From dependency injection to dependency rejection

Mark Seemann

http://blog.ploeh.dk

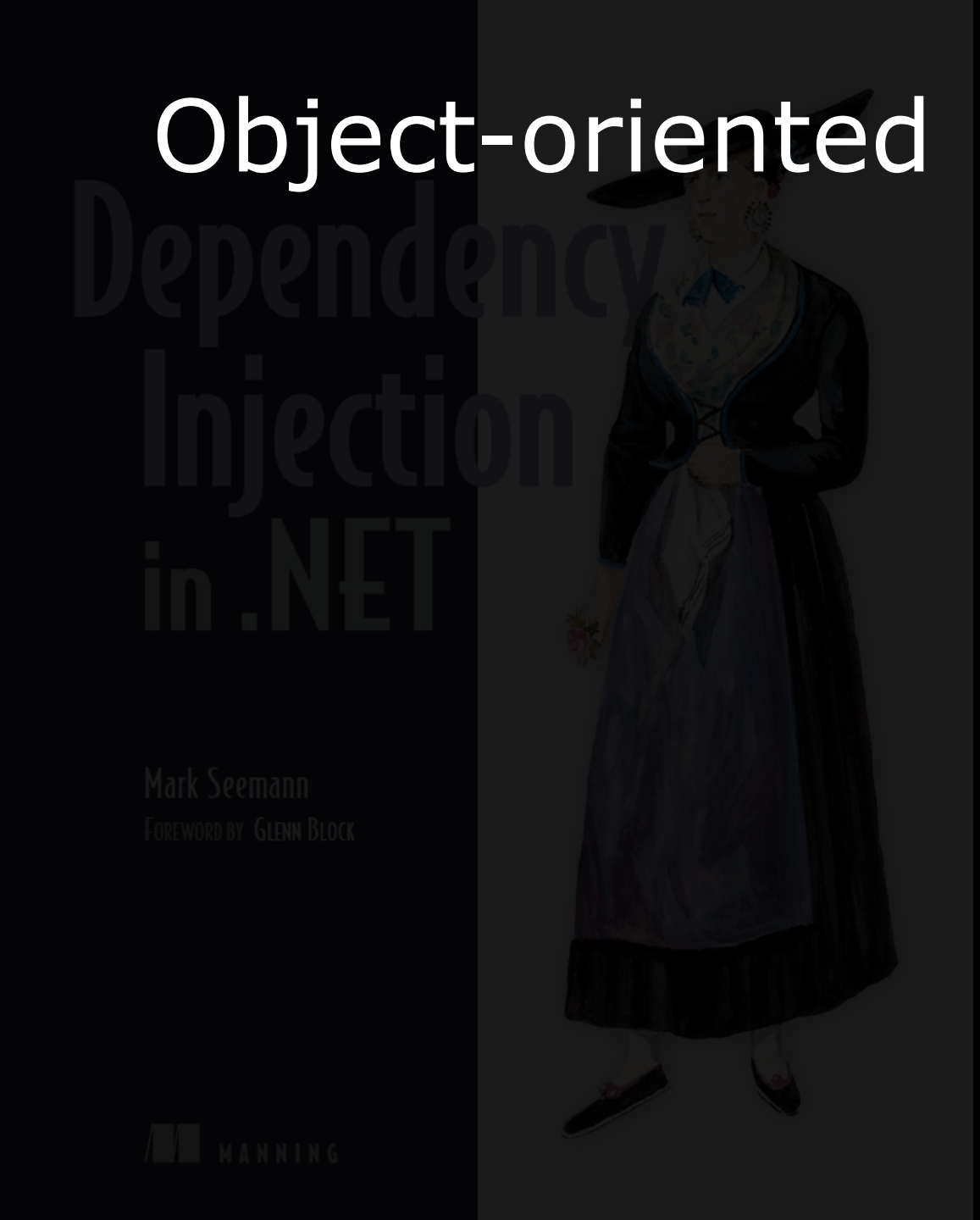
@ploeh

Dependency Injection in .NET

Mark Seemann
Foreword by Glenn Block

MANNING

Object-oriented



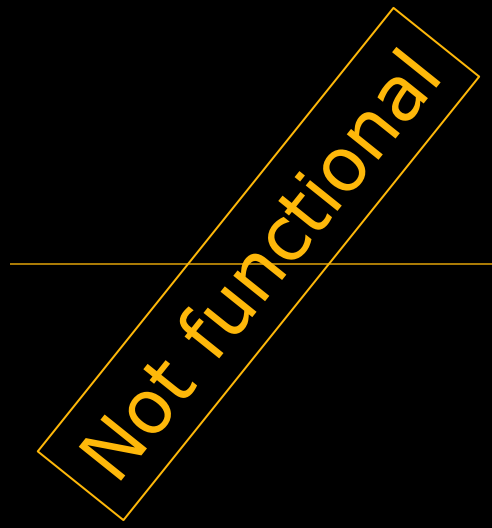
Functional

?

Object-oriented

Functional

Dependency injection



Partial application

Composition

Dependency injection is “really just a pretentious way to say ‘taking an argument’”

- Rúnar Bjarnason (@runarorama)

@ploeh

Example

Restaurant Reservation

Restaurant Reservation

Make a reservation	
Date	
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	20
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	201
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-1
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-2
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	M
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Ma
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mar
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark S
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Se
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark See
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seem
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seema
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seeman
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	ma
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mar
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@p
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@pl
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@plo
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@ploe
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@ploeh
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@ploeh.
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@ploeh.d
Quantity	
	Submit

Restaurant Reservation

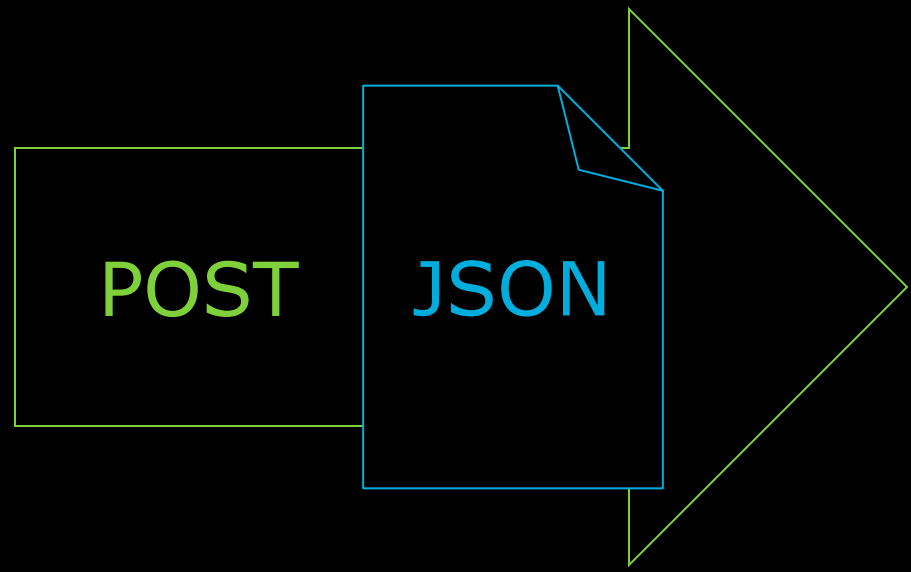
Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@ploeh.dk
Quantity	
	Submit

Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@ploeh.dk
Quantity	4
	Submit

