

Fundamentals | Turbofish

■ What?

- ◆ Sometimes the compiler cannot determine the type of some data
- ◆ A few options are available when this happens:
 - Type annotations
 - Turbofish

■ Type Annotations Review

```
let numbers: Vec<u32> = vec![1, 2, 3];
```

```
let numbers: Vec<_> = vec![1, 2, 3];
```

```
let odds: Vec<_> = numbers
```

```
    .iter()
```

```
    .filter(|n| **n % 2 == 1)
```

```
    .collect();
```

■ Missing Type Annotation

```
let numbers: Vec<_> = vec![1, 2, 3];  
let odds = numbers  
    .iter()  
    .filter(|n| **n % 2 == 1)  
    .collect();
```


Turbofish

```
let numbers: Vec<_> = vec![1, 2, 3];  
let odds = numbers  
    .iter()  
    .filter(|n| **n % 2 == 1)  
    .collect::
```

■ Syntax

```
ident :: <type>  
      :: <>
```

■ When Turbofish Can Be Used

- ◆ Any item having a generic parameter

```
pub fn collect<B>(self) -> B  
    collect::<>()
```

■ Recap

- ◆ Turbofish is a way to specify a type when working with generics
 - Only needed if the compiler cannot determine the type being used
- ◆ Usually optional; type annotations suffice