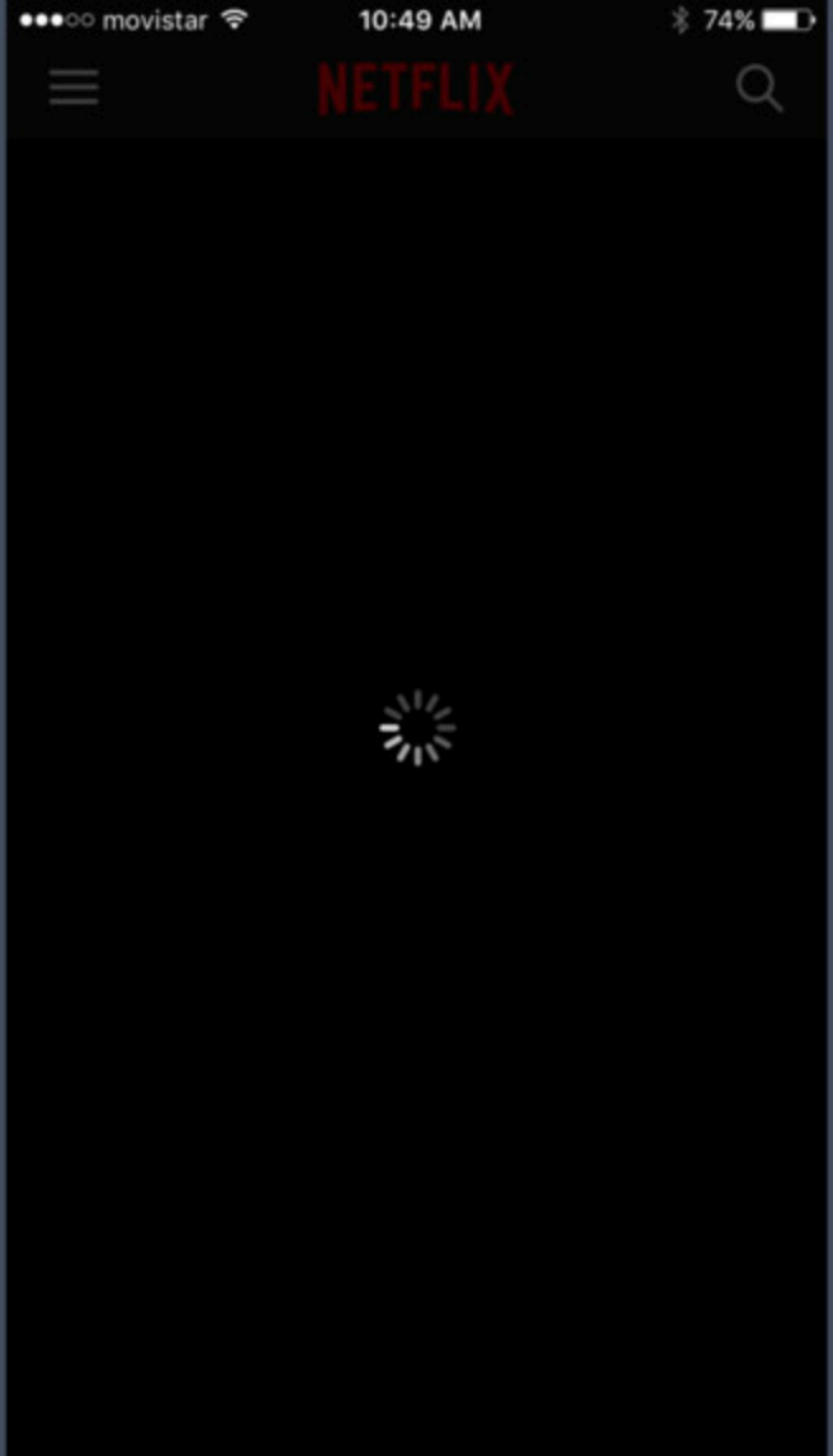