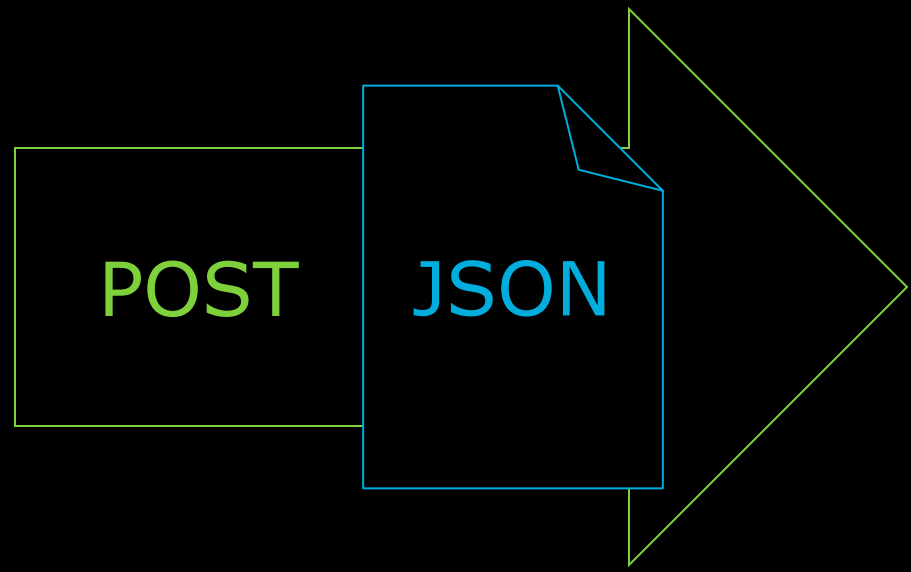
Restaurant Reservation

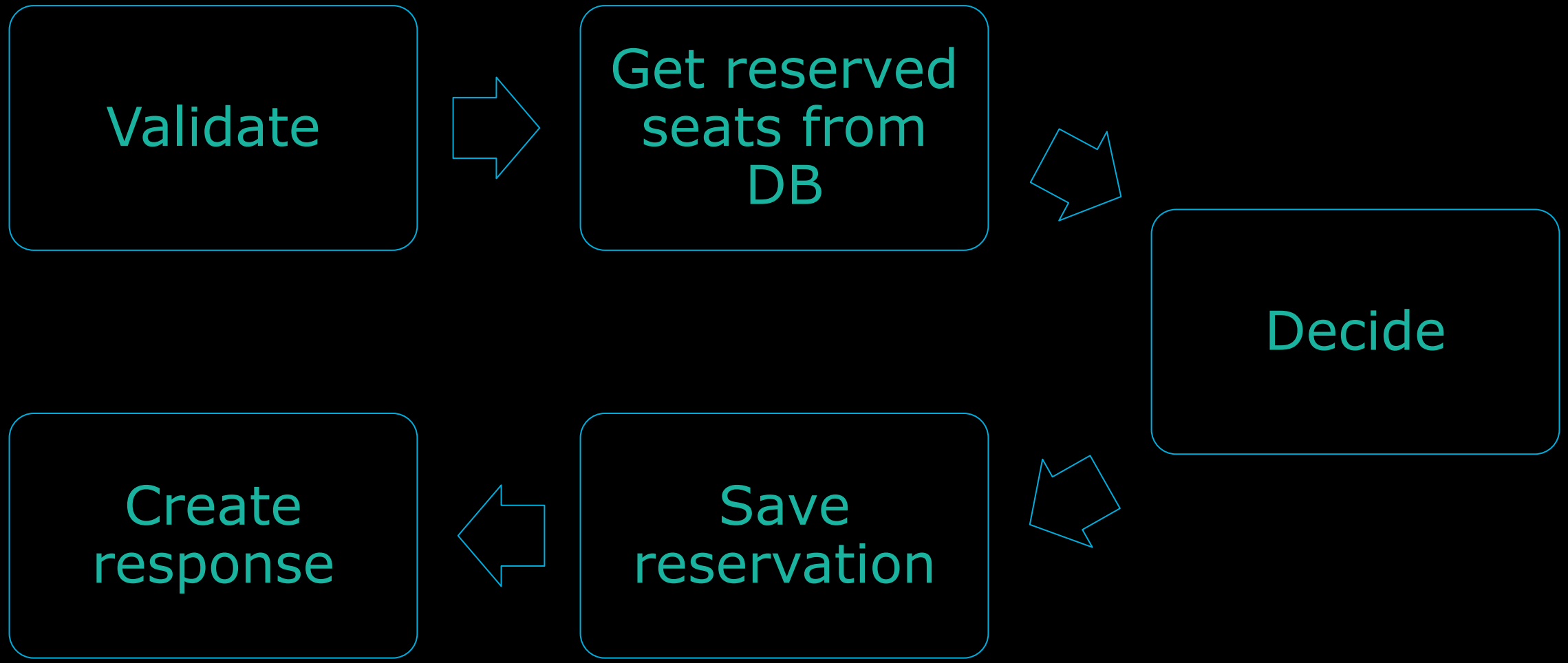
Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@ploeh.dk
Quantity	4
	Submit



Restaurant Reservation

Make a reservation	
Date	2016-12-29
Name	Mark Seemann
Email	mark@ploeh.dk
Quantity	4
	Submit




Validate
Get reserved seats from DB
Decide
Save reservation
Create response

Object-oriented

Dependency injection

```
public class ReservationsController : ApiController
{
    public ReservationsController(IValidator validator, IMapper mapper, IMaîtreD maîtreD)
    {
        this.validator = validator;
        this.mapper = mapper;
        this.maîtreD = maîtreD;
    }

    public IHttpActionResult Post(ReservationRequestDto dto)
    {
        var validationMsg = validator.Validate(dto);
        if (validationMsg != "")
            return this.BadRequest(validationMsg);

        var r = mapper.Map(dto);
        var id = maîtreD.TryAccept(r);
        if (id == null)
            return this.StatusCode(HttpStatusCode.Forbidden);

        return this.Ok();
    }
}
```

```
POST /reservations HTTP/1.1

{
  "date": "2016-12-08",
  "name": "Mark Seemann",
  "email": "mark@ploeh.dk",
  "quantity": 4
}
```

C#

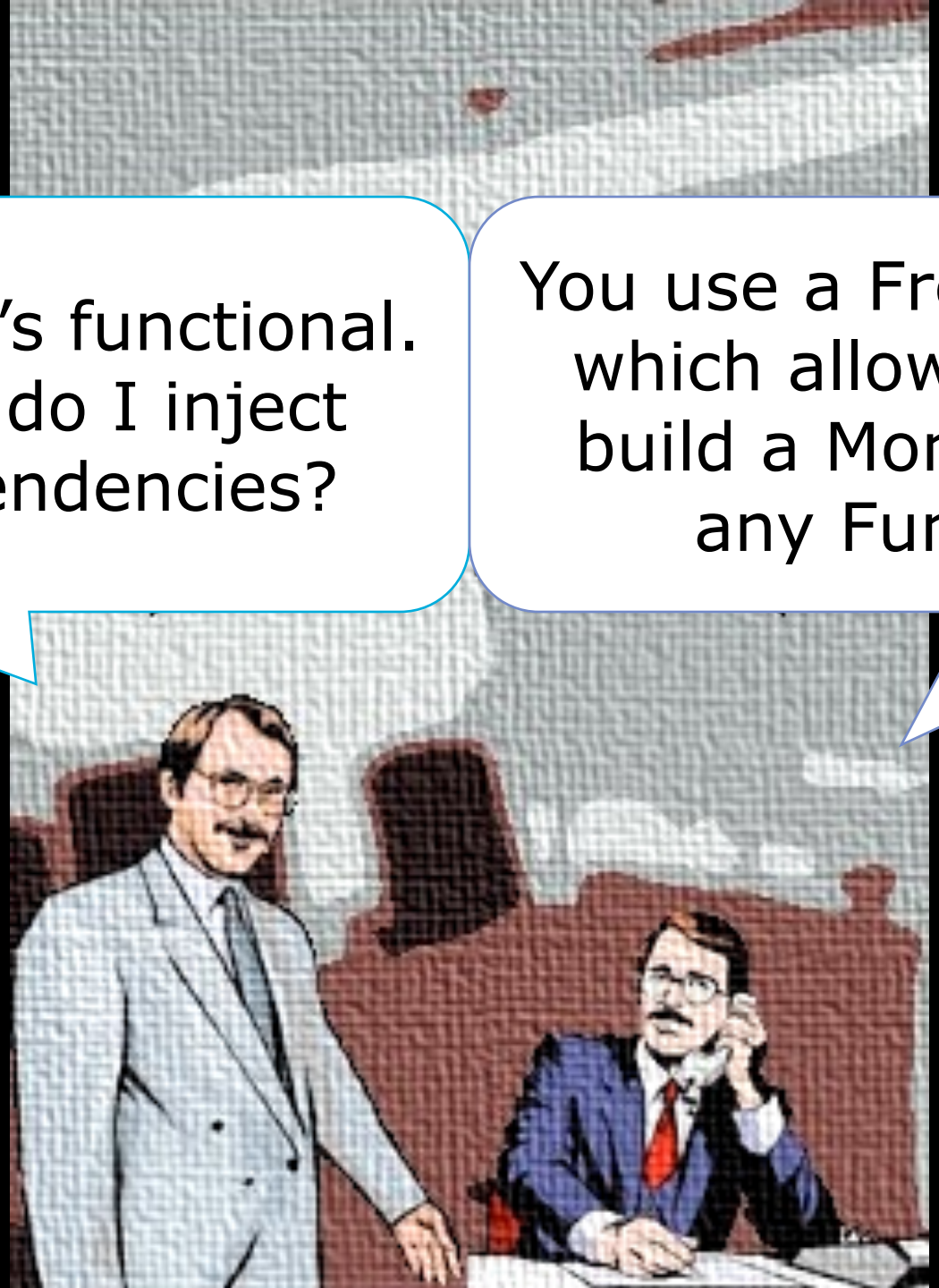
```
public class MaîtreD : IMaîtreD
{
    public MaîtreD(int capacity, IReservationsRepository reservationsRepository)
    {
        this.capacity = capacity;
        this.reservationsRepository = reservationsRepository;
    }

    public int? TryAccept(Reservation reservation)
    {
        var reservedSeats = reservationsRepository
            .ReadReservations(reservation.Date)
            .Sum(r => r.Quantity);
        if (reservedSeats + reservation.Quantity <= capacity)
        {
            reservation.IsAccepted = true;
            return reservationsRepository.Create(reservation);
        }

        return null;
    }
}
```

```
public interface IMaîtreD
{
    int? TryAccept(Reservation reservation);
}
```

C#

```
public interface IMaîtreD
{
    int? TryAccept(Reservation reservation);
}
```

Comic courtesy of
@jrecursive and @hmemcpy



Comic courtesy of @jrecursive and @hmemcpy

Comic courtesy of
@jrecursive and @hmemcpy

Partial application

An attempt at dependency injection with a functional language

Repository

```
// int -> (DateTimeOffset -> Reservation list) -> (Reservation -> int) -> Reservation
// -> int option
let tryAccept capacity readReservations createReservation reservation =
    let reservedSeats =
        readReservations reservation.Date |> List.sumBy (fun x -> x.Quantity)
    if reservedSeats + reservation.Quantity <= capacity
    then createReservation { reservation with IsAccepted = true } |> Some
    else None
```

```
string -> DateTimeOffset -> Reservation list

// Reservation -> int option
let tryAcceptComposition reservation =
    let read   = DB.readReservations  connectionString
```

```
string -> DateTimeOffset -> Reservation list
```

```
// Reservation -> int option
let tryAcceptComposition reservation =
    let read   = DB.readReservations  connectionString
```

```
connectionString -> DateTimeOffset -> Reservation list
```

```
// Reservation -> int option
let tryAcceptComposition reservation =
    let read   = DB.readReservations  connectionString
```

