Static vs dynamic dispatch

- Dispatch
  - Dispatch = How the compiler determines which function to call when a trait method is invoked.
  -

	Type	Determined At	Performance	Flexibility
1	✅ Static Dispatch	Compile Time	Fast	Less flexible
2	✅ Dynamic Dispatch	Runtime	Slower	More flexible

  - static
    - The compiler generates separate code for each type of T
      - mono-morphization
      -

```
fn print_value<T: std::fmt::Debug>(value: T) {
    println!("{:?}", value);
}
fn main() {
    print_value(42);           // i32
    print_value("Hello");      // &str
}
// when compiled

fn print_value_i32(value: i32) {
    println!("{:?}", value);
}

fn print_value_str(value: &str) {
    println!("{:?}", value);
}



```

    - No runtime cost (method call is resolved at compile time)

  - dynamic
    - Method call uses a vtable (virtual function table)
    - Extra runtime indirection to look up method
    - Useful when you don’t know the concrete type in advance