

Testing with

Mock



in NgRx v9

Store



John Crowson



@John_Crowson



angularday 2020



John Crowson



Software Engineer



Angular & iOS



NgRx Contributor

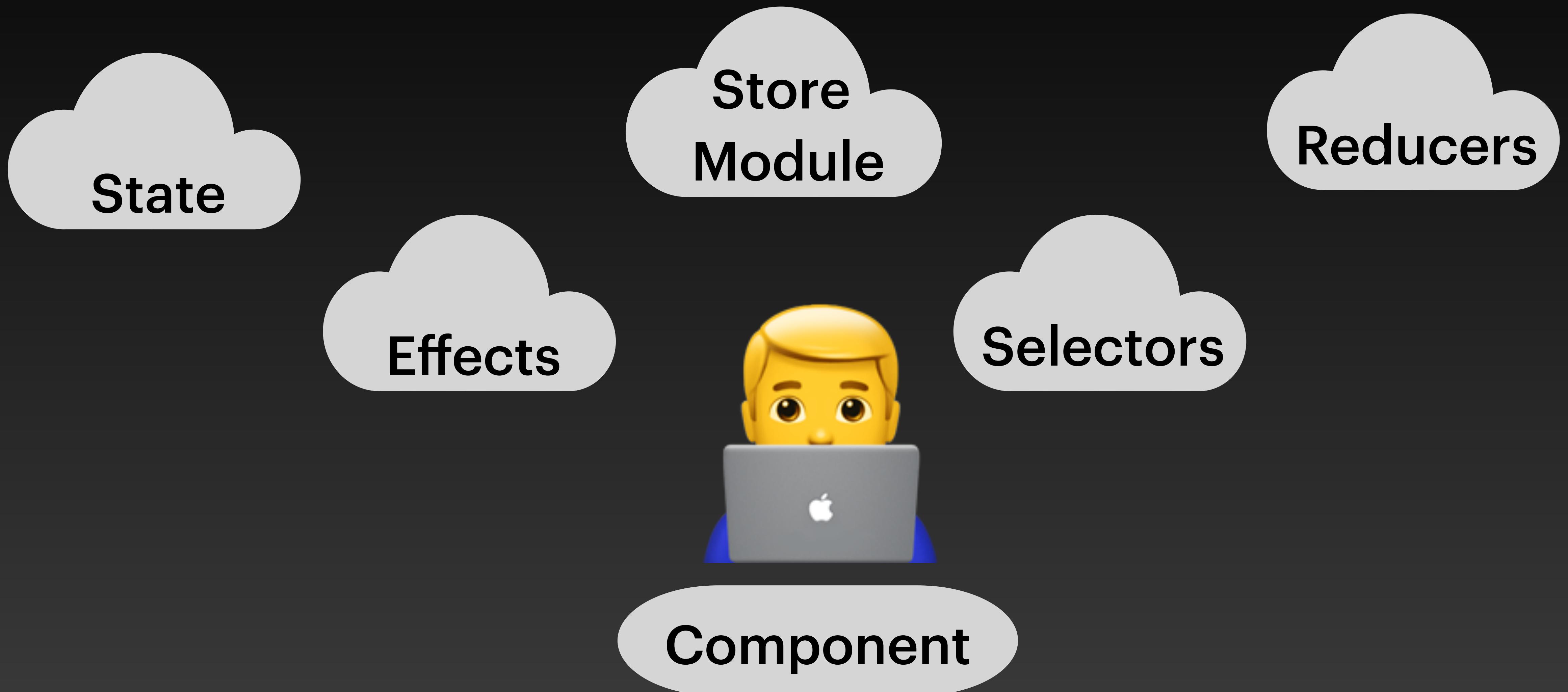


@John_Crowson



Scuba Diver

When I was learning to test with NgRx...



When I was learning to scuba dive...