DateTimeOffset -> Reservation list

```
// Reservation -> int option
let tryAcceptComposition reservation =
    let read   = DB.readReservations  connectionString
```

Reservation -> int

```
// Reservation -> int option
let tryAcceptComposition reservation =
    let read   = DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reservation
```

```
// Reservation -> int option
let tryAcceptComposition reservation =
    let read   = DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reservation
```

```
// Reservation -> int option
let tryAcceptComposition             =
    let read   = DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create
```

```
// Reservation -> int option
let tryAcceptComposition =
    let read   = DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create
```

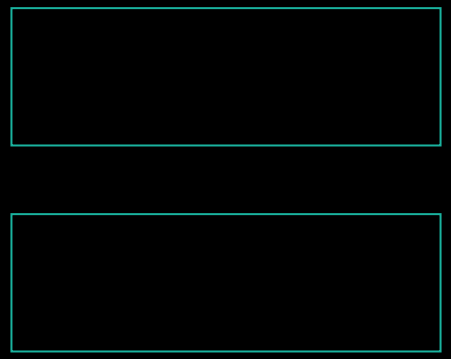

IL
C#

```
internal class tryAcceptComposition@17 : FSharpFunc<Reservation, FSharpOption<int>>
{
    public int capacity;
    public FSharpFunc<Reservation, int> createReservation;
    public FSharpFunc<DateTimeOffset, FSharpList<Reservation>> readReservations;

    internal tryAcceptComposition@17(
        int capacity,
        FSharpFunc<DateTimeOffset, FSharpList<Reservation>> readReservations,
        FSharpFunc<Reservation, int> createReservation)
    {
        this.capacity = capacity;
        this.readReservations = readReservations;
        this.createReservation = createReservation;
    }

    public override FSharpOption<int> Invoke(Reservation reservation)
    {
        return MaîtreD.tryAccept<int>(
            this.capacity, this.readReservations, this.createReservation, reservation);
    }
}
```

C#

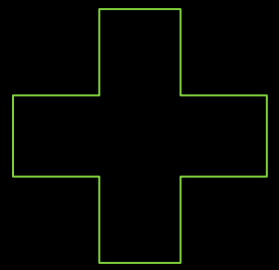
Is it Functional?