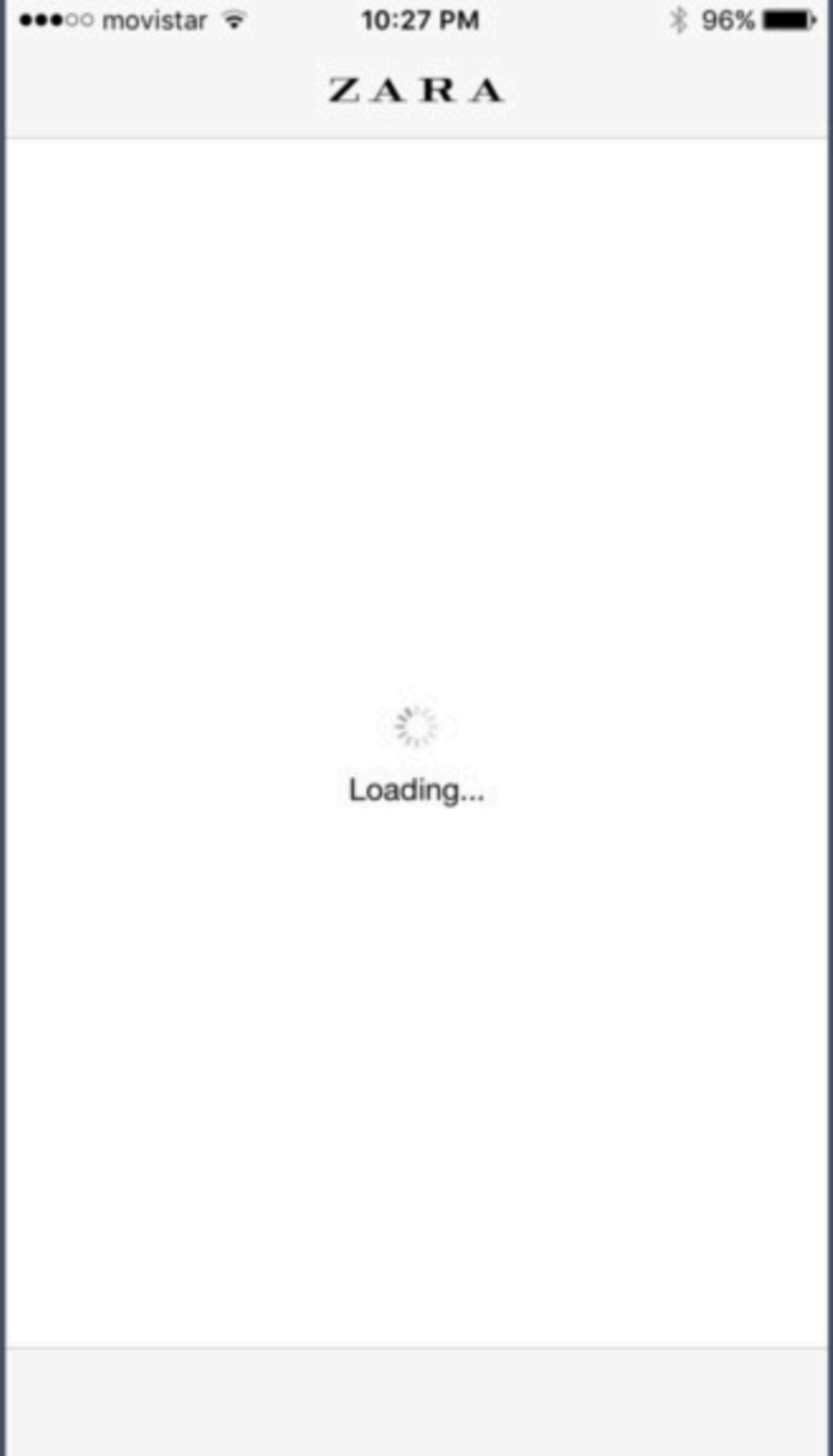
