

Session Types

What They Are and Why They Matter

What Are Session Types?

Session types are like types for conversations.

They describe the *structure* of communication between two or more parties.

- ▶ Specify message **order**
- ▶ Specify message **type**
- ▶ Prevent communication mismatches at compile time

Example Scenario

Client-Server Authentication Protocol

Steps:

1. Client sends "LOGIN"
2. Server sends "Username?"
3. Client sends username
4. Server sends "Password?"
5. Client sends password
6. Server responds with "Success" or "Failure"
7. Session ends

Server Session Type

```
?LOGIN;  
!Username?;  
?String;  
!Password?;  
?String;  
!(Success | Failure);  
end
```

- ▶ ? means receive
- ▶ ! means send

Client Session Type (Dual)

Each side's behavior is dual (send \leftrightarrow receive)

```
!LOGIN;  
?Username?;  
!String;  
?Password?;  
!String;  
?(Success | Failure);  
end
```

What Happens Without Session Types?

A buggy client might:

```
send("LOGIN")  
send("WrongOrderData")  # mistake: didn't wait for prompt  
send("Username")  
send("Password")  
response = receive()
```

What Goes Wrong?

- ▶ Messages sent in the wrong order
- ▶ Server misinterprets data
- ▶ No error until runtime
- ▶ Authentication fails or the server crashes

What Session Types Prevent

- ▶ Sending before receiving
- ▶ Receiving unexpected messages
- ▶ Mismatched types
- ▶ Deadlocks from bad coordination

Summary

With Session Types	Without Session Types
Checked at compile time	Fails at runtime
Order is enforced	Order must be remembered
Dual types align roles	Manual protocol matching
Communication is safe	Easy to desynchronize

Takeaway

Session types give you:

- ▶ A blueprint for communication
- ▶ Compile-time guarantees
- ▶ Fewer bugs in concurrent systems

Build conversations that work by design.