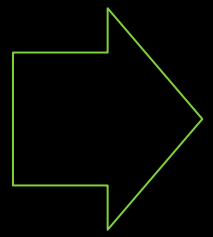
Partial application



Dependency injection

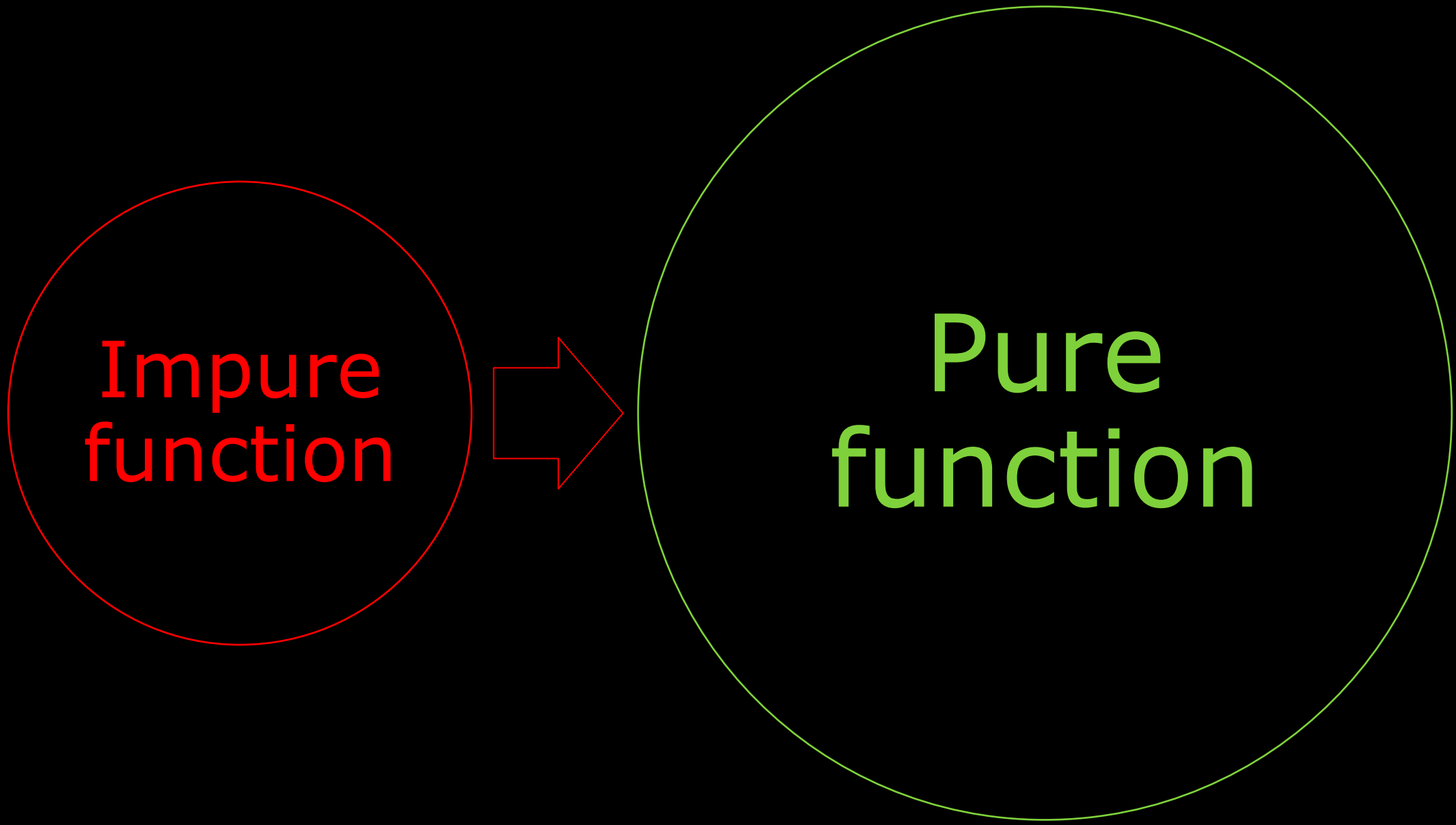
Same return value for same input



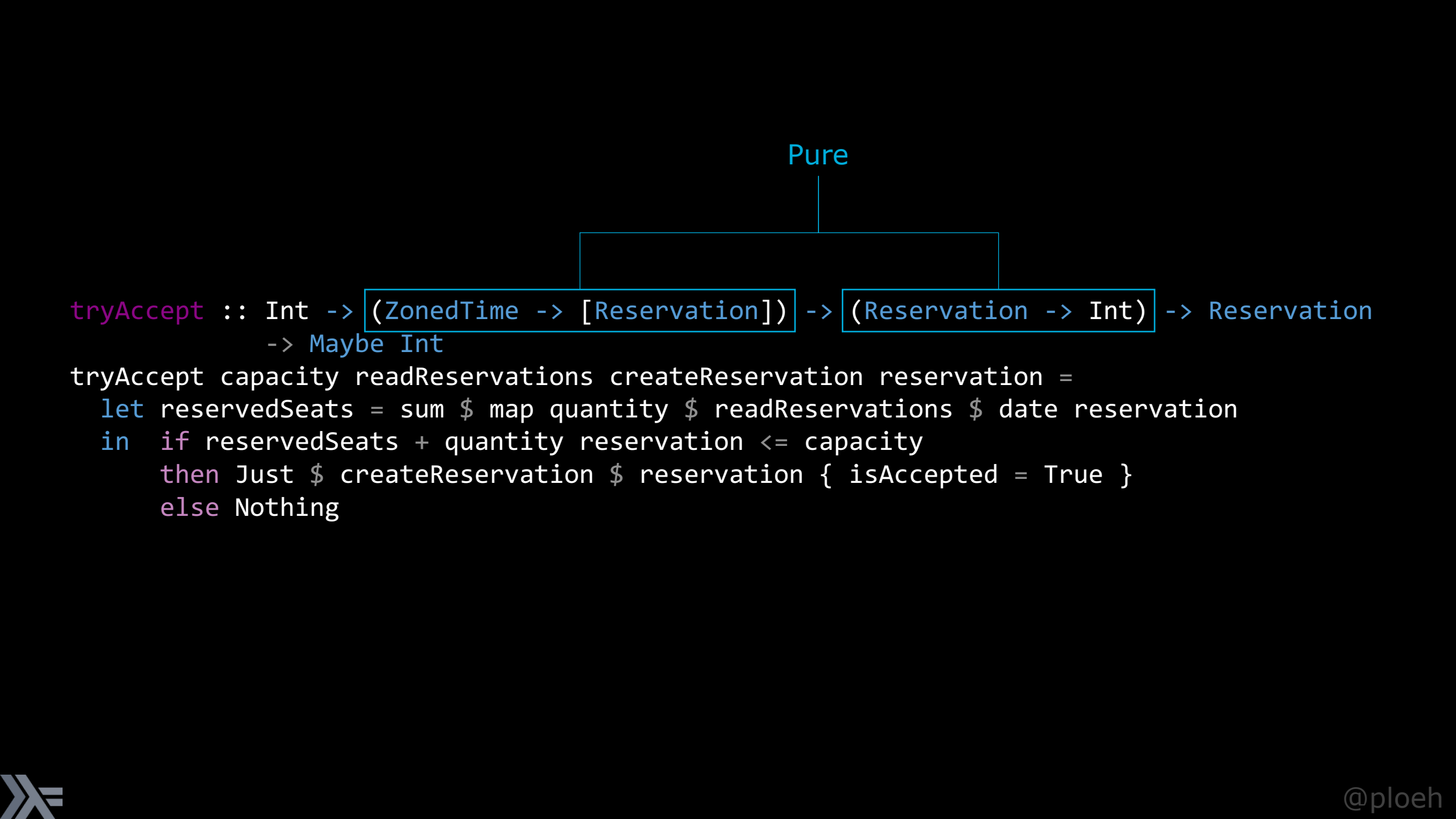


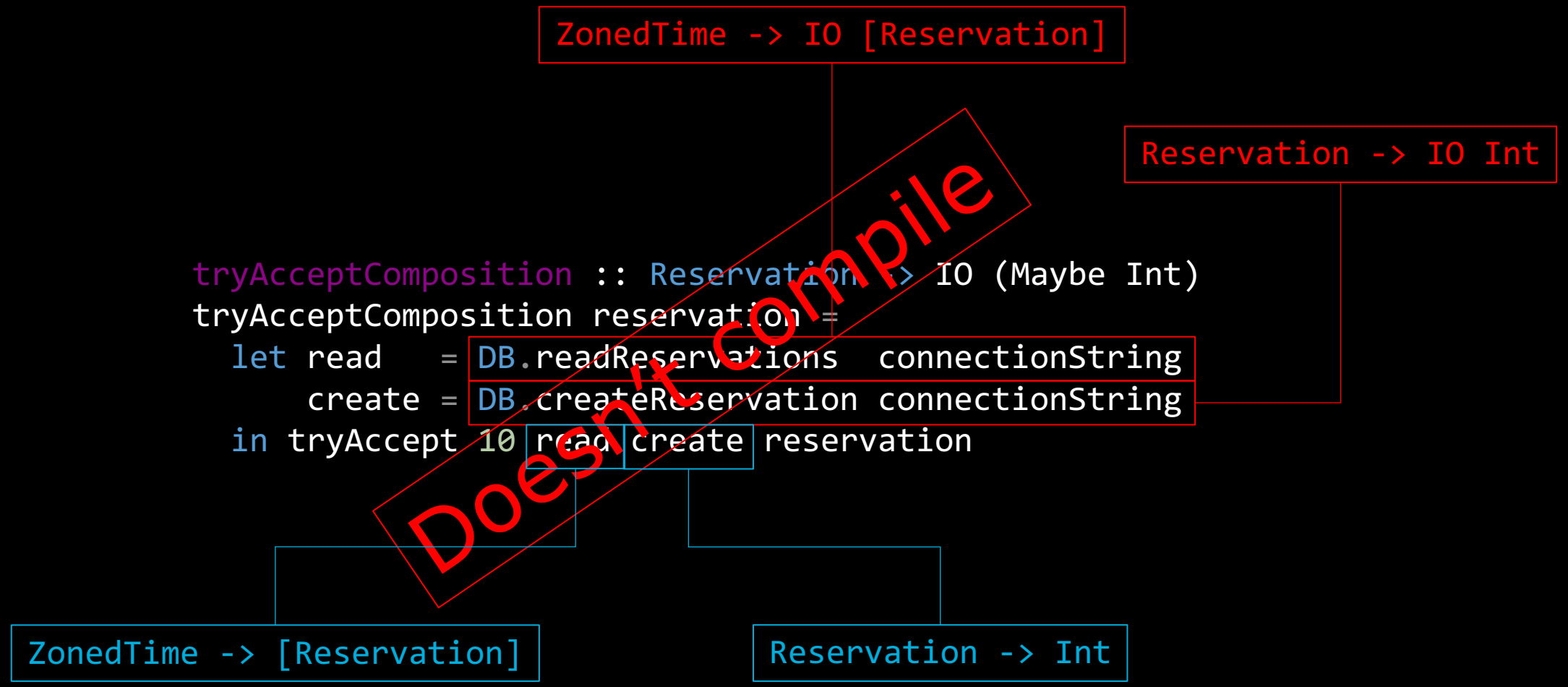
Pure function

No side-effects


Impure
function
Pure
function

Sanity check


Pure
tryAccept :: Int -> (ZonedTime -> [Reservation]) -> (Reservation -> Int) -> Reservation
-> Maybe Int
tryAccept capacity readReservations createReservation reservation =
let reservedSeats = sum $ map quantity $ readReservations $ date reservation
in if reservedSeats + quantity reservation <= capacity
then Just $ createReservation $ reservation { isAccepted = True }
else Nothing


ZonedTime -> IO [Reservation]
Reservation -> IO Int
tryAcceptComposition :: Reservation -> IO (Maybe Int)
tryAcceptComposition reservation =
let read = DB.readReservations connectionString
create = DB.createReservation connectionString
in tryAccept 10 read create reservation
Doesn't compile
ZonedTime -> [Reservation]
Reservation -> Int

```
// Reservation -> int option
let tryAcceptComposition =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create
```

```
// Reservation -> int option
let tryAcceptComposition =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create
```

```
// Reservation -> int option
let tryAcceptComposition  =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create
```

```
// Reservation -> int option
let tryAcceptComposition r =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create r