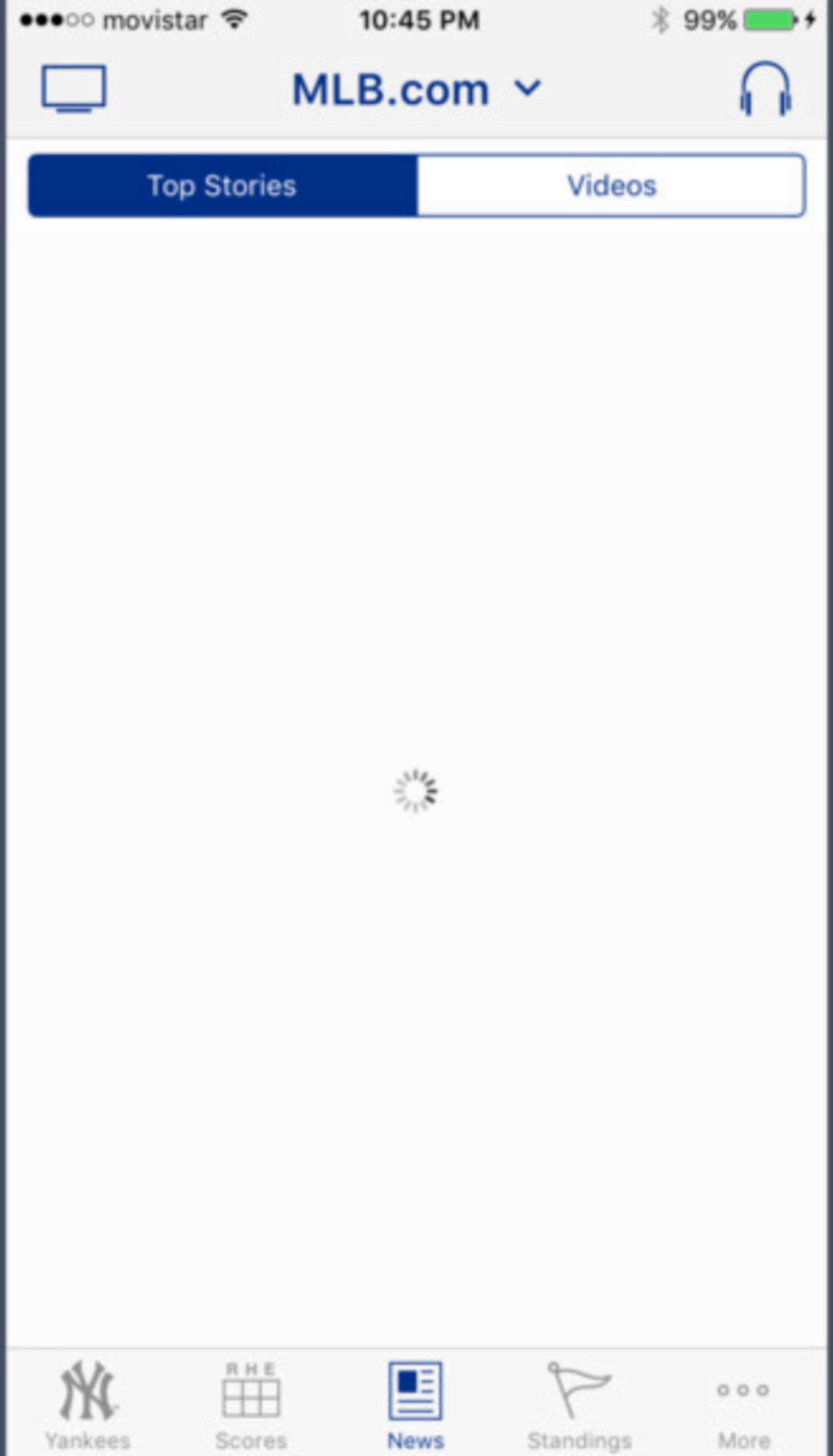
