

[By OnlineInterviewQuestions.com](http://OnlineInterviewQuestions.com)

Scala Interview Questions

What is Scala?

Scala is a high-level programming language that combines object-oriented and functional programming in one concise. finally, Practice here the Top 30+ **Scala Interview Questions**, which are mostly asked during scala job interviews. these interview questions are very popular and written by industry experts.

Q1. What do you mean by a Scala map?

The collection of key-value pairs where the key can retrieve the values present in a map is known as a scala map. The keys, unlike the values in a scala map are unique. There are two types of maps present in scala:

- Mutable
- Immutable

By default, immutable map is supported by scala. To use the mutable map, the programmer needs to import the “scala.collection.mutable”. It is an explicit class. “mutable.map” is the syntax for using the mutable and immutable class in the same program. To access the immutable class, we just have to use the name of the map and it can be accessed.

Q2. List the advantages of using scala over other functional programming languages.

Some of the advantages due to which scala is preferred over other functional programming languages are listed below:

- As the name suggests, it is a scalable programming language. It is highly scalable. Its high maintainability, productivity, and testability features make it more usable which turns out to be an advantage and hence it is preferred more.
- Scala consists of singleton and companion objects. These objects in scala, unlike the JVM languages provide a clearer solution to every problem.
- The need of ternary operator gets eliminated in scala as “if blocks”, “for-yield loops”, and “code” in braces return a value.

Q3. Tell the advantages of companion objects when used in Scala?

Instead of having static methods or variables, scala has singleton or companion objects. These objects are then compiled to classes which have static methods. Some advantages of these companion objects are listed below:

- These are beneficial as they can be used for encapsulating things. They also act as a bridge and hence functional and object-oriented programs can be written easily.
- These companion objects also help in keeping the scala programming code more concise because the static keyword need not be added to each and every attribute.
- A clear separation between the static and non-static methods is maintained with the help of these companion objects.

Q4. Which Scala library is used for the functional programming?

pure functional data structure is present in the scala library. It also complements the standard Scala library. It also has a pre-defined set of foundational type classes' like- Monad, Functor, etc. The standard library of Scala is automatically set as a dependency by default. The Scala library is used for compiling scala sources and for that it needs to be on the classpath.

Q5. Differentiate Nil, Null, None, and Nothing in scala.

The difference between these four attributes in scala is given below:

- Nil- it is used to initialize an empty list since it is an object which extends list.
- Null- null in scala is used to provide compatibility with the java null keyword or to provide a type for the null keyword. It also represents the absence of type information for complex types.
- None- the “none” pattern in scala is used to remove null values from the scala code.
- Nothing- it is used for providing the return type for the operations that can affect the normal flow of program.

Q6. What do you mean by a case class in scala?

These classes in scala are the classes which are declared by using a special modifier case. The constructor parameters are exported using these case classes which hence provide a recursive decomposition mechanism through pattern matching. We treat the constructor parameters of the case classes as public values and so they can be accessed directly. The companion objects and its related methods can also be generated automatically for the case classes. All the methods in the class and the companion objects are generated based on the parameter list. It helps in automatically generating the methods from the parameter list. By default, case objects and classes are serializable.

Q7. What do you understand by apply and unapply methods in scala?

These methods in scala are used for mapping and unmapping the data between the form and model data.

- Apply method- this method is used to assemble an object from its components. For eg.- if an employee object needs to be created, then the two components namely- first name and last name should be used and should be composed using the apply method.
- Unapply method- when we want to decompose the objects from its components, then we make use of this

unapply method. Reverse process is followed while making use of the unapply method. So, the employee object can be decomposed into two components namely- first name and last name.

Q8. Differentiate between Array and List in Scala.

The differences between array and list in Scala are listed below-

- An array is a sequential mutable data structure. Whereas, a list is an immutable recursive data structure.
- In scala, an array is an invariant. Whereas, a list is a covariant data structure.
- The size of an array is fixed and cannot be changed easily. Whereas, the size of a list may increase or decrease based on the operations performed by it.

Q9. What do you understand by an implicit parameter in Scala ?

Implicit parameters in Scala are used when we want to invoke a function without passing all the parameters. The default values of all the parameters or the parameters which we want to use are set as implicit. When implicit parameters are not used the local value of that parameter is used. An implicit keyword needs to be used if you want to make value, function parameter, or a variable as an implicit parameter. Once the value becomes implicit, we need not pass all the parameters to invoke a particular function.

Q10. What do you understand by tail recursion in Scala ?

The programmers face various situations where they need to write codes which are recursive in nature. But when we use the recursive functions, we may face a problem in which the function may eat up all the stack space. So, to overcome this problem, Scala provides this particular feature of tail recursion which optimizes the recursive function. After optimization, these functions do not create a new stack space, instead uses the current function stack space. “@annotation.tailrec” annotation has to be used before defining the function and for the last statement, recursive call can be used. After all this, your function will be compiled successfully or else it will give an error.

Q11. What is PreDef in Scala?