```

```
// Reservation -> int option
let tryAcceptComposition re =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create re
```

```
// Reservation -> int option
let tryAcceptComposition res =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create res
```

```
// Reservation -> int option
let tryAcceptComposition rese =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create rese
```

```
// Reservation -> int option
let tryAcceptComposition reser =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reser
```

```
// Reservation -> int option
let tryAcceptComposition reserv =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reserv
```

```
// Reservation -> int option
let tryAcceptComposition reserva =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reserva
```

```
// Reservation -> int option
let tryAcceptComposition reservat =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reservat
```

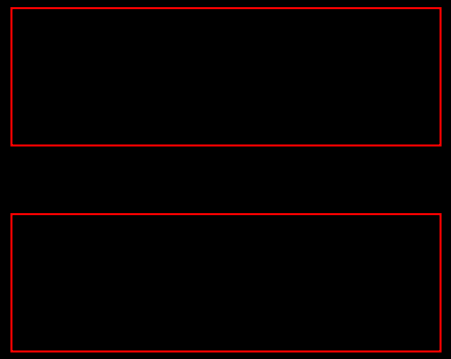
```
// Reservation -> int option
let tryAcceptComposition reservati =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reservati
```

```
// Reservation -> int option
let tryAcceptComposition reservatio =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reservatio
```

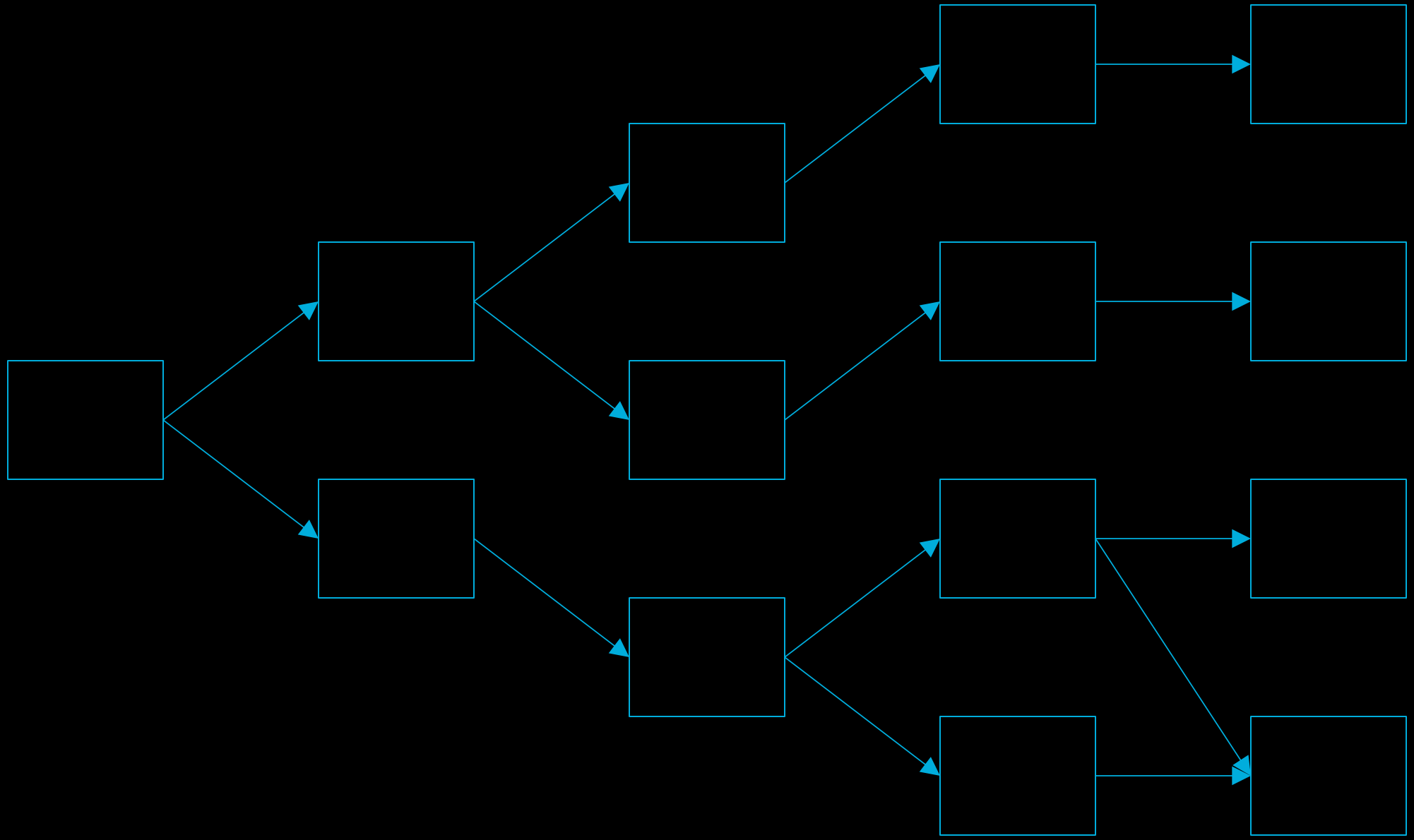
```
// Reservation -> int option
let tryAcceptComposition reservation =
    let read =   DB.readReservations  connectionString
    let create = DB.createReservation connectionString
    tryAccept 10 read create reservation
```

