**Immediately Calling a Closure In Rust**

[[Dylan Muraco](https://medium.com/@dmuraco?source=post_page-----afb5caa98af8--------------------------------)](https://medium.com/@dmuraco?source=post_page-----afb5caa98af8--------------------------------)

[Dylan Muraco](https://medium.com/@dmuraco?source=post_page-----afb5caa98af8--------------------------------)

·

Follow

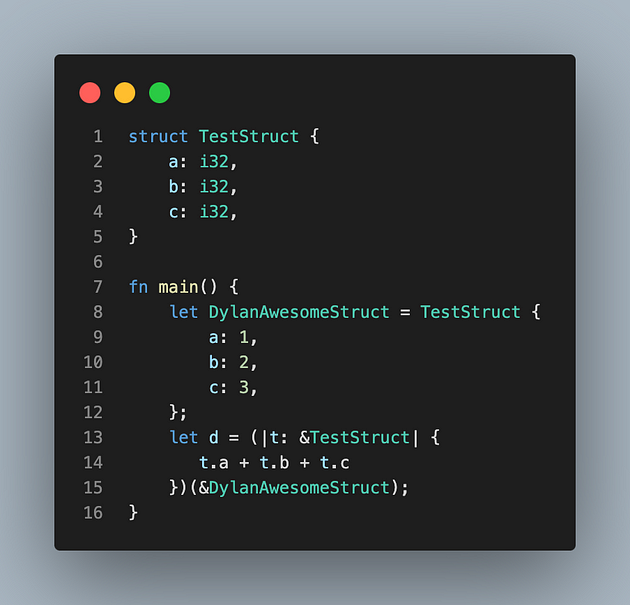
2 min read

·

5 days ago

15

2



Immediate Closure Calls in Action

During my development journey with [rust\_grad](https://github.com/dmuraco3/rust_grad), my deep learning library in Rust, I stumbled upon an intriguing syntax feature that allowed me to call closures instantly. In Rust, closures are powerful constructs that encapsulate functionality, enabling flexible and concise code.

What caught my attention was the succinctness and elegance of the syntax that enables immediate invocation of closures. Consider the following code snippet:

fn main() {  
 (|| {  
 println!("I'm printing from an immediate macro invocation");  
 })();  
}

This seemingly compact structure is, in fact, a closure declaration wrapped within parentheses and immediately followed by another set of parentheses, effectively invoking the closure. What’s fascinating about this approach is its ability to execute the code block defined within the closure without assigning it to a variable or binding it to a name.

This pattern can be incredibly useful in situations where a one-time execution of a block of code is required without the necessity of reusing the closure elsewhere in the codebase. It’s a concise way to encapsulate functionality that doesn’t need to be reused and executes seamlessly within its scope.

Rust’s expressive syntax and powerful features continually surprise and delight, making it a language where even seemingly minor syntax nuances can unveil powerful capabilities.

Have you used this immediate closure invocation syntax in your Rust projects? Share your experiences and thoughts in the comments below — I’d love to hear about different applications and insights into leveraging Rust’s features!

[Rust](https://medium.com/tag/rust?source=post_page-----afb5caa98af8---------------rust-----------------)

[Development](https://medium.com/tag/development?source=post_page-----afb5caa98af8---------------development-----------------)

[Software Development](https://medium.com/tag/software-development?source=post_page-----afb5caa98af8---------------software_development-----------------)

[Backend Development](https://medium.com/tag/backend-development?source=post_page-----afb5caa98af8---------------backend_development-----------------)

[Rust Programming Language](https://medium.com/tag/rust-programming-language?source=post_page-----afb5caa98af8---------------rust_programming_language-----------------)