# Nightly Night: The Never Type

By Jonathan Louie

# Topics

**What is the Never type?**

**Why is the Never type useful?**

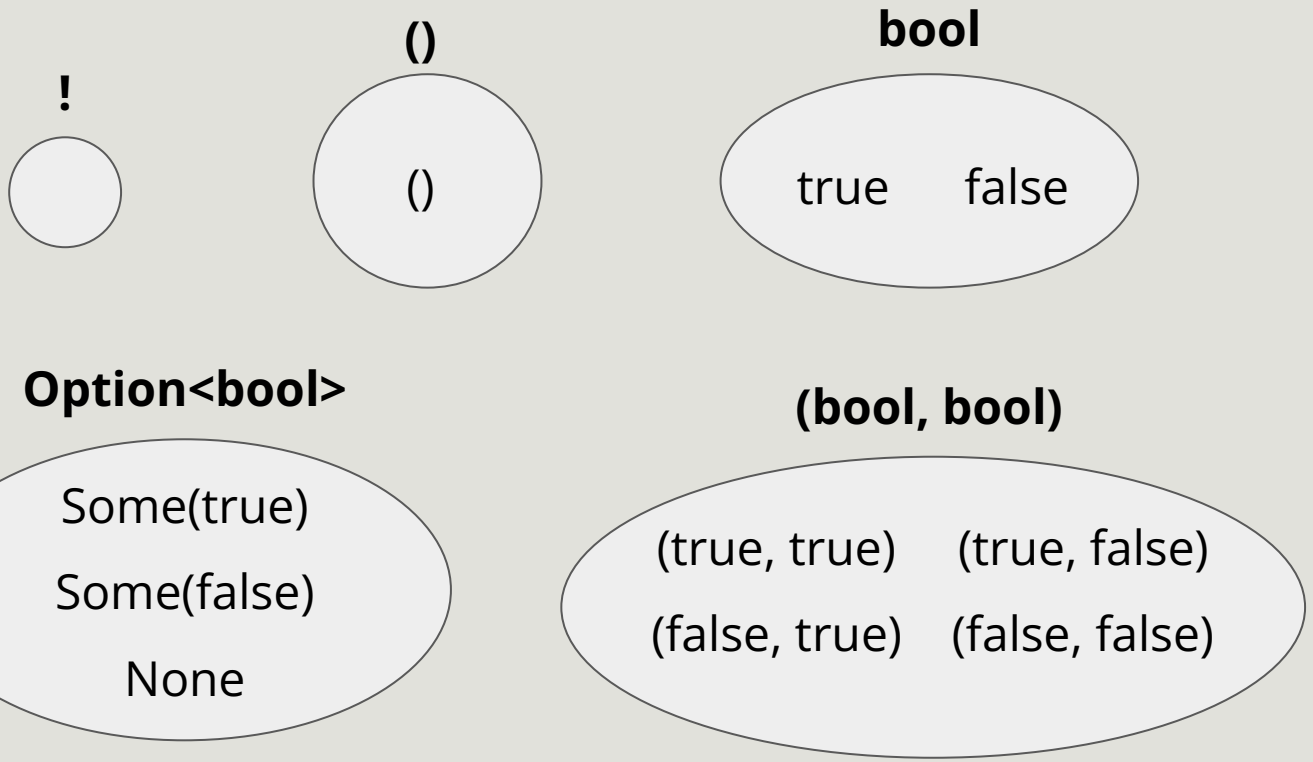
**Why is the Never type unstable?**

# What is the Never type?

# What is the Never type?

- Equivalent to an enum with no variants
- An empty type (uninhabited)
- Is written out in Rust as “!”
- NOT the bottom type that used to exist in Rust a long time ago

# Types as Sets



# Some properties of empty types

- They don’t exist at runtime
- Code handling them cannot execute
- They represent return type of functions that don’t execute
  - This includes `std::process::exit` and `break`, `continue` and `return`
- They can be converted to any other type

# Divergent functions vs. ! as a type

https://play.rust-lang.org/?version=nightly&mode=debug&edition=2021&gist=8c64a36e922df5f1f1357dcb25e9447d

# Infallible

- An empty enum in std::convert
- Used as an error type for Result when no error can happen
- Will be turned into an alias for ! and deprecated eventually
- More details: https://doc.rust-lang.org/std/convert/enum.Infallible.html

# Why is the Never type useful?

# Why is the Never type useful?

- **Can be used with Result to indicate no error can occur, which allows us to avoid usage of `unreachable!()` which often masks bugs**
- **Can implement any trait for !**
  - **Example: https://play.rust-lang.org/?version=nightly&mode=debug&edition=2021&gist=c298ea5b77871778a7a4babcef4ba496**
- **Improved dead code detection**
  - **Example: code that follows a panic**

# Why not use an empty enum?

- It’s simpler to have a standard empty type for use throughout Rust code
  - Libraries could each define their own empty types, creating busy work for apps
- Improved dead code detection requires a canonical empty type
- Simplifies code: https://rust-lang.github.io/never-type-initiative/RFC.html

# Comparison with other languages

- **More composable as a type:** https://stackoverflow.com/questions/51832396/why-does-rust-have-a-never-primitive-type

# Why is the Never type unstable?

# Why is the Never type unstable?

- **Tracking issue: https://github.com/rust-lang/rust/issues/35121**
- **Regressions caused by fallback to () type**
- **Example: https://play.rust-lang.org/?version=nightly&mode=debug&edition=2021&gist=f8b1d9e5ec9877fd680e18a0489021d1**

# References


https://github.com/rust-lang/rust/issues/35121

https://github.com/rust-lang/rust/issues/67225

https://doc.rust-lang.org/rust-by-example/fn/diverging.html

https://rust-lang.github.io/never-type-initiative/RFC.html

https://stackoverflow.com/questions/73262372/return-value-from-loop-expression-with-break

https://stackoverflow.com/questions/51832396/why-does-rust-have-a-never-primitive-type