Nightly Night: impl Trait in Type Aliases

By Jonathan Louie

Topics

Type Aliases

Impl Trait

Impl Trait in Type Aliases

Type Aliases

Type Aliases

- Defines a new name for an existing type
- Often used in place of longer type names
- Can be used for refactoring
- Is not considered separate type from aliased type (use newtype pattern if you want this)

Example

https://play.rust-lang.org/?version=stable&mode=debug&edition=2021&g ist=57653df8db7fea3b310f160b2f41ab3d

Impl Trait

What is Impl Trait?

- Represents a value of some hidden type that implements a trait
 - Can be used to avoid extra pointer indirections and allocations
- Used for nameless types (closures, for example)
- It's actually kind of complicated
 - https://varkor.github.io/blog/2018/07/03/existential-types-in-rust.html
 - Impl Trait is contextual and its meaning changes depending on where it is used

Some History

https://github.com/rust-lang/rfcs/pull/2444

https://www.reddit.com/r/rust/comments/8ldh0q/regarding visibility of
rfcs/

https://web.archive.org/web/20190408003051/http://aturon.github.io/2018/05/25/listening-part-1/

Where Impl Trait is Used

- On stable: 2 main positions
 - Argument position (APIT)
 - Return position (RPIT)
- In the future:
 - https://rust-lang.github.io/impl-trait-initiative/explainer/where ok.html

Example

https://play.rust-lang.org/?version=stable&mode=debug&edition=2021&g ist=e253ec8a98895641f4ab2d7fb00767c7

Impl Trait in Type Aliases

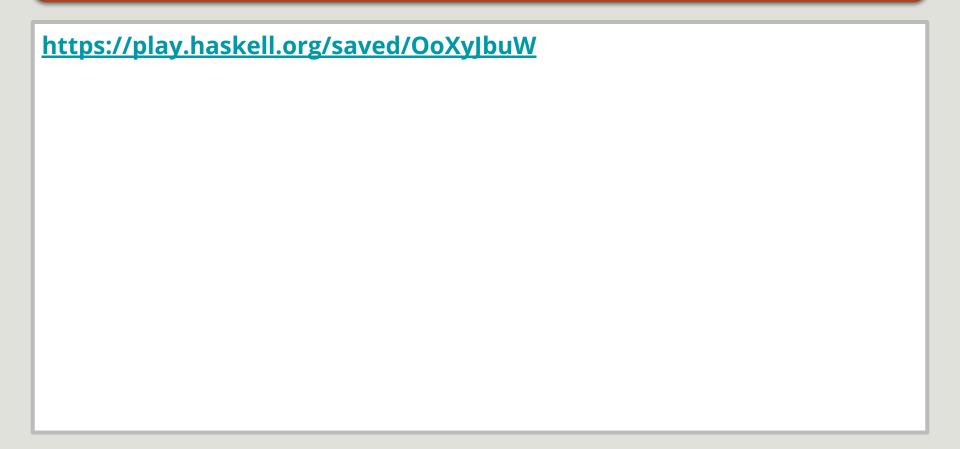
Impl Trait in Type Aliases

- Allows Impl Trait to be used in Type Aliases
- The type alias defined is used like impl Trait in return position
- When used in argument position, the alias does not behave like APIT
- Once constrained to a concrete type, can be used in various places
 - let bindings, type definitions, etc.
- Warning: Be careful when using it in refactoring!
 - https://github.com/rust-lang/rfcs/blob/master/text/2515-type alias impl trait.md# drawbacks

Experiments

https://play.rust-lang.org/?version=nightly&mode=debug&edition=2021&gist=1e97c3e457f595f5dee36258fe564b98

Comparison with Haskell



More Practical Example

https://play.rust-lang.org/?version=nightly&mode=debug&edition=2021&gist=38634d89c15d4742dfb97152de8bfca1

Why is this not Stable yet?

https://github.com/rust-lang/rust/issues/63063

Further Reading

https://varkor.github.io/blog/2018/07/03/existential-types-in-rust.html

https://serokell.io/blog/universal-and-existential-quantification

https://github.com/rust-lang/rfcs/blob/master/text/2515-type alias impl trait.md