96 Kotlin Interview Questions to Hire Top Developers

Questions

1. What is Kotlin, in simple words?

2. Can you name a few things that make Kotlin different from Java?

3. What are variables in Kotlin, and how do you declare them?

4. Explain what a function is and how you write one in Kotlin.

5. What are data classes in Kotlin and why are they useful?

6. What is null safety in Kotlin and why is it important?

7. Can you describe what a 'loop' is, and how you'd use one in Kotlin?

8. What are classes and objects? What's the relationship between them?

9. What are the different ways to compare two things in Kotlin?

10. Tell me about the 'when' expression in Kotlin. What's it for?

11. What are some basic types of data (like numbers or text) that Kotlin uses?

12. Explain what an array is and how to create one in Kotlin.

13. If you have a list of names, how can you pick out just the names that start with the letter 'A'?

14. What is the use of companion object in Kotlin?

15. What is difference between 'val' and 'const val' in Kotlin?

16. How does Kotlin handle errors? What are 'try' and 'catch' for?

17. Can you explain what extension functions are and why you might want to use them?

18. What is the use of scope functions in Kotlin?

19. What is the difference between 'apply' and 'also' scope functions?

20. What is the difference between 'let' and 'run' scope functions?

21. Have you used coroutines in Kotlin? If so, what was your experience?

22. In Kotlin, is it possible to write code that can work with different types of data? How?

23. How can you make sure a variable is never null in Kotlin?

24. What are interfaces in Kotlin, and how do they compare to classes?

25. How do you create a simple program that prints 'Hello, world!' in Kotlin?

26. What's the difference between `val` and `var` in Kotlin? Imagine `val` is like having a toy that you can't change, but `var` is like a toy you can swap for another.

27. Can you explain what a function is in Kotlin? Think of it like a recipe that tells the computer how to do something.

28. What is a string in Kotlin? Think of it as a bunch of letters, numbers, or symbols all stuck together to make words.

29. What are some basic data types in Kotlin (like Int, Boolean, String)? Imagine you have different boxes to hold different things — numbers, true/false values, and words.

30. What does `null` mean in Kotlin? It's like an empty box. There's nothing inside!

31. How do you print something to the screen in Kotlin? It's like shouting something so everyone can hear!

32. What is an `if` statement? It's like making a choice — 'If it's raining, take an umbrella'.

33. What is a `when` statement in Kotlin? It's like choosing between many options, like picking your favorite color from a box of crayons.

34. What is a `for` loop? Imagine you have a line of toys and you want to play with each one, one after the other.

35. What is a `while` loop? It's like doing something again and again as long as something is true, like singing a song until bedtime.

36. What is a list in Kotlin? It's like a shopping list where you can add and remove items.

37. How do you add an element to a list? Think of it as putting another toy into your toy box.

38. How do you get the size of a list? Like counting how many toys are in your toy box.

39. What is a function parameter? Think of it as an ingredient you need to give to a recipe to make it work.

40. What does it mean for a function to `return` a value? Imagine the recipe gives you a cake at the end.

41. What is a class in Kotlin? Think of it like a blueprint for building something, like a toy car.

42. What is an object in Kotlin? It's like the actual toy car built from the blueprint (class).

43. What is a constructor in Kotlin? It's like the instructions on how to assemble the toy car when you first get it.

44. What are properties of a class? Think of them as the features of the toy car, like its color or how many wheels it has.

45. How do you create an instance of a class (an object)? It's like building the toy car from the blueprint.

46. What is inheritance in Kotlin? It's like saying one toy car is a special kind of another toy car, like a race car being a type of car.

47. What is a package in Kotlin? Imagine it as a folder on your computer to organize your files, or a box to store similar toys.

48. What is the difference between `==` and `===` in Kotlin? Think of `==` as checking if two toys look the same and `===` as checking if they are the exact same toy.

49. What are extension functions in Kotlin? It's like teaching a toy to do a new trick that it didn't know before.

50. Explain the concept of immutability. Imagine you have a Lego castle. An immutable castle is one you can admire but not change. A mutable one? Go wild, rearrange those bricks!

51. Explain the difference between `also` and `apply` scope functions in Kotlin. When would you choose one over the other?