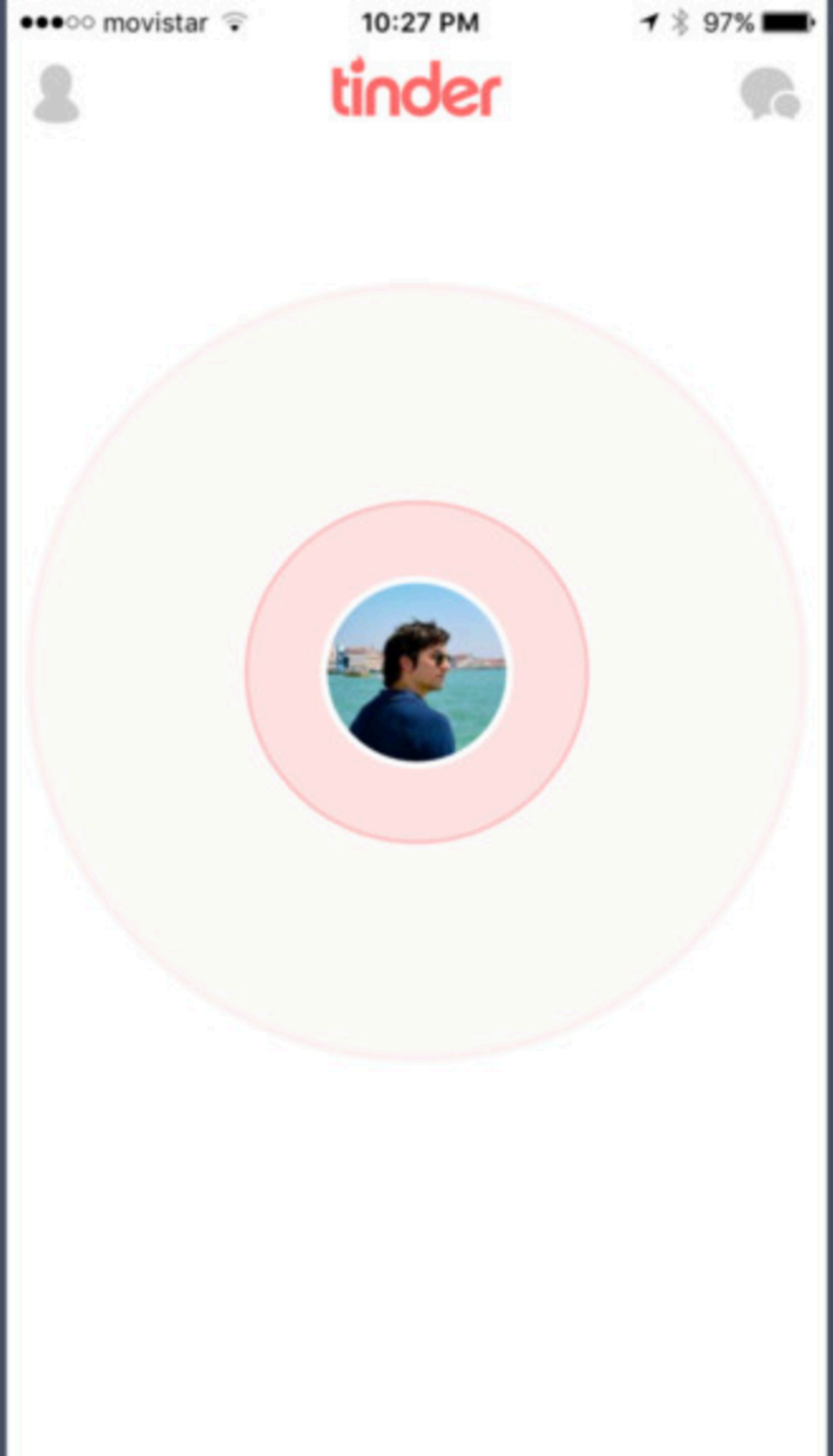
