# Rust 编程语言入门


杨旭，微软MVP
Rust、Go、C#开发者

# 10.2 泛型

# 泛型

- 泛型：提高代码**复用**能力
  - 处理重复代码的问题
- 泛型是具体类型或其它属性的抽象代替：
  - 你编写的代码不是最终的代码，而是一种**模板**，里面有一些“**占位符**”。
  - 编译器在**编译时**将“占位符”**替换为具体的类型**。
- 例如：fn largest<**T**>(list: &[**T**]) -> **T** { … }
- 类型参数：
  - 很短，通常一个字母
  - CamelCase
  - T：type 的缩写

# 函数定义中的泛型

- 泛型函数：
  - 参数类型
  - 返回类型
- （例子）

# Struct 定义中的泛型

- （例子）
- 可以使用多个泛型的类型参数
  - 太多类型参数：你的代码需要重组为多个更小的单元

# Enum 定义中的泛型

- 可以让枚举的变体持有泛型数据类型
  - 例如 Option<T>，Result<T, E>
- （例子）

# 方法定义中的泛型

- 为 struct 或 enum 实现方法的时候，可在定义中使用泛型
- （例子）
- 注意：
  - 把 T 放在 impl 关键字后，表示在类型 T 上实现方法
    - 例如：impl<T> Point<T>
  - 只针对具体类型实现方法（其余类型没实现方法）：
    - 例如：impl Point<f32>
- struct 里的泛型类型参数可以和方法的泛型类型参数不同
  - （例子）

# 泛型代码的性能

- 使用泛型的代码和使用具体类型的代码运行速度是一样的。
- 单态化（monomorphization）：
  - 在编译时将泛型替换为具体类型的过程
- （例子）

再见