52. Describe how you would handle null safety in Kotlin using `let`, `run`, `with`, and the Elvis operator. Provide examples.

53. What are extension functions in Kotlin, and how can they be used to add functionality to existing classes without inheritance?

54. Explain the purpose of sealed classes in Kotlin and how they differ from enums. Provide use cases.

55. How does Kotlin's coroutines simplify asynchronous programming compared to traditional threading models? Explain with examples.

56. What is the difference between `const` and `val` in Kotlin? When should each be used?

57. Describe Kotlin's data classes. What benefits do they provide, and what methods are automatically generated?

58. Explain how you would use Kotlin's delegation feature. Provide an example of delegated properties.

59. What is the purpose of inline functions in Kotlin, and how can they improve performance? What are their limitations?

60. Describe Kotlin's higher-order functions. How can they be used with lambda expressions to create more flexible and reusable code?

61. Explain the concept of reified type parameters in Kotlin. How do they allow you to access type information at runtime?

62. How does Kotlin support operator overloading? Provide an example of overloading an operator for a custom class.

63. What are Kotlin's collections? Explain the difference between mutable and immutable collections and when to use each.

64. Describe how you would handle exceptions in Kotlin. What are the key differences between Kotlin's exception handling and Java's?

65. Explain the use of Kotlin's `use` function for resource management. How does it ensure resources are properly closed?

66. What are Kotlin's sequences, and how do they differ from collections? When would you use a sequence instead of a collection?

67. Describe how to create and use annotations in Kotlin. Provide an example of a custom annotation.

68. Explain the difference between `==` and `===` in Kotlin. When should each be used for comparison?

69. How does Kotlin support interoperability with Java? Explain how you can call Java code from Kotlin and vice versa.

70. What is the purpose of the `lateinit` modifier in Kotlin? When and why would you use it?

71. Describe how you would implement a singleton pattern in Kotlin. What are the different approaches you can use?

72. Explain how to use destructuring declarations in Kotlin. Provide examples of destructuring data classes and collections.

73. What are inline classes in Kotlin, and how do they differ from type aliases? When would you use inline classes?

74. How do you handle concurrency using Kotlin coroutines? Can you give an example of launching multiple coroutines?

75. How does Kotlin's `inline` keyword impact performance, and when should it be used judiciously?

76. Explain the differences between `Sequence` and `Iterable` in Kotlin, focusing on their use cases and performance characteristics.

77. Discuss the advantages and disadvantages of using Kotlin's data class compared to a regular class, especially in terms of memory usage and performance.

78. Describe how you would implement a custom delegation pattern in Kotlin, providing a practical example.

79. Explain Kotlin's coroutines, including how they work under the hood and how they differ from threads.

80. How would you handle null safety in a complex Kotlin project, including strategies for avoiding NullPointerExceptions?

81. Describe the use cases for Kotlin's sealed class and sealed interface constructs.

82. Explain how Kotlin's reflection capabilities can be used, and what are the potential drawbacks in terms of performance and security?

83. Discuss the differences between `lateinit` and by `lazy` for property initialization in Kotlin.

84. How does Kotlin support functional programming, and what are the benefits of using functional paradigms in Kotlin development?

85. Explain Kotlin's support for extension functions and properties, and when their use might be considered bad practice.

86. Describe the usage of Kotlin's `use` function and its importance in resource management.

87. How can you achieve immutability in Kotlin, and why is it important for concurrent programming?

88. Explain the concept of reified type parameters in Kotlin and when they are useful.

89. Discuss how you would approach testing Kotlin coroutines effectively.

90. Describe how you might implement a custom DSL (Domain Specific Language) in Kotlin.

91. Explain how you would use Kotlin's annotations for code generation or compile-time processing.

92. How does Kotlin interoperate with Java, and what are some best practices for mixed Kotlin/Java projects?

93. Describe the various ways to handle concurrency in Kotlin, focusing on coroutines and actors.

94. Explain how Kotlin's contract feature can be used to improve code safety and readability.

95. Discuss the use of `const val` vs `val` for defining constants in Kotlin, considering compile-time and runtime behavior.