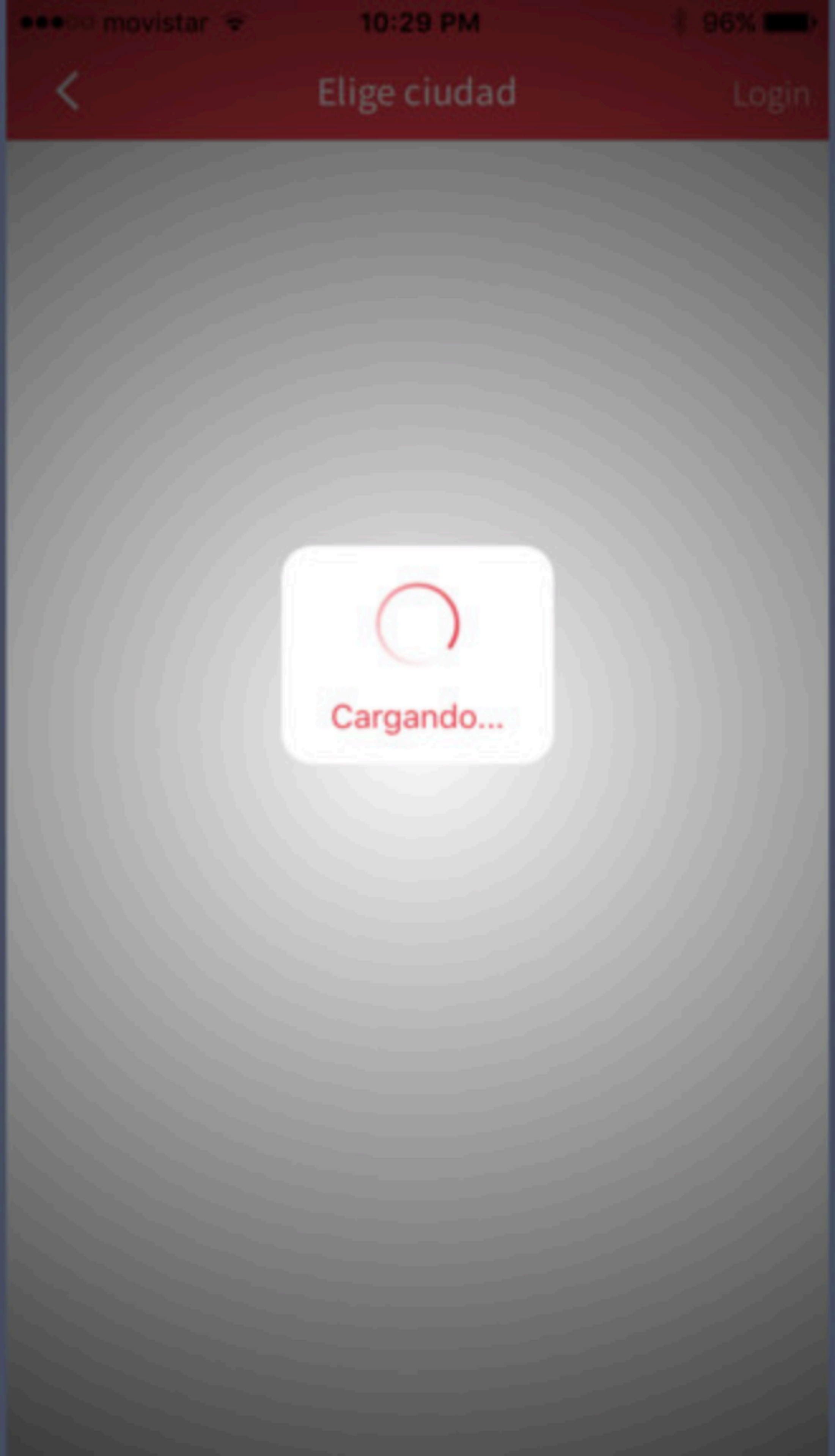
Stateful **DataSource**

