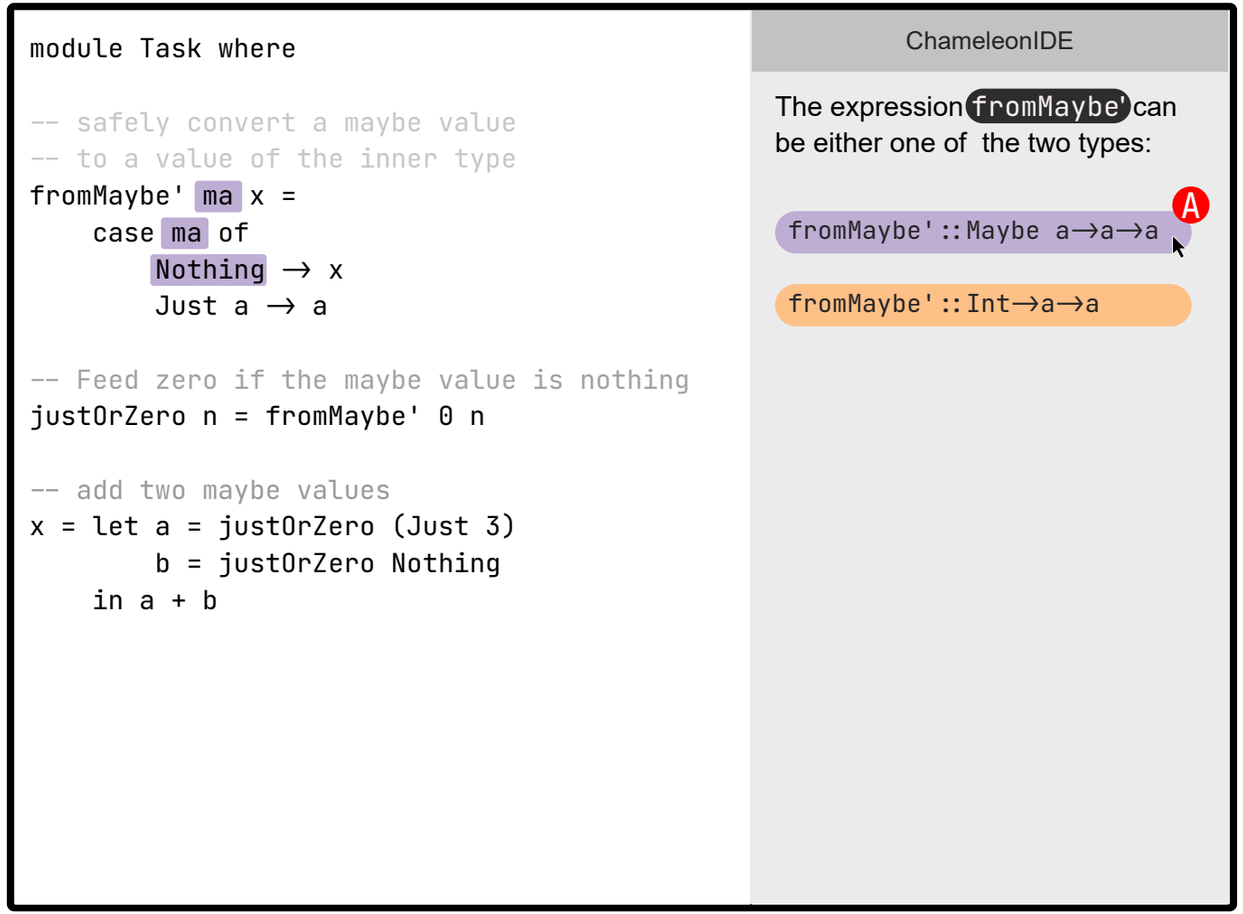
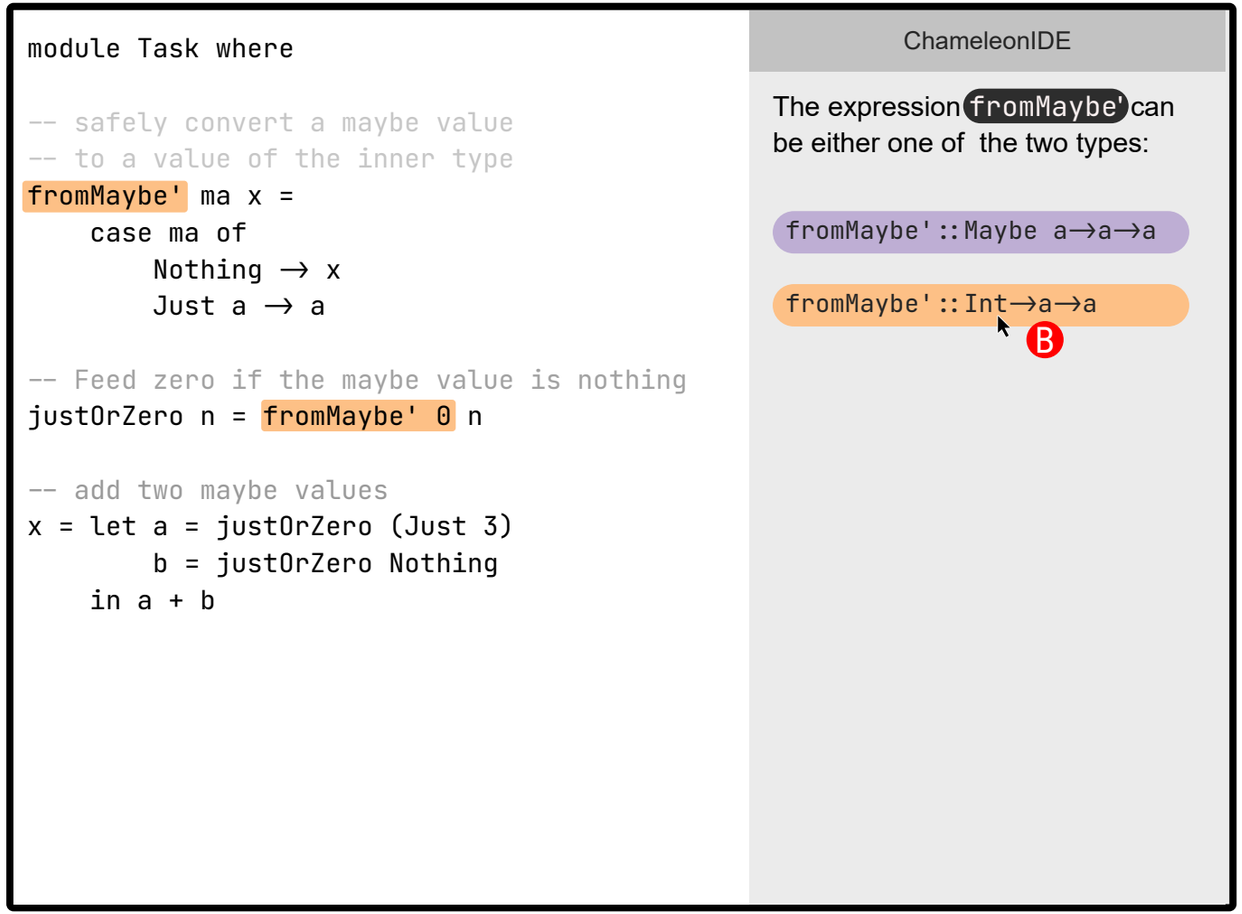

module Task where
-- safely convert a maybe value
-- to a value of the inner type
fromMaybe' ma x =
case ma of
Nothing → x
Just a → a
-- Feed zero if the maybe value is nothing
justOrZero n = fromMaybe' 0 n
-- add two maybe values
x = let a = justOrZero (Just 3)
b = justOrZero Nothing
in a + b
ChameleonIDE
The expression fromMaybe' can be either one of the two types:
fromMaybe' :: Maybe a→a→a
A
fromMaybe' :: Int→a→a


module Task where
-- safely convert a maybe value
-- to a value of the inner type
fromMaybe' ma x =
case ma of
Nothing → x
Just a → a
-- Feed zero if the maybe value is nothing
justOrZero n = fromMaybe' 0 n
-- add two maybe values
x = let a = justOrZero (Just 3)
b = justOrZero Nothing
in a + b
ChameleonIDE
The expression fromMaybe' can be either one of the two types:
fromMaybe' :: Maybe a→a→a
fromMaybe' :: Int→a→a
B