JavaScript and TypeScript Interview Questions and Answers

JavaScript Interview Questions and Answers

1. **What is the difference between var, let, and const?**
   var: function-scoped, hoisted, can be re-declared.
   let: block-scoped, not hoisted (in TDZ), can be reassigned.
   const: block-scoped, cannot be reassigned.

2. **What is hoisting?**
   Hoisting means variable and function declarations are moved to the top of their scope before code execution.
   Example:
   console.log(a); // undefined
   var a = 5;

3. **What is a closure?**
   A closure is when a function remembers variables from its outer scope even after the outer function has finished executing.
   function outer() {
   let count = 0;
   return function inner() {
   count++;
   return count;
   }
   }
   const counter = outer();
   console.log(counter()); // 1

4. **Difference between == and ===?**
   == → checks value only (performs type coercion).
   === → checks value and type.

5. **Difference between undefined and null?**
   undefined → variable declared but not assigned.
   null → explicitly assigned “no value”.

6. **What are arrow functions?**
   Shorter syntax for functions; they don’t have their own this context.
   const sum = (a,b) => a+b;

7. **What is the “this” keyword?**
   Refers to the object that calls the function. Its value depends on how a function is invoked.

8. **What is a callback function?**
   A function passed as an argument to another function, executed later.
   function greet(name, callback){
   console.log(’Hi ’ + name);
   callback();
   }

9. **What is a Promise?**
   A Promise represents the eventual completion or failure of an asynchronous operation.
   States pending, fulfilled, rejected.

10. **What is async/await?**
    Syntactic sugar over Promises to write asynchronous code synchronously.
    async function getData(){
    const res = await fetch(’url’);
    return res.json();
    }

11. **What are higher-order functions?**
    Functions that take other functions as arguments or return them.
    Example: map, filter, reduce.

12. **What is event bubbling and capturing?**
    Bubbling: event goes from child parent.
    Capturing: event goes from parent child.

13. **What is the event loop?**
    It continuously checks the call stack and message queue to manage asynchronous operations.

14. **What are microtasks and macrotasks?**
    Microtasks: Promises, MutationObserver.
    Macrotasks: setTimeout, setInterval, setImmediate.

15. **What is debouncing?**
    Delays execution of a function until after a specified time has elapsed since the last call.
    Used in search boxes, resize events, etc.

16. **What is throttling?**
    Limits function execution to once every specified period.

17. **What is the difference between shallow and deep copy?**
    Shallow copy: copies references.
    Deep copy: copies all nested objects.

18. **How do you deep copy an object?**
    let deepCopy = JSON.parse(JSON.stringify(obj));

19. **What are template literals?**
    String literals allowing embedded expressions using backticks.
    ‘Hello $name‘

20. **What is destructuring?**
    Extracting values from arrays/objects into variables.
    const {name, age} = person;

21. **What are rest and spread operators?**
    Rest: combines arguments array (function(...args){})
    Spread: expands array/object elements.

22. **What is prototypal inheritance?**
    Objects inherit properties from other objects using prototype.

23. **How to prevent object modification?**
    Use Object.freeze(obj).

24. **What are pure functions?**
    Functions with no side effects and same output for same inputs.

25. **What is immutability?**
    Once created, object state cannot change. Instead, create a new object.

26. **Difference between shallow and deep comparison?**
    Shallow compares top-level properties only; deep checks nested objects.

27. **What is call(), apply(), bind()?**
    call() calls function with given this and args.
    apply() same but takes arguments as array.
    bind() returns new function with bound this.

28. **What are JavaScript data types?**
    Primitive string, number, boolean, null, undefined, symbol, bigint.
    Non-primitive object.

29. **What is the difference between map() and forEach()?**
    map() returns a new array.
    forEach() just iterates (no return).

30. **What is the difference between slice() and splice()?**
    slice() returns new array (non-mutating).
    splice() modifies original array.

31. **What is NaN?**
    Represents “Not a Number”, result of invalid arithmetic operation.

32. **How to check if a variable is an array?**
    Array.isArray(arr)

33. **What is an Immediately Invoked Function Expression (IIFE)?**
    A function executed immediately after creation.
    (function(){
    console.log(”IIFE”);
    })();

34. **What is the use of typeof and instanceof?**
typeof returns type of primitive value.
instanceof checks if object belongs to a class.

35. **What is the difference between localStorage, sessionStorage, and cookies?**
localStorage persists indefinitely.
sessionStorage cleared on tab close.
cookies sent with HTTP requests.

36. **What are generators?**
Functions that can pause and resume using yield.
function* gen(){
yield 1;
yield 2;
}

37. **What is the difference between setTimeout and setInterval?**
setTimeout runs once after delay.
setInterval runs repeatedly.

38. **What is event delegation?**
Using a single event listener on a parent to manage all child elements.

39. **What is JSON?**
Lightweight data format for data exchange.
JSON.parse() and JSON.stringify() are used for conversion.

40. **What is garbage collection?**
Automatic memory cleanup for unreachable objects.

41. **What is use strict?**
Enables strict mode, preventing silent errors and unsafe operations.

42. **What are symbols?**
Unique and immutable primitive values, often used as object keys.

43. **What is currying?**
Breaking a function with multiple arguments into a sequence of single-argument functions.

44. **What are default parameters?**
Allow assigning default values to parameters in functions.