Stateful `UICollectionView`
DataSource


●●●○○ movistar
10:49 AM
74%
NETFLIX

●●●○○ movistar
10:27 PM
96%
ZARA
Loading...

●●●○○ movistar
10:45 PM
99%
MLB.com
Top Stories
Videos
Yankees
Scores
News
Standings
More


●●●○○ movistar
10:27 PM
97%
tinder


●●●○○ movistar
10:29 PM
96%
Elige ciudad
Login
Cargando...

●●●○○ movistar
10:25 PM
98%
Instagram

Swift == Stateless

🤔

In Swift you can be more explicit about the state

…back in Objective-C

```
- (void)requestWithCompletionBlock:(AAPLCompletionBlock)block
{
    self.executingRequest = YES;

    [self showLoadingView];

    [self requestUsersWithPage:self.currentPagination completionBlock:^(NSArray *array, NSError *error) {
        if (error) {
            [self showError];
        } else {
            self.array = array;
            [self.tableView reloadData];
        }

        self.executingRequest = NO;

        block(array, error);
    }];
}
```

what if…

```swift
private func fetchData() {
  self.collectionView.state = .loading
  presenter.fetchLocationAndUsers { (result) in
    switch result {
    case .failure(let error):
        self.collectionView.state = .failure(error)
    case .success(let users):
        self.collectionView.state = .loaded(users)
    }
  }
}
```

However, last time I tried to do `collectionView.state` this happened:

```
Value of type 'UICollectionView' has no member 'state'
```

So what, then?

Apple provides 3 point of customizations:

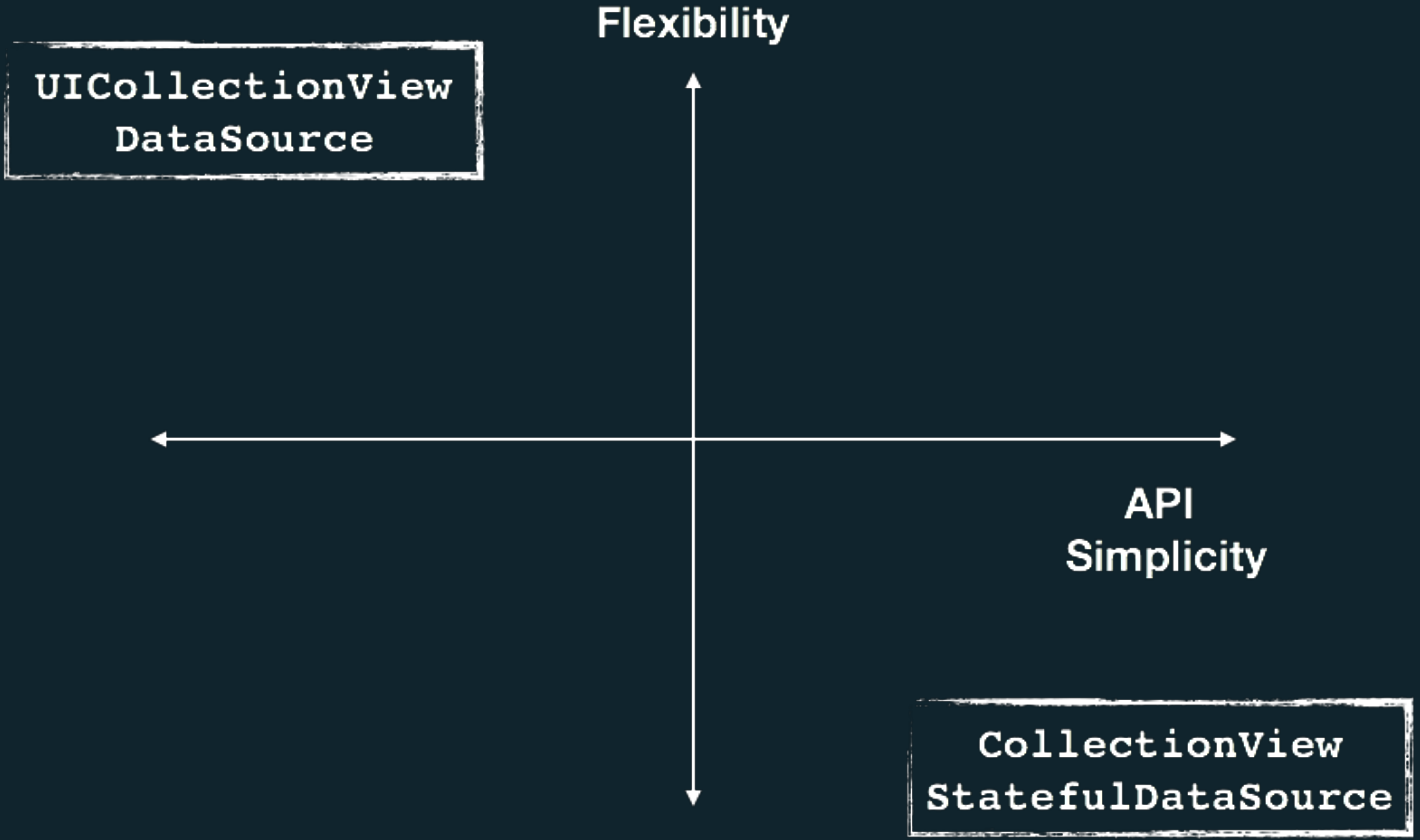
1. `Subclassing`
2. `UICollectionViewFlowLayout`
3. `UICollectionViewDataSource`

