Table of Contents

ref

This package provides functionality for working with mutable references, allowin
g you to create sharable mutable values that can be modified safely.

Creating and Accessing References

References can be created using @ref.new(). The reference value can be accesse
d through the val field:

```
1
2   test "creating and accessing refs" {
3     let r1 = @ref new(42)
4     inspect(r1 val, content="42")
5   }
```

Updating Reference Values

The update function allows modifying the contained value using a transformation
function:

```
1
2   test "updating refs" {
3     let counter = @ref new(0)
4     counter update(x => x + 1)
5     inspect(counter val, content="1")
6     counter update(x => x * 2)
7     inspect(counter val, content="2")
8   }
```

Mapping References

The map function transforms a reference while preserving the reference wrapper:

```
1
2   test "mapping refs" {
3     let num = @ref new(10)
4     let doubled = num map(x => x * 2)
5     inspect(doubled val, content="20")
6     let squared = num map(x => x * x)
7     inspect(squared val, content="100")
8   }
```

Swapping Reference Values

You can exchange the values of two references using the swap function:

```
1
2   test "swapping refs" {
3     let r1 = @ref new("first")
4     let r2 = @ref new("second")
5     @ref swap(r1, r2)
6     inspect(r1 val, content="second")
7     inspect(r2 val, content="first")
8   }
```

Temporary Value Protection

The protect function temporarily sets a reference to a value and restores it aft
er executing a block:

```
1
2   test "protected updates" {
3     let state = @ref new(100)
4     let mut middle = 0
5     let result = state protect(50, () => {
6       middle = state val
7       42
8     })
9     inspect(middle, content="50")
10    inspect(result, content="42")
11    inspect(state val, content="100")
12  }
```

This is useful for temporarily modifying state that needs to be restored afterwa
rds.