7



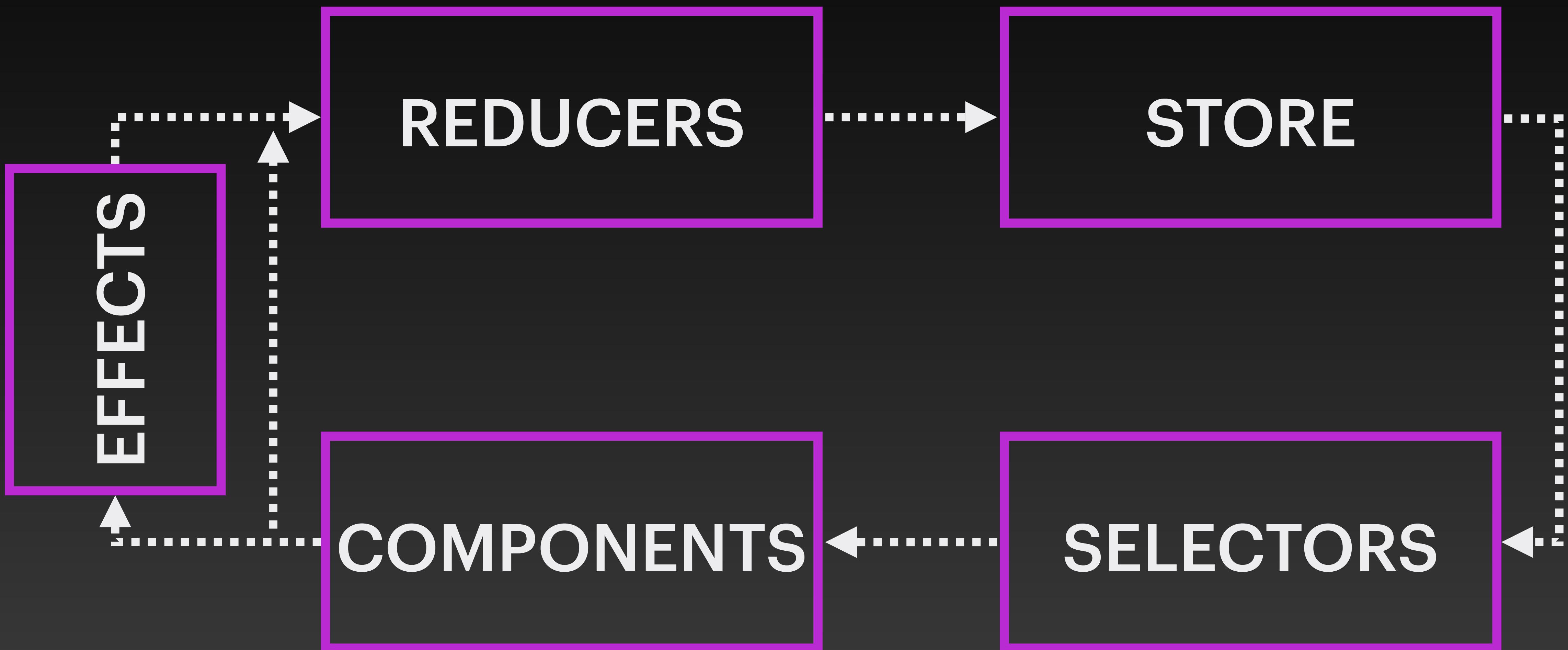
Agenda

- Testing Angular classes that inject the NgRx Store
- Unit testing in NgRx v6
- Mock Store in NgRx v7
- Mock Selectors in NgRx v8
- Examples updated for latest NgRx



```
class AngularClass {  
  constructor(ngrxStore: Store) { }  
}
```

NgRx State Data Flow



Unit Testing



Unit Testing Goal:

- Condition mock state to assert component behavior
- Validate that actions are dispatched appropriately

