

[Scala 3 Reference](#) / [New Types](#) / [Union Types](#)

LEARN

INSTALL

PLAYGROUND

FIND A LIBRARY

COMMUNITY

BLOG

Union Types

[Edit this page on GitHub](#)

A union type `A | B` has as values all values of type `A` and also all values of type `B`.

```
case class UserName(name: String)
case class Password(hash: Hash)

def help(id: UserName | Password) =
  val user = id match
    case UserName(name) => lookupName(name)
    case Password(hash) => lookupPassword(hash)
  ...
```

Union types are duals of intersection types. `|` is *commutative*: `A | B` is the same type as `B | A`.

The compiler will assign a union type to an expression only if such a type is explicitly given. This can be seen in the following [REPL](#) transcript:

```
scala> val password = Password(123)
val password: Password = Password(123)

scala> val name = UserName("Eve")
val name: UserName = UserName(Eve)

scala> if true then name else password
val res2: Object = UserName(Eve)

scala> val either: Password | UserName = if true then name else password
val either: Password | UserName = UserName(Eve)
```

The type of `res2` is `Object & Product`, which is a supertype of `UserName` and `Password`, but not the least supertype `Password | UserName`. If we want the least supertype, we have to give it explicitly, as is done for the type of `either`.















More details



< Interse...

Union ... >

Contributors to this page

-  duanebester
-  smarter
-  k0ala
-  aserrallerios
-  hrhino
-  felixmulder
-  odersky
-  pikinier20
-  BarkingBad
-  julienrf
-  michelou
-  DaniRey
-  TheElectronWill
-  sideefffect



Copyright (c) 2002-2022, LAMP/EPFL

