

泛型

Rust中泛型无运行时性能开销 即 编译器会为每个实际使用的泛型类型生成专门的具体版本代码，而不是在运行时动态处理，这称为 单态化(monomorphization)

结构体、方法都可以使用泛型

```
struct BrowserCommand<T> {  
    name: String,  
    payload: T,  
}
```

Run | Debug

```
fn main() {  
    let cmd1: BrowserCommand<String> = BrowserCommand {  
        name: "navigate".to_owned(),  
        payload: "https://www.letsgetrusty.com".to_owned(),  
    };  
    let cmd2: BrowserCommand<i32> = BrowserCommand {  
        name: "zoom".to_owned(),  
        payload: 200  
    };  
}
```

以T开头的大写字母，代表类型。

```
struct BrowserCommand<T> {  
    name: String,  
    payload: T,  
}
```

```
impl<T> BrowserCommand<T> {  
    fn new(name: String, payload: T) -> Self {  
        BrowserCommand {  
            name,  
            payload,  
        }  
    }  
}
```

- 注意使用泛型 与 具体类型实现时的签名

```
impl<T> BrowserCommand<T> {  
    fn new(name: String, payload: T) -> Self  
    {  
        BrowserCommand {  
            name,  
            payload,  
        }  
    }  
}
```

```
impl BrowserCommand<String> {  
    fn print_payload(&self) {  
        println!("{}", self.payload);  
    }  
}
```

打印有效负载引用self并打印