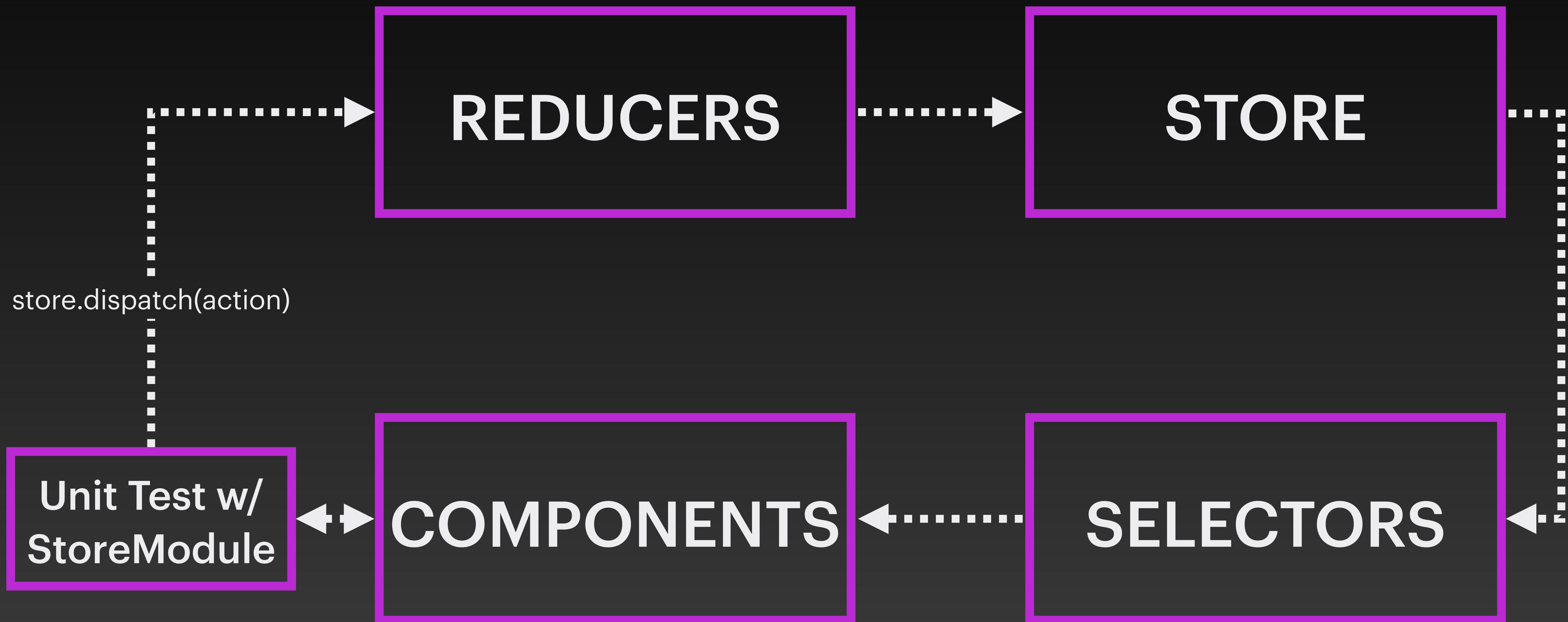
Documentation in NgRx v6:

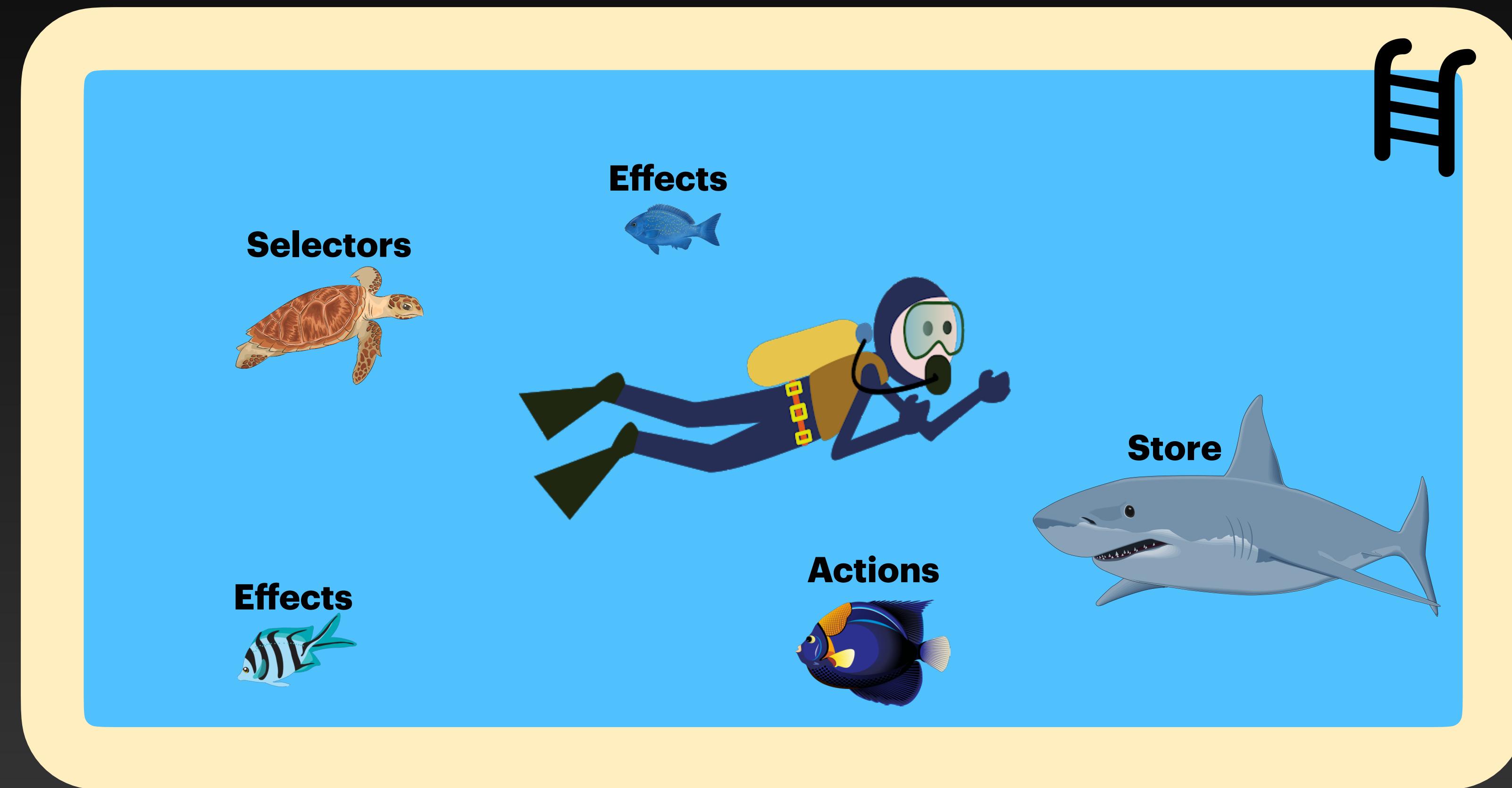
1. Import StoreModule in testing module
2. Dispatch sequence of actions to condition state