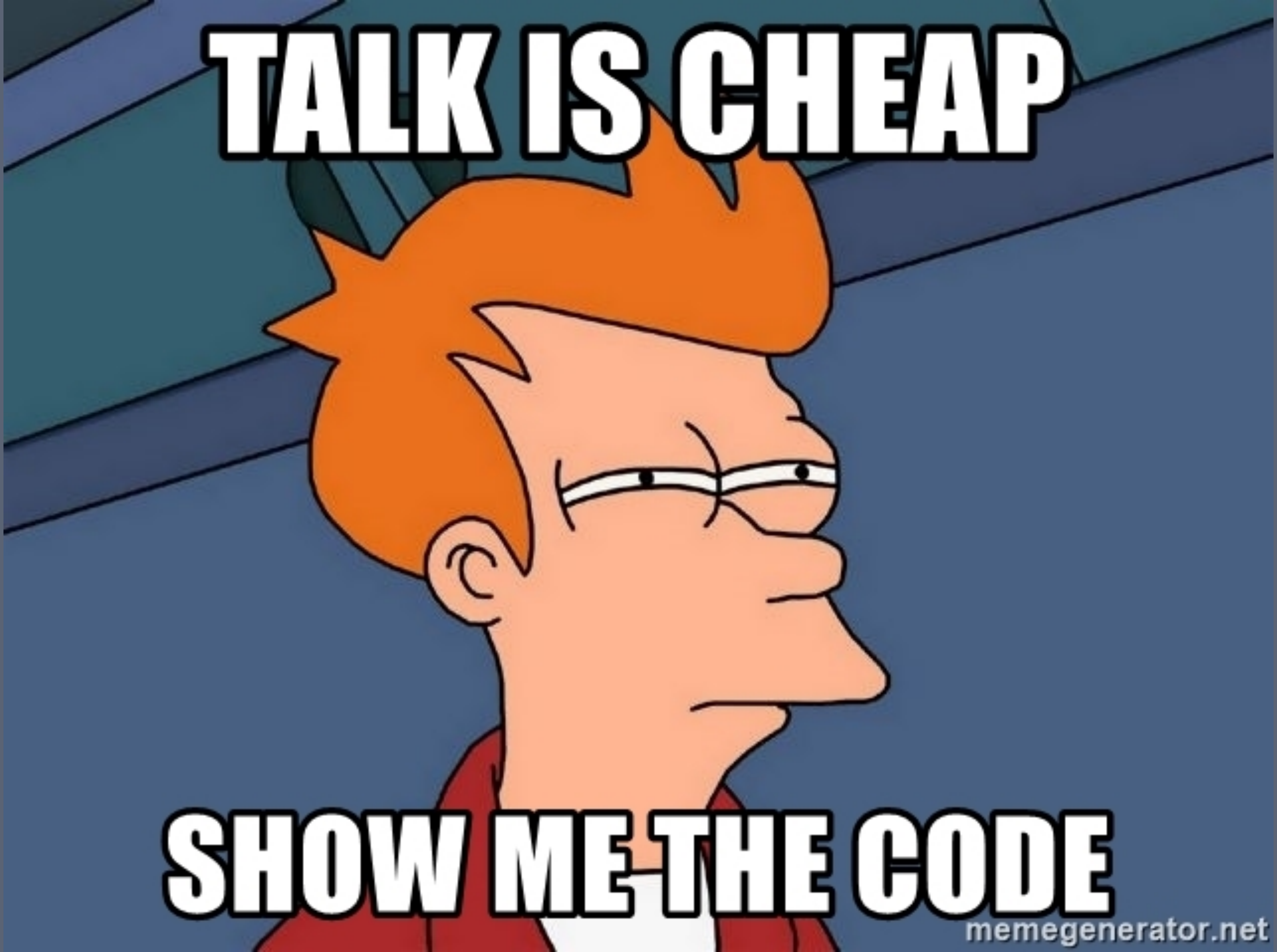
Apple provides 3 point of customizations:

