



More Structs

CS128 Honors

Slides by Matt Geimer (FA21)
Presented 10/4/2021



Making functions for structs

- In Java, we can write classes that have instance methods
- In Rust, we can accomplish the same thing with structs

```
struct Rectangle {  
    width: u32,  
    height: u32,  
}
```



Making functions for structs

```
struct Rectangle {  
    width: u32,  
    height: u32,  
}  
  
impl Rectangle {  
    fn area(&self) -> u32 {  
        self.width * self.height  
    }  
}
```

We use `impl` to tell Rust we're implementing in the context of the `Rectangle` struct

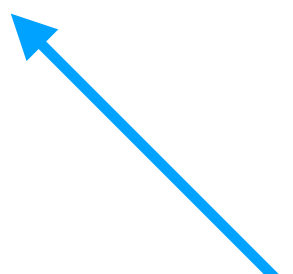
The first argument of every method function in Rust **must be `&self`** (can also have `&mut self`)



Making functions for structs

```
struct Rectangle {  
    width: u32,  
    height: u32,  
}  
  
impl Rectangle {  
    fn area(&self) -> u32 {  
        self.width * self.height  
    }  
}  
  
fn main() {  
    let rect1 = Rectangle {  
        width: 30,  
        height: 50,  
    };  
  
    println!(  
        "The area of the rectangle is {} square pixels.",  
        rect1.area()  
    );  
}
```

We can then use dot notation syntax
to call the function instead of doing
`area(&rect1)`





Sidenotes on method functions

- `impl` blocks can have multiple functions declared inside of them
- A struct can have multiple `impl` blocks
- Method functions can have more than just `&self` arguments
- Method functions can also **not** take `&self` (useful for constructors)



An Example