Unit Testing in NgRx v6

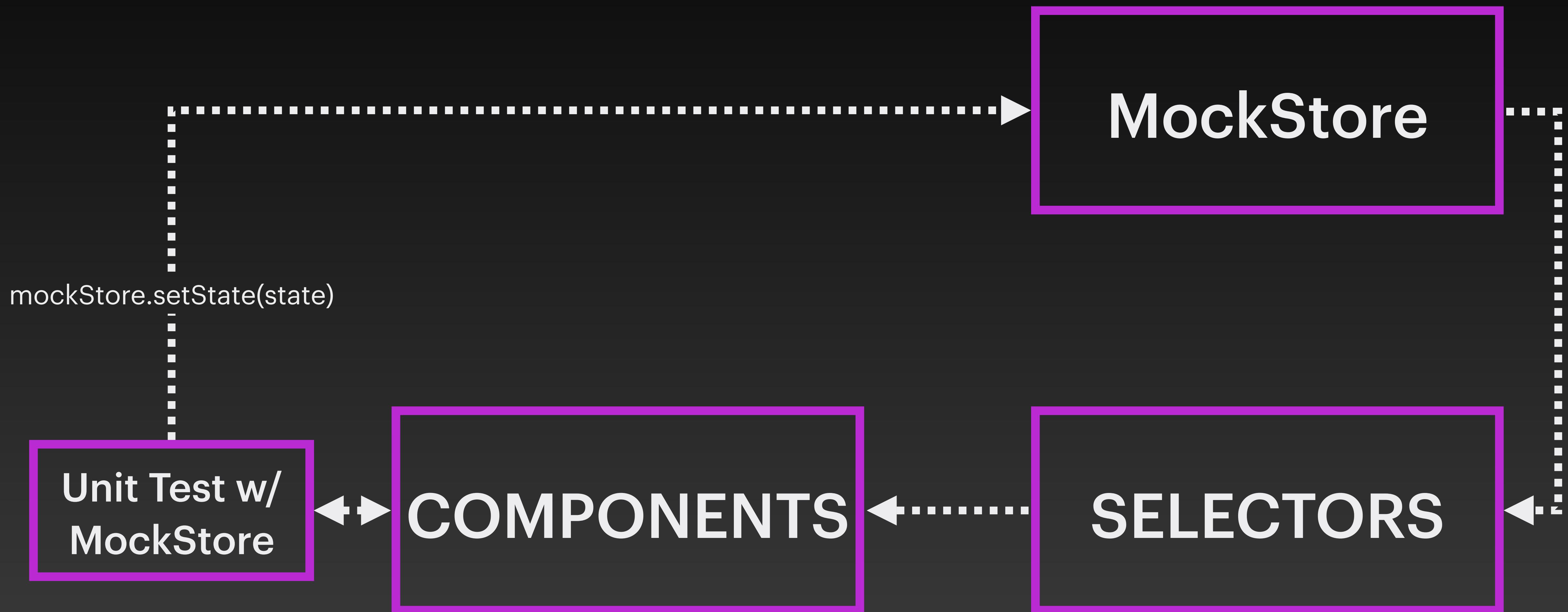
```
TestBed.configureTestingModule({  
    // ...  
    imports: [ StoreModule.forRoot({ feature: featureReducer }) ]  
});  
  
it('should display a list of all items after the data is loaded', () => {  
    store.dispatch(loadSaleItemsSuccess({ items: [1, 2, 3] }));  
    store.dispatch(loadClearanceItemsSuccess({ items: [1, 2, 3] }));  
    store.dispatch(loadNewItemsSuccess({ items: [1, 2, 3] }));  
    // ...  
  
    component.allItems$.subscribe(allItems =>  
        expect(allItems.length).toBe(6)  
    );  
});
```

