

TypeScript Fundamentals

1. What is TypeScript?
2. Why was TypeScript created?
3. TypeScript vs JavaScript – key differences?
4. Is TypeScript statically or dynamically typed?
5. How does TypeScript improve code quality?
6. What are the advantages of TypeScript?
7. What are the limitations of TypeScript?
8. How does TypeScript compile to JavaScript?
9. What is the TypeScript compiler (tsc)?
10. What is tsconfig.json?
11. What happens if TypeScript code has type errors?
12. Can browsers run TypeScript directly?
13. What is type inference?
14. What is structural typing?
15. Nominal vs structural typing?
16. What is transpilation?
17. What are .d.ts files?
18. What is --strict mode?
19. How do you migrate a JS project to TypeScript?
20. When should you NOT use TypeScript?

Basic Types

21. What are basic data types in TypeScript?
22. Difference between any and unknown?
23. What is never?
24. Difference between void and never?
25. What is null and undefined handling?
26. What is union type?
27. What is intersection type?
28. What are literal types?

- 29. What is type alias?
 - 30. Type alias vs interface?
 - 31. What is enum?
 - 32. Numeric enum vs string enum?
 - 33. What is tuple?
 - 34. How are arrays typed in TypeScript?
 - 35. What is readonly property?
-

Functions in TypeScript

- 36. How do you type a function?
 - 37. Optional parameters vs default parameters?
 - 38. What are rest parameters?
 - 39. How do you type arrow functions?
 - 40. Function overloading in TypeScript?
 - 41. How does TypeScript handle callbacks?
 - 42. What is this parameter in functions?
 - 43. How do you type async functions?
 - 44. What is contextual typing?
 - 45. How do you type higher-order functions?
 - 46. What are function signatures?
 - 47. How do you handle nullable parameters?
 - 48. How do you enforce return types?
 - 49. What is a type predicate?
 - 50. What is assertion function?
-

Interfaces & Object Types

- 51. What is an interface?
- 52. Interface vs type – when to use what?
- 53. How do interfaces support inheritance?
- 54. What is optional property?
- 55. What is readonly interface property?

- 56. Can interfaces describe functions?
 - 57. Can interfaces describe classes?
 - 58. What is excess property check?
 - 59. How does TypeScript handle duck typing?
 - 60. What is index signature?
 - 61. What are hybrid interfaces?
 - 62. What is declaration merging?
 - 63. How do you extend multiple interfaces?
 - 64. How do you enforce strict object shapes?
 - 65. How do interfaces work at runtime?
-

Classes & OOP

- 66. How do you define classes in TypeScript?
 - 67. Access modifiers: public, private, protected?
 - 68. What is readonly in classes?
 - 69. How does inheritance work?
 - 70. Abstract classes vs interfaces?
 - 71. What is constructor parameter property?
 - 72. What is method overriding?
 - 73. What is static property/method?
 - 74. What is polymorphism in TypeScript?
 - 75. What is encapsulation?
 - 76. What is implements keyword?
 - 77. How does TypeScript handle multiple inheritance?
 - 78. What is super keyword?
 - 79. How do you enforce immutability in classes?
 - 80. Can TypeScript classes exist at runtime?
-

Generics & Advanced Types

- 81. What are generics?
- 82. Why are generics needed?

83. Generic functions example?
 84. Generic interfaces?
 85. Generic classes?
 86. What are generic constraints?
 87. What is keyof?
 88. What is typeof in TypeScript?
 89. What is indexed access type?
 90. What is mapped type?
 91. What are utility types?
 92. Explain Partial, Required, Readonly?
 93. Explain Pick and Omit?
 94. Explain Record<K,T>?
 95. What are conditional types?
 96. What is infer keyword?
 97. What are template literal types?
 98. What is recursive type?
 99. Difference between union & generic?
 100. How do generics work at runtime?
-

Modules, Tooling & Configuration

101. What are modules in TypeScript?
102. ES modules vs CommonJS?
103. What is export default vs named export?
104. How does module resolution work?
105. What is path mapping?
106. What is baseUrl?
107. What is incremental compilation?
108. What is source map?
109. Babel vs TypeScript compiler?
110. How do you configure TypeScript for Node.js?

Asynchronous & Error Handling

- 111. How does TypeScript handle Promises?
- 112. How do you type `async/await`?
- 113. How do you type `Promise.all()`?
- 114. How do you handle errors in `async` code?
- 115. What is unknown in `catch` block?

Real-world & Scenario-Based

- 116. How do you enforce API contracts using TypeScript?
- 117. How do you share types between frontend and backend?
- 118. How do you type third-party JS libraries?
- 119. How do you avoid any in large projects?
- 120. How do you design scalable TypeScript applications?