1. ~~Subclassing~~
2. ~~`UICollectionViewFlowLayout`~~
3. `UICollectionViewDataSource` ✅

Objectives:

- Easier to use than `UICollectionViewDataSource`.
- Map one model object to one cell kind at **compile time**.
- Support for loading, error and empty state.


Flexibility
UICollectionView DataSource
API Simplicity
CollectionView StatefulDataSource


TALK IS CHEAP
SHOW ME THE CODE
memegenerator.net

Now, Snapshot Tests

FB Snapshot Tests

- Allow us to refactor view code with ease.
- Tests results are analized in the problem domain: Pixels
  - $f(vm, ss) = screenshot.png$
- Built by Facebook, currently maintained by Uber.

How it works:

- Set expectations
  - Create `UIView/UIViewController`
  - Inject dependencies
- Run the method
  - Layout the View
  - Generate a PNG for the View
- Compare to expectation
  - Run a diff

```swift
class ProfileViewControllerTests: BSWSnapshotTest {

    func testSampleLayout() {
        let viewModel = ProfileViewModel.sampleVM()
        let detailVC = ProfileViewController(
          viewModel: viewModel
        )
        let navController = UINavigationController(
          rootViewController: detailVC
        )
        waitABitAndVerify(viewController: navController)
    }
}
```

We Can Do It!
POST FEB. 15 TO FEB. 28
WAR PRODUCTION CO-ORDINATING COMMITTEE

Takeaways

- Be aware of the tradeoffs before writing any abstraction.
  - Bad abstractions are very, very expensive.
- Don't be afraid to refactor.
  - Cover yourself here with Tests.

Takeaways

- Use Swift's extensions to declare types.
  - Don't litter the global namespace.
  - Improves compile time and code completion.
  - Very useful with types like VM or Cells that are only used within a class.