Not Functional

Partial application

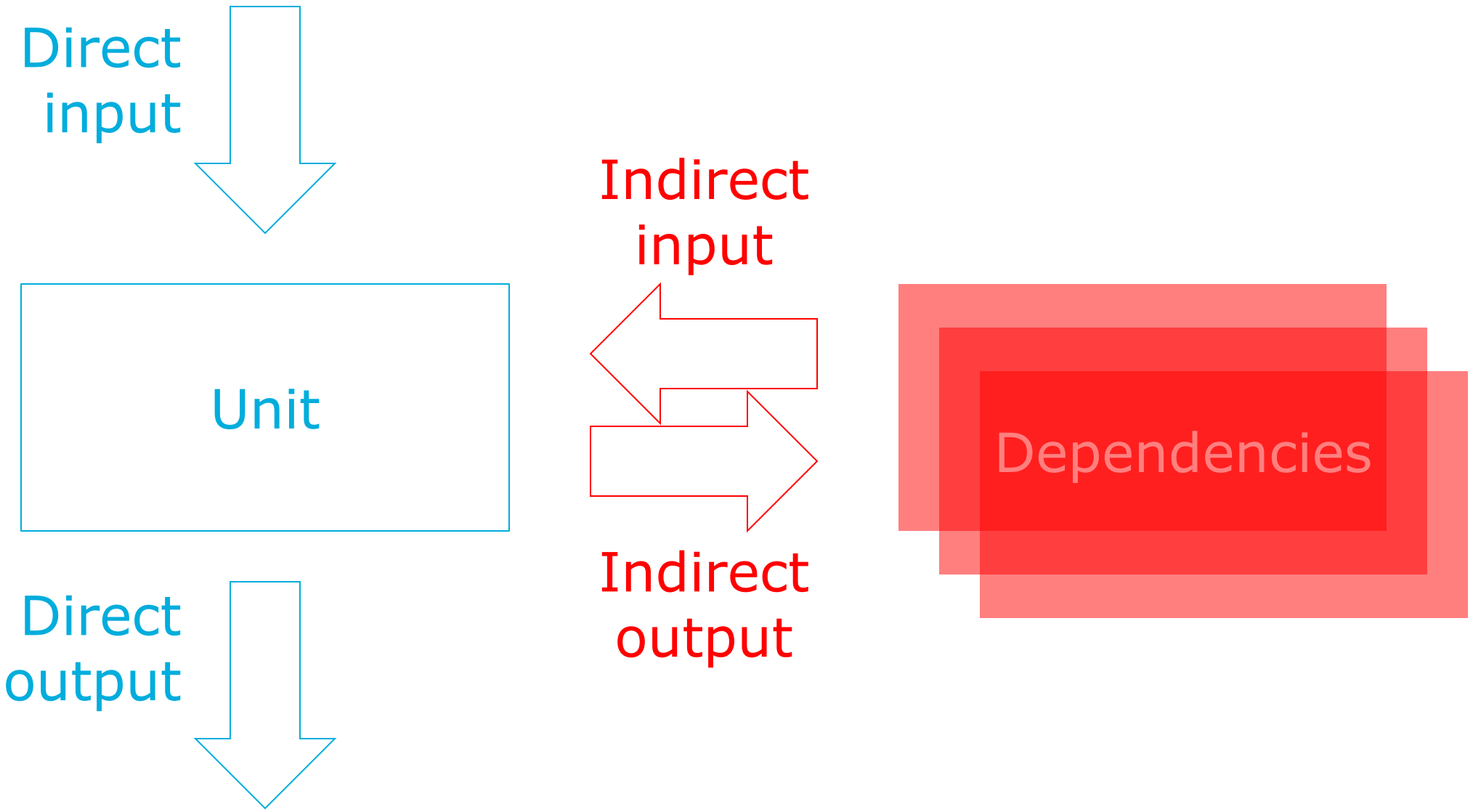


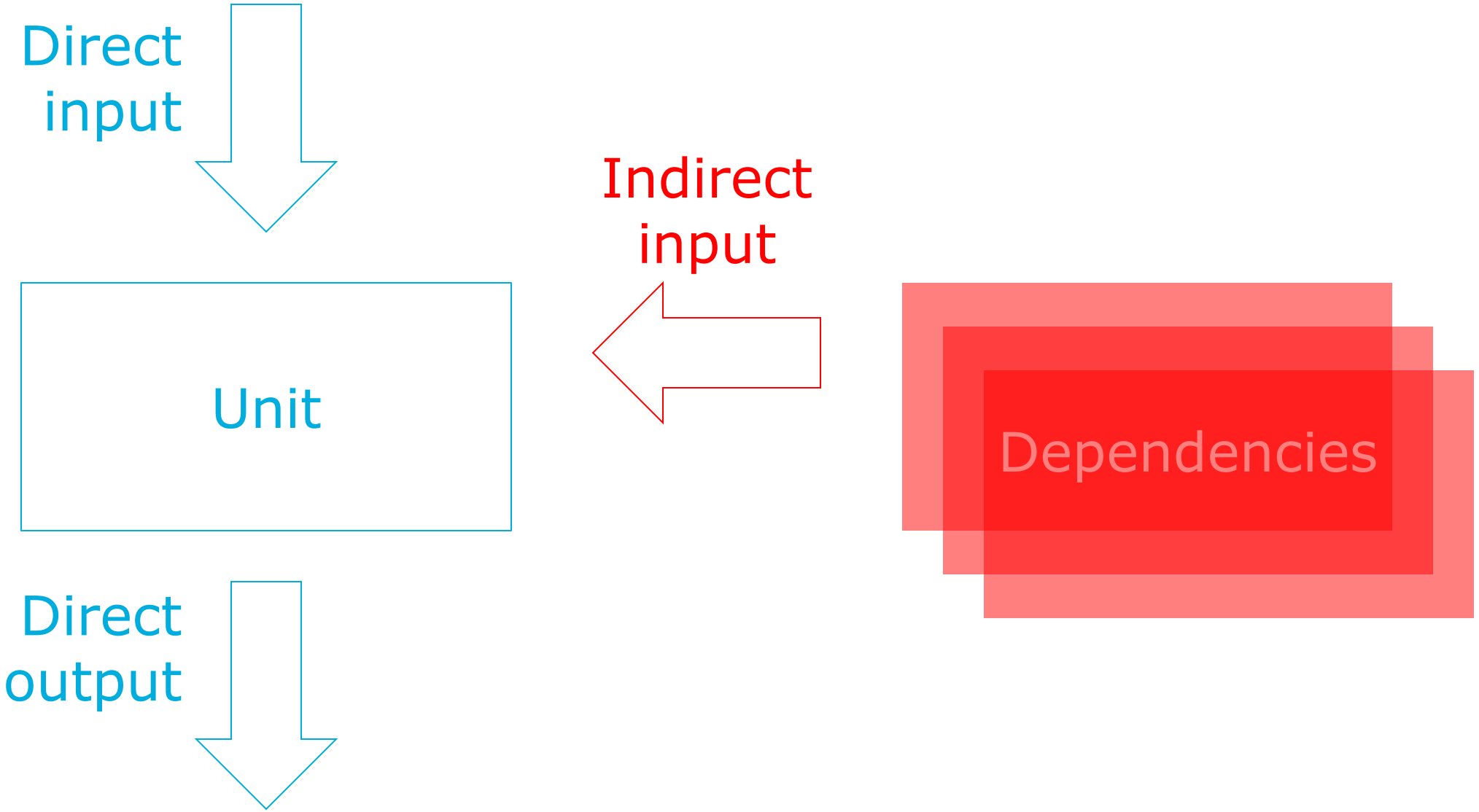
Dependency injection

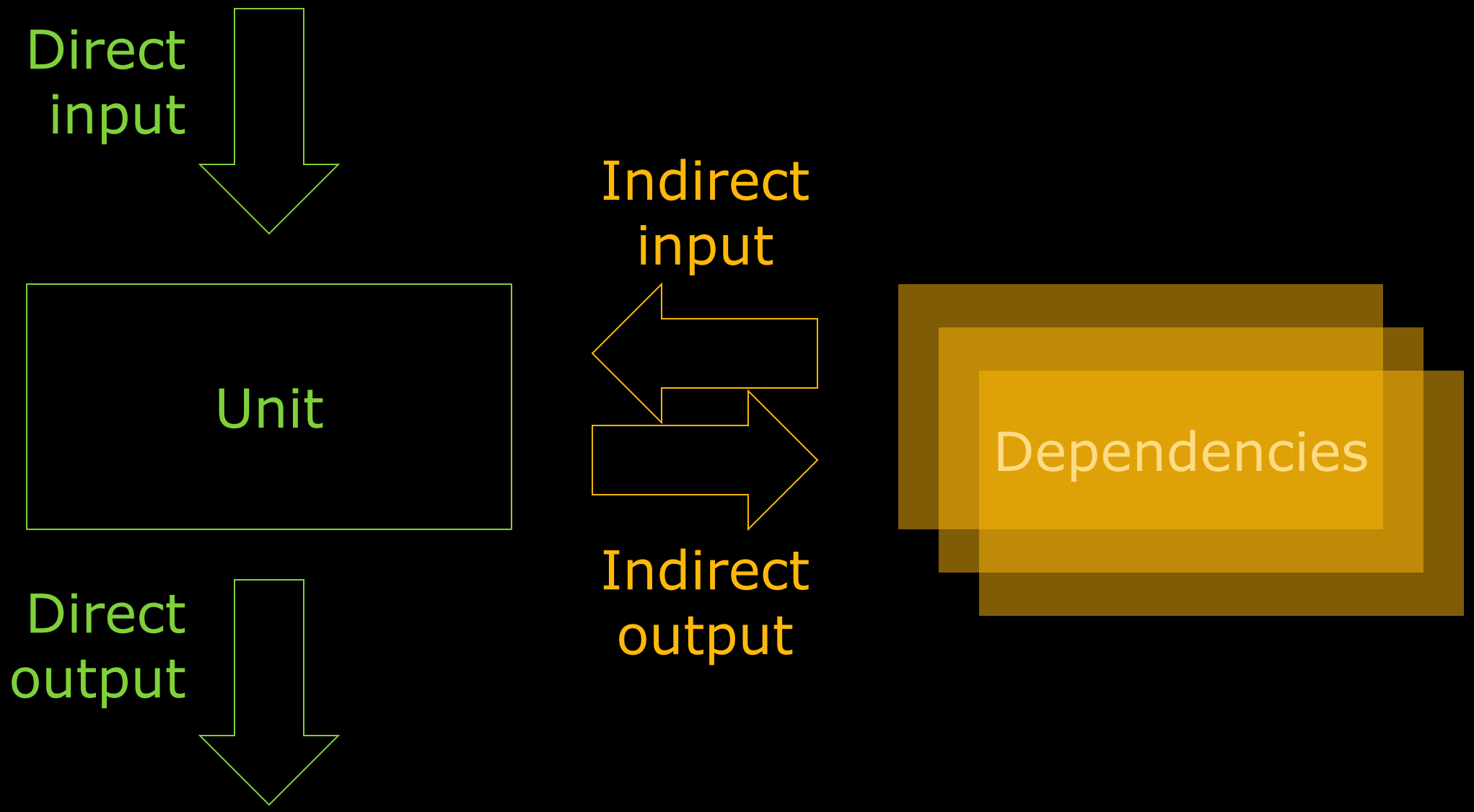


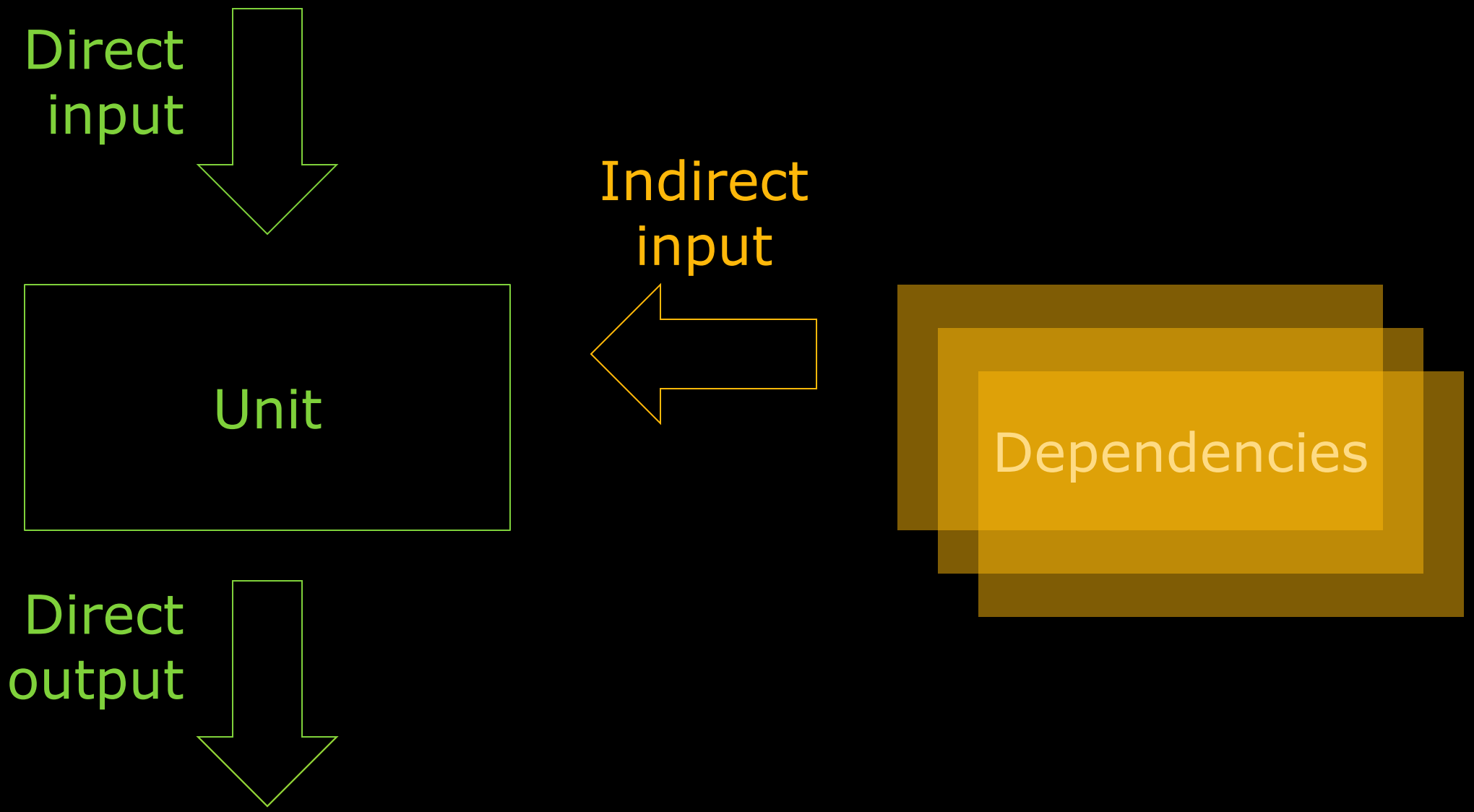
Dependency injection
makes everything impure

Dependency Rejection


Direct input
Unit
Direct output
Indirect input
Indirect output
Dependencies


Direct input
Indirect input
Unit
Dependencies
Direct output


Direct input
Unit
Direct output
Indirect input
Indirect output
Dependencies


Direct input
Indirect input
Unit
Dependencies
Direct output

```
// int -> (DateTimeOffset -> Reservation list) -> (Reservation -> int) -> Reservation
// -> int option
let tryAccept capacity readReservations createReservation reservation =
```

```
// int -> (DateTimeOffset -> Reservation list) ->                    Reservation
// -> Reservation option
let tryAccept capacity readReservations                  reservation =
```

```
// int -> (DateTimeOffset -> Reservation list) -> Reservation
// -> Reservation option
let tryAccept capacity readReservations reservation =
```

```
// int -> (DateTimeOffset -> Reservation list) -> Reservation -> Reservation option
let tryAccept capacity readReservations reservation =
```

```
// int -> (DateTimeOffset -> Reservation list) -> Reservation -> Reservation option
let tryAccept capacity readReservations reservation =
    let reservedSeats =
        readReservations reservation.Date |> List.sumBy (fun x -> x.Quantity)
    if reservedSeats + reservation.Quantity <= capacity
    then { reservation with IsAccepted = true } |> Some
    else None
```

```
// Reservation -> int option
let tryAcceptComposition reservation =
    match tryAccept 10 (DB.readReservations connectionString) reservation with
    | None -> None
    | Some r -> (DB.createReservation connectionString) r |> Some
```

```
// Reservation -> int option
let tryAcceptComposition reservation =
    tryAccept 10 (DB.readReservations connectionString) reservation
    |> Option.map (fun r -> DB.createReservation connectionString r)
```