45. **What is the difference between Object.seal() and Object.freeze()?**
seal() prevents adding/removing properties.
freeze() prevents adding/removing/modifying properties.

46. **What are modules in JavaScript?**
Reusable pieces of code that can be imported/exported using import and export.

47. **What are ES6 features?**
Let/Const, Arrow Functions, Classes, Modules, Template Literals, Destruc-

turing, Promises, etc.

48. **How does async/await improve readability?**
    It makes asynchronous code look synchronous and easier to maintain.
49. **What is memoization?**
    Caching function results to improve performance.
50. **What is event propagation?**
    Describes event order: capturing target bubbling.
51. **What is object destructuring with alias?**
    const {name: userName} = person;
52. **What is Object.assign() used for?**
    To copy properties from one or more objects into another.
53. **What is the difference between shallow merge and deep merge?**
    Shallow merges only top-level properties; Deep merges nested objects too.
54. **What is the output of [] + []?**
    Empty string "" — because both arrays are converted to strings.
55. **What is the difference between function and class?**
    Class is a syntactic sugar over prototype-based inheritance.
56. **What is promise chaining?**
    Calling .then() successively to handle sequential asynchronous operations.
57. **What is an async iterator?**
    An iterator that works with asynchronous data sources using for await...of.
58. **What are tagged template literals?**
    Template literals processed by a function.
59. **What is shadowing in JS?**
    When a variable in local scope hides one in outer scope.
60. **How to handle errors in async/await?**
    Use try...catch blocks.

TypeScript Interview Questions and Answers

61. **What is TypeScript?**
    TypeScript is a superset of JavaScript that adds static typing and compile-time checks.
62. **What are the benefits of TypeScript?**
    Type safety
    Better code completion
    Early error detection
    Enhanced readability

63. **What is the difference between TypeScript and JavaScript?**
TypeScript is compiled (transpiled) to JavaScript and supports static types.

64. **How do you declare a variable with a type?**
let age: number = 25;

65. **What are basic data types in TypeScript?**
string, number, boolean, any, unknown, void, null, undefined, never.

66. **What is an interface?**
Defines the structure (shape) of an object.
interface User { name: string; age: number; }

67. **What is the difference between type and interface?**
Interface used for object shapes, extendable.
Type can define unions, primitives, intersections.

68. **What is type inference?**
TypeScript automatically infers variable types when not explicitly declared.

69. **What is union type?**
Variable can hold more than one type.
let value: string | number;

70. **What is intersection type?**
Combines multiple types.
type Person = Name & Contact;

71. **What are generics?**
Allow defining reusable components with variable types.
function identity<T>(arg: T): T { return arg; }

72. **What is type assertion?**
Tell the compiler to treat a value as a specific type.
let someValue: any = ”hello”;
let strLength = (someValue as string).length;

73. **What is readonly modifier?**
Makes properties immutable.
interface Person { readonly id: number; }

74. **What is enum in TypeScript?**
Defines a set of named constants.
enum Direction { Up, Down, Left, Right }

75. **What are tuples?**
Fixed-length arrays with specific types.
let user: [string, number] = [’John’, 25];

76. **What is unknown type?**
A safer alternative to any — must be type-checked before use.

77. **What is the difference between any and unknown?**
any: disables all checks.
unknown: requires checks before usage.

78. **What is never type?**
Represents values that never occur (e.g., errors, infinite loops).

79. **What are access modifiers?**
public, private, protected — control visibility of class members.

80. **What is abstract class?**
A class that cannot be instantiated directly and may contain abstract methods.

81. **What is the difference between interface and abstract class?**
Interface only method signatures.
Abstract can include implementation.

82. **What are namespaces?**
Logical grouping of code within a single scope (rarely used now, replaced by modules).

83. **What are modules?**
Files with exported/imported code blocks.

84. **What is implements keyword?**
Ensures a class follows a specific interface structure.

85. **What are decorators?**
Functions that add metadata or modify classes, methods, or properties. (Used heavily in Angular.)

86. **What is Partial<T>?**
A utility type that makes all properties optional.

87. **What is Pick<T, K>?**
Creates a type by picking a subset of properties.

88. **What is Omit<T, K>?**
Creates a type by excluding properties.

89. **What are mapped types?**
Types that transform properties of another type.
type ReadOnly<T> = { readonly [P in keyof T]: T[P] };

90. **What is Record<K, T>?**
Constructs a type with keys K and values T.
type UserRoles = Record<'admin' | 'user', string>;

91. **What is keyof?**
Gets the keys of a type.
type Keys = keyof Person; // 'name' | 'age'

92. **What is typeof in TypeScript?**
Gets the type of a variable at compile time.

93. **What are conditional types?**
Types based on conditions.
type IsString<T> = T extends string ? true : false;

94. **What is type narrowing?**
TypeScript automatically narrows down union types using checks (typeof, instanceof).

95. **What are utility types?**
Built-in helpers like Partial, Required, Pick, Omit, Readonly.

96. **What is discriminated union?**
Union types with a common literal property to identify variant types.

97. **What are index signatures?**
Allow defining objects with flexible property names.
interface Dictionary { [key: string]: string; }

98. **What is the purpose of declare keyword?**
Used to define variables or modules that exist elsewhere (e.g., global scripts).

99. **What is as const?**
Makes object properties immutable and literal-typed.

100. **What is the difference between compile-time and runtime errors?**
Compile-time detected by TypeScript compiler.
Runtime detected during program execution (JS level).