types.

Answer: a, b, d

40. **Which of the following are valid ways to define a recursive type in TypeScript?**
- a) Use a type alias that references itself.
- b) Use an interface that references itself.
- c) Use a class that references itself.
- d) Use a union type that references itself.

Answer: a, b, c

41. **Which of the following are valid uses of the `infer` keyword in TypeScript?**
- a) It is used in conditional types to infer the type of a variable.
- b) It can be used to create new types based on existing types.
- c) It can be used to infer the return type of a function.
- d) It can be used to enforce a specific type.

Answer: a, b, c

42. **Which of the following are valid ways to create a custom type guard function in TypeScript?**
- a) Use a function that returns `value is Type`.
- b) Use a function that returns `boolean`.
- c) Use a function that returns `asserts value is Type`.
- d) Use a function that returns `type`.

Answer: a, c

43. **Which of the following are correct about the `Readonly` utility type in TypeScript?**
- a) It makes all properties of a type read-only.

- b) It can be used with both interfaces and type aliases.

- c) It can enforce immutability at compile-time.

- d) It can make only specific properties read-only.

Answer: a, b, c

44. **Which of the following are valid ways to define a conditional type in TypeScript?**

- a) `type MyType = T extends U ? X : Y;`

- b) `type MyType = T extends U ? X : T;`

- c) `type MyType = T & U ? X : Y;`

- d) `type MyType = T | U ? X : Y;`

Answer: a, b

45. **Which of the following are valid use cases for the `Omit` utility type in TypeScript?**

- a) To remove specific keys from a type.

- b) To create a new type without certain properties.

- c) To add new properties to a type.

- d) To enforce required properties in a type.

Answer: a, b

46. **Which of the following are correct about `template literal types` in TypeScript?**

- a) They allow creating complex string types based on union types.

- b) They can be used to create strongly typed string literals.

- c) They can interpolate types within a string type.

- d) They are the same as string templates in JavaScript.

Answer: a, b, c

47. **Which of the following are true about the `Parameters` utility type in TypeScript?**

- a) It extracts the parameter types of a function type.
- b) It returns a tuple of the parameter types.
- c) It can be used with both regular and arrow functions.
- d) It can infer the return type of a function.

Answer: a, b, c

48. **Which of the following are valid ways to define an `intersection type` in TypeScript?**
   - a) `type MyType = A & B;`
   - b) `type MyType = A | B;`
   - c) `type MyType = A & B & C;`
   - d) `type MyType = A | B | C;`

   Answer: a, c

49. **Which of the following are true about TypeScript's `Mapped Types`?**
   - a) They allow creating new types by transforming properties of an existing type.
   - b) They can make all properties of a type optional.
   - c) They can enforce strict immutability.
   - d) They can make all properties of a type required.

   Answer: a, b, d

50. **Which of the following are valid ways to use the `NonNullable` utility type in TypeScript?**
   - a) To remove `null` and `undefined` from a type.
   - b) To create a type that excludes `null` and `undefined`.
   - c) To enforce non-nullable properties in a type

.

   - d) To ensure a value is never `null` or `undefined`.

Answer: a, b