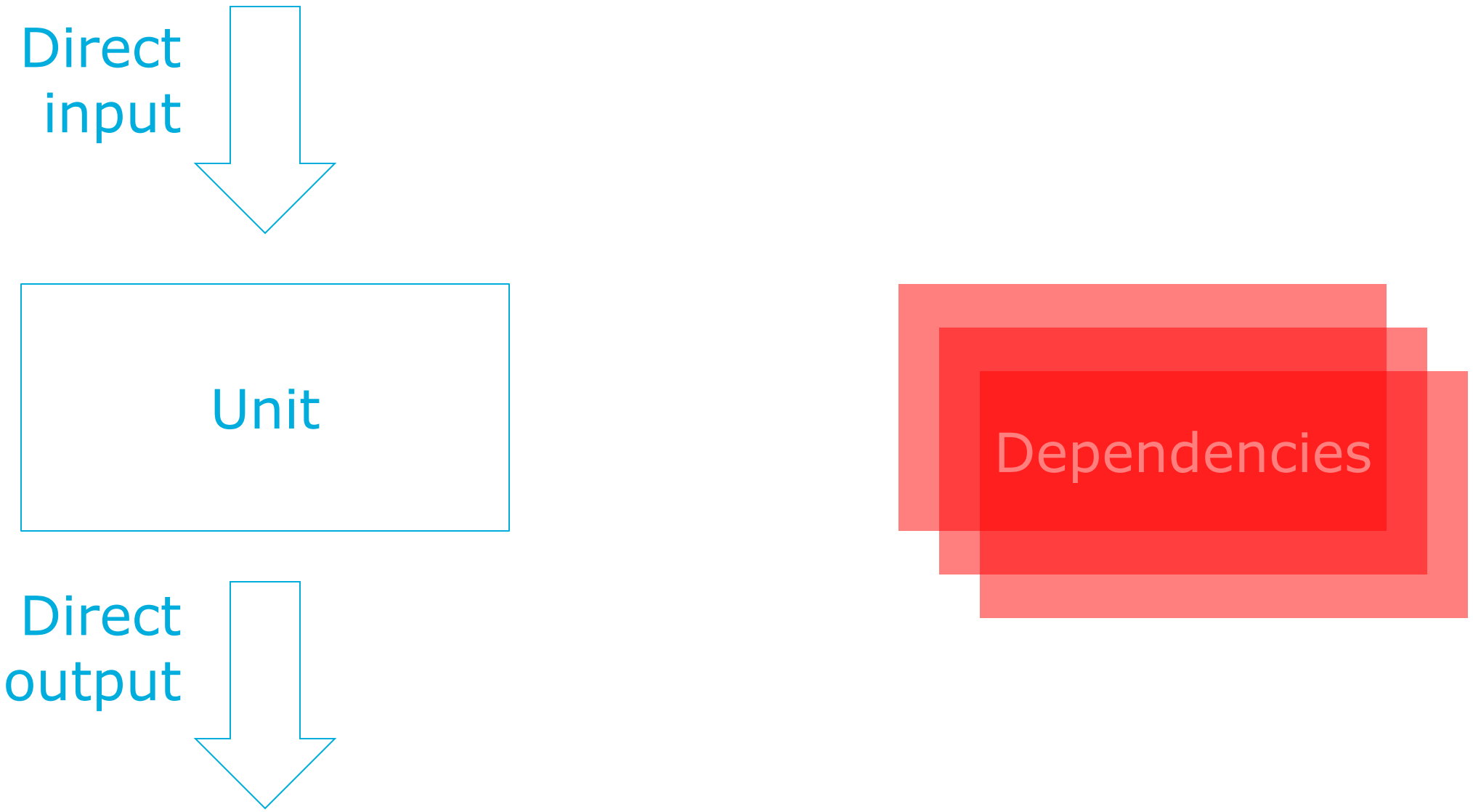
```
// Reservation -> int option
let tryAcceptComposition reservation =
    tryAccept 10 (DB.readReservations connectionString) reservation
    |> Option.map (         DB.createReservation connectionString  )
```

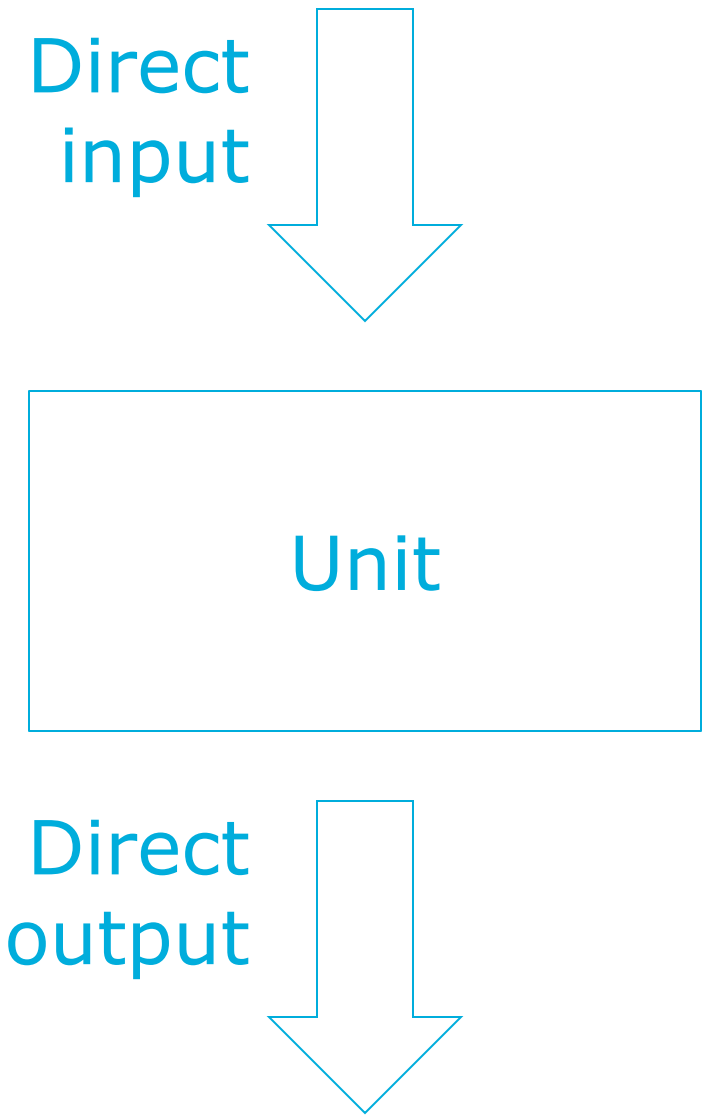
```
// Reservation -> int option
let tryAcceptComposition reservation =
    tryAccept 10 (DB.readReservations connectionString) reservation
    |> Option.map (DB.createReservation connectionString)
```

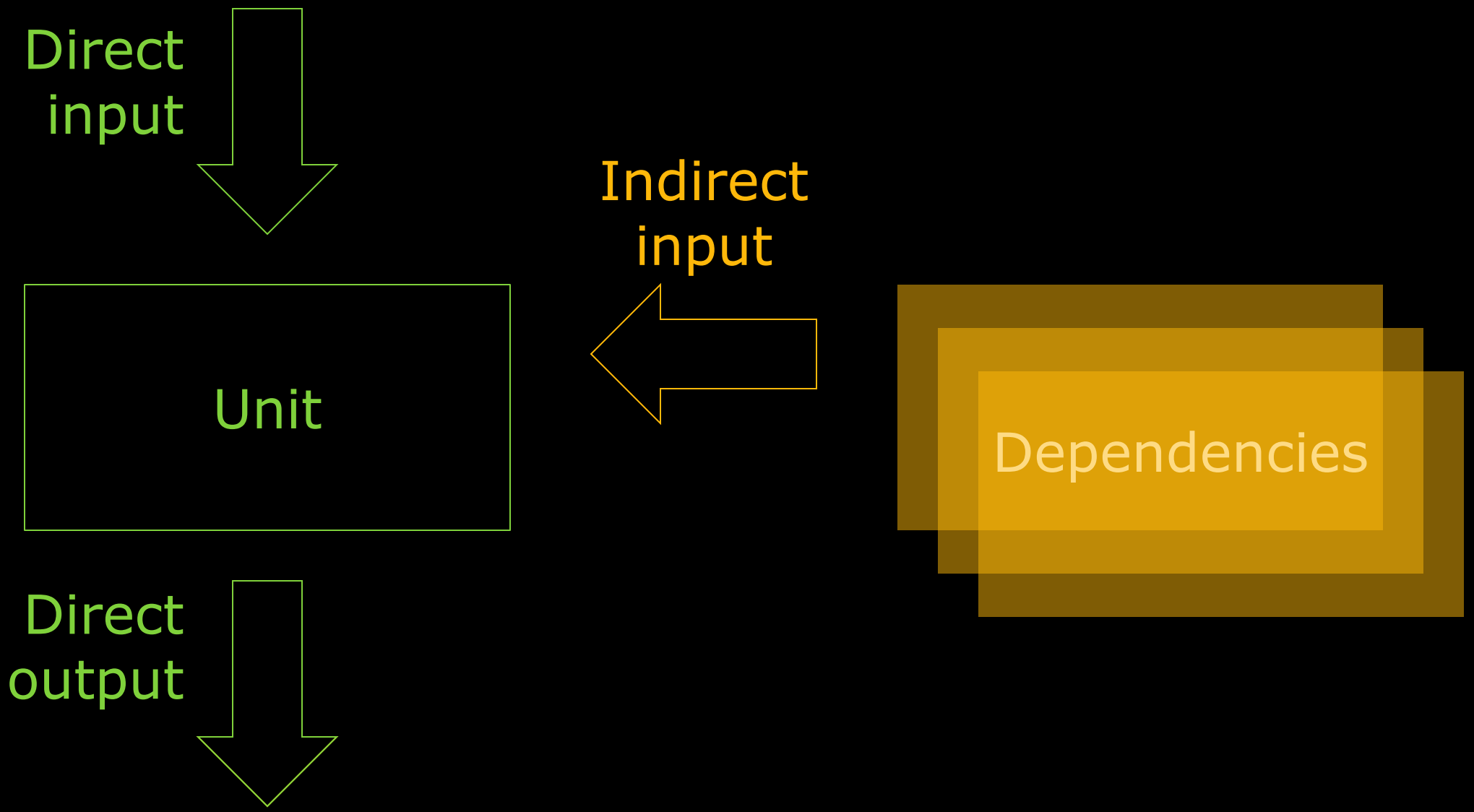
```
// Reservation -> int option
let tryAcceptComposition reservation =
    reservation
    |> tryAccept 10 (DB.readReservations connectionString)
    |> Option.map (DB.createReservation connectionString)
```