Predef in Scala is used for providing type aliases for the commonly know scala types such as collection types Map, Set, and the List constructors.

Q12. Explain what is Unit in Scala?

Unit is a type in Scala that is used as a return statement for a function when no value is to be returned. It is a subtype of scala.AnyVal.

Q13. What is "Type Inference" in Scala?

In Scala, the **Type Inference** makes it optional to specify the type of variable provided that type mismatch is handled.

Q14. What is Monad in Scala?

A **Monad** is an object that wraps another object in Scala that performs successive calculations such that the output of a calculation at any step is the input to other calculations, which run as a parent to the current step.

Q15. List the default imports are available in Scala Language?

The default imports in Scala language are:

lang - contains String, Object, Exception

imported - contains Int, Nothing, and Function

Q16. Explain BitSet in Scala?

Bitset is a common base class for mutable and immutable bitsets. Basically, the bitsets are sets of non-negative integers that are represented as variable-size arrays of bits packed into 64-bit words.

Q17. Explain trait in Scala and its uses?

In scala, **Trait** is a collection of abstract and non-abstract methods such that it can have all abstract methods or a mixture of the abstract and non-abstract methods. Traits are generally used to define the object types by specifying the signature of the supported methods.

Q18. What is a higher-order function in Scala?

In scala, **Higher-order** functions take other functions as parameters or return a function as a result. As these functions are first-class values in Scala, for example, the map function on arrays.

Q19. Explain function currying in Scala?

Currying is a process of transforming a function in Scala. This function takes multiple arguments into a function that takes a single argument, therefore it is applied widely in multiple functional languages. It can be represented as : **def function name(argument1, argument2) = operation.**

Q20. List Types of identifiers available in Scala?

There are four types of identifiers available in Scala:

- Alphanumeric Identifiers.
- Operator Identifiers.
- Mixed Identifiers.
- Literal Identifiers.

Q21. What is Scala option?

Scala Option is a container for zero or one element of a given type such that it can be either Some or None object, which represents a missing value. Therefore it is referred to as a carrier of single or no element for a stated type.

Q22. Please explain closure is Scala?

A **closure** is a function, whose return value depends on the value of one or more variables declared outside this function such that the return value of this function is dependent on these variables.

Q23. What is auxiliary constructor in Scala?

In Scala, the **auxiliary constructor** is used for constructor overloading and therefore defined as a method in the class using this keyword or name.

Q24. How does yield work in Scala?

In scala, the **yield keyword** is used to return a result after the completion of loop iterations. The for loop used buffer internally to store the iterated results and after finishing all iterations it yields the ultimate result from the particular buffer.

Q25. What do you mean by implicit parameters?

In scala, an **Implicit parameter** is the parameter that is passed to a function with an implicit keyword such that the values will be taken from the context in which they are called. Therefore, when the implicit keyword is used in the parameter scope of the function then all the parameters are marked as implicit.

Q26. [Explain the difference between a trait and an abstract class?](#)

Traits are stackable and are dynamically bound as well as they are interoperable with java if they don't have any implementation. Whereas the **Abstract classes** are not stackable and are statically bound and they are interoperable with java without any restriction.

Q27. [What is a applicative?](#)

Basically, **applicative** functor forms a type class that can be implemented in Scala such that it allows applying a wrapped function to a wrapped value.

Q28. [List different types of literals available in Scala?](#)

The different types of literals available in Scala are Character literals, **String literals**, **Multi-Line String literals**, **Boolean literals**, **Integer literals**, and **Floating-point literals**.

Q29. [What is a companion object?](#)

A **companion object** is an object that is declared in the same file as a class and has the same name as the class such that its class can access each other's, private members. It uses regular virtual methods which can be overridden through inheritance.

Q30. [What is the difference between a Java future and a Scala future?](#)

A **Java Future** works in a synchronous blocking way such that it is a bit clumsy, not elegant, and not concise.

Scala Future works in an asynchronous non-blocking way such that it supports very elegant and concise ways of writing concurrency code and support Concurrency and true Parallelism.

Q31. [What is Akka in Scala?](#)

Akka in Scala is a toolkit and runtime for building highly concurrent, distributed, and fault-tolerant

applications on the JVM and is written in Scala, with language bindings provided for both Scala and Java. Akka's approach to handling concurrency is based on the Actor Model. In an actor-based system, everything is an actor, in much the same way that everything is an object in object-oriented design. A key difference, though – particularly relevant to our discussion – is that the Actor Model was specifically designed and architected to serve as a concurrent model whereas the object-oriented model is not or in other words, in a Scala actor system, actors interact and share information, without any presupposition of sequentiality. The mechanism by which actors share information, and task one another, is message passing.

Q32. Explain what is Scala?

Scala is a general-purpose high-level programming language that combines object-oriented and functional programming in one concise. Scala was designed by Martin Odersky and supports Inferred, static, strong, structural typing discipline.

Please Visit OnlineInterviewquestions.com to download more pdfs