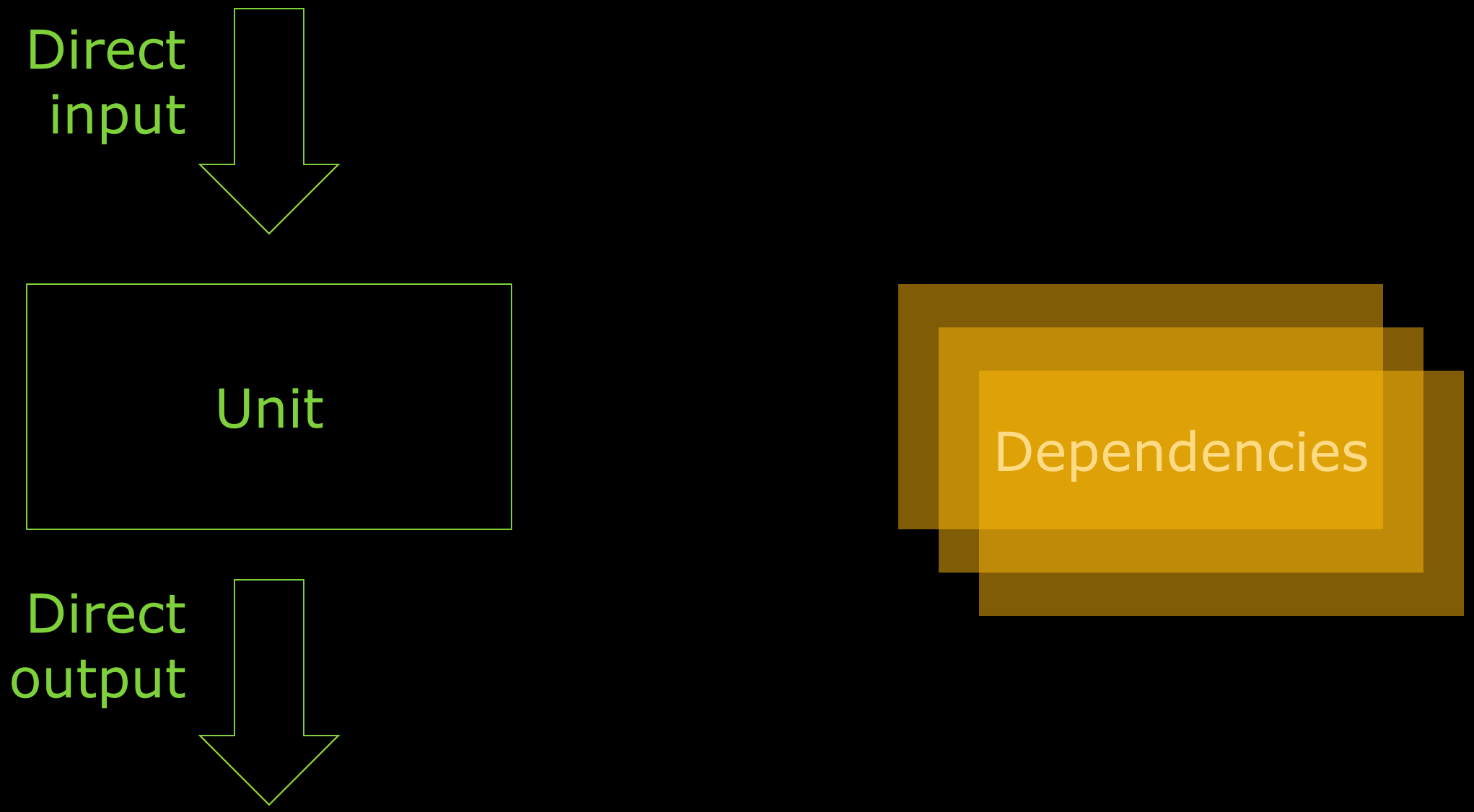
- How about testability?

- Humble object


Direct input
Unit
Direct output
Dependencies


Direct
input
Unit
Direct
output


Direct input
Indirect input
Unit
Dependencies
Direct output


Direct input
Unit
Direct output
Dependencies

```
// int -> (DateTimeOffset -> Reservation list) -> Reservation -> Reservation option
let tryAccept capacity readReservations reservation =
```

```
// int -> (                  Reservation list) -> Reservation -> Reservation option
let tryAccept capacity readReservations reservation =
```

```
// int ->                    Reservation list  -> Reservation -> Reservation option
let tryAccept capacity readReservations reservation =
```

```
// int ->                    Reservation list -> Reservation -> Reservation option
let tryAccept capacity radReservations reservation =
```

```
// int ->                  Reservation list -> Reservation -> Reservation option
let tryAccept capacity rdReservations reservation =
```

```
// int ->                  Reservation list -> Reservation -> Reservation option
let tryAccept capacity rReservations reservation =
```

```
// int ->                Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->                Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->              Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->              Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->            Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->           Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->          Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->         Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->        Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->       Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->      Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->     Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->    Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->   Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int ->  Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int -> Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
```

```
// int -> Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
    let reservedSeats =
        reservations |> List.sumBy (fun x -> x.Quantity)
    if reservedSeats + reservation.Quantity <= capacity
    then { reservation with IsAccepted = true } |> Some
    else None
```

```
// int -> Reservation list -> Reservation -> Reservation option
let tryAccept capacity reservations reservation =
    let reservedSeats = reservations |> List.sumBy (fun x -> x.Quantity)
    if reservedSeats + reservation.Quantity <= capacity
    then { reservation with IsAccepted = true } |> Some
    else None
```

```
// ('a -> 'b -> 'c) -> 'b -> 'a -> 'c
let flip f x y = f y x

// Reservation -> int option
let tryAcceptComposition reservation =
    reservation.Date
    |> DB.readReservations connectionString
    |> fun reservations -> tryAccept 10 reservations reservation
    |> Option.map (DB.createReservation connectionString)
```

```fsharp
// ('a -> 'b -> 'c) -> 'b -> 'a -> 'c
let flip f x y = f y x

// Reservation -> int option
let tryAcceptComposition reservation =
    reservation.Date
    |> DB.readReservations connectionString
    |> fun reservations -> flip (tryAccept 10) reservation reservations
    |> Option.map (DB.createReservation connectionString)
```

```fsharp
// ('a -> 'b -> 'c) -> 'b -> 'a -> 'c
let flip f x y = f y x

// Reservation -> int option
let tryAcceptComposition reservation =
    reservation.Date
    |> DB.readReservations connectionString
    |>                    -> flip (tryAccept 10) reservation
    |> Option.map (DB.createReservation connectionString)
```

```fsharp
// ('a -> 'b -> 'c) -> 'b -> 'a -> 'c
let flip f x y = f y x

// Reservation -> int option
let tryAcceptComposition reservation =
    reservation.Date
    |> DB.readReservations connectionString
    |> flip (tryAccept 10) reservation
    |> Option.map (DB.createReservation connectionString)
```

Sanity check

```haskell
tryAccept :: Int -> [Reservation] -> Reservation -> Maybe Reservation
tryAccept capacity reservations reservation =
  let reservedSeats = sum $ map quantity reservations
  in  if reservedSeats + quantity reservation <= capacity
      then Just $ reservation { isAccepted = True }
      else Nothing
```

```haskell
tryAcceptComposition :: Reservation -> IO (Maybe Int)
tryAcceptComposition reservation = runMaybeT $
  liftIO (DB.readReservations connectionString $ date reservation)
  >>= MaybeT . return . flip (tryAccept 10) reservation
  >>= liftIO . DB.createReservation connectionString
```

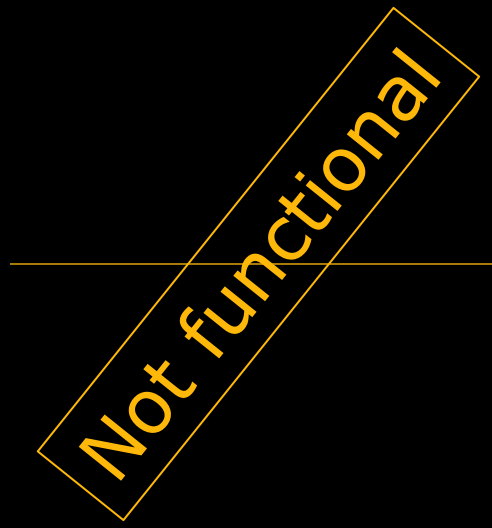
Compiles

- Object-oriented: use dependency injection
- Partial application is dependency injection
  - Not functional, though
- Functional: compose functions

Object-oriented

Functional

Dependency injection



Partial application

Composition

Mark Seemann
http://blog.ploeh.dk
http://bit.ly/ploehralsight
@ploeh