Unit Testing in NgRx v6





Unit Testing in NgRx v7



```
import { MockStore } from '@ngrx/store/testing'
```

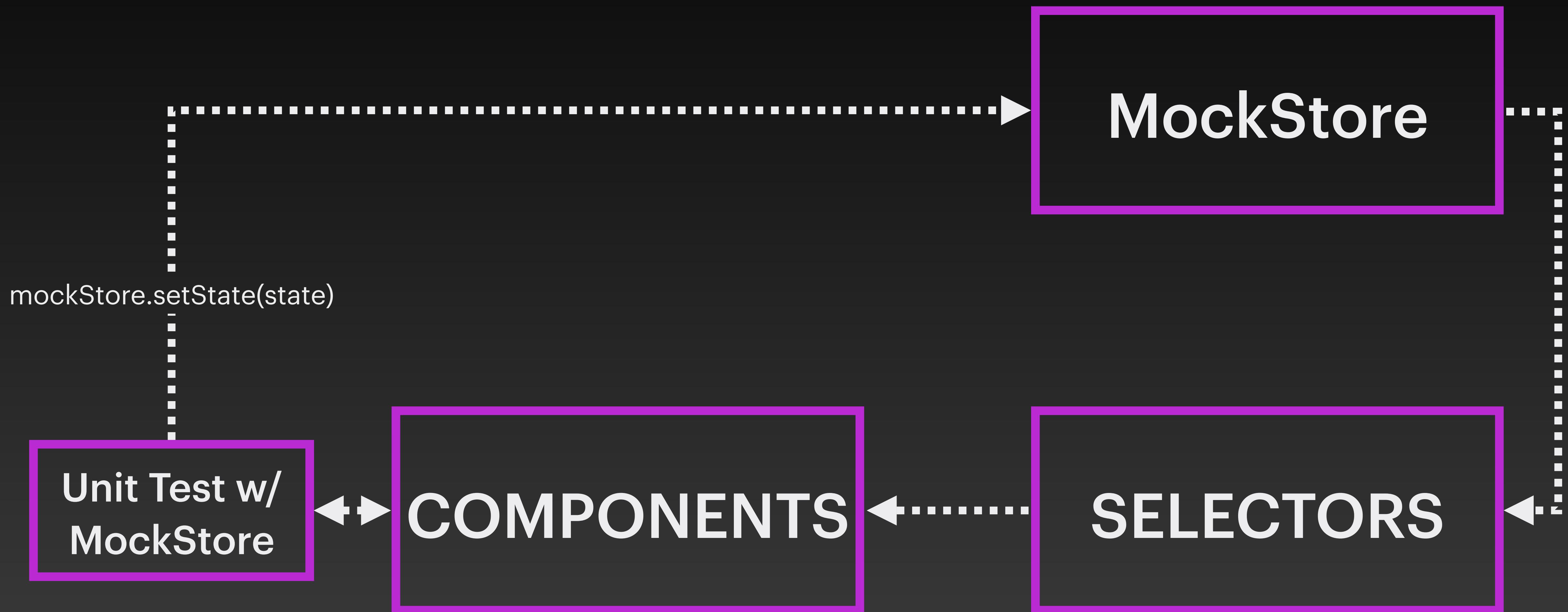
- Directly condition a given mock state to test against
- Selectors automatically trigger
- Conditioning steps:
 1. Provide provideMockStore({ initialState }) in testing module
 2. Update state with mockStore.setState(state)

Unit Testing with MockStore

```
import { provideMockStore, MockStore }  
  from '@ngrx/store/testing';  
  
let mockStore: MockStore<fromItems.State>;  
const initialState = {  
  items: {  
    saleItems: [],  
    clearanceItems: [],  
    newItems: []  
  }  
} as fromItems.State;  
  
beforeEach(() => {  
  TestBed.configureTestingModule({  
    // ...  
    providers: [  
      provideMockStore({ initialState })  
    ]  
  });  
  mockStore = TestBed.inject(MockStore);  
});
```

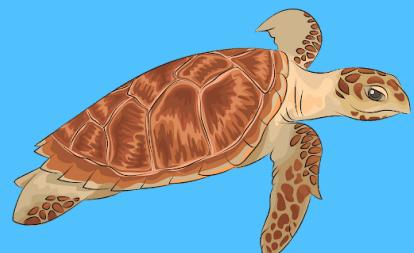
```
it('should display a list of all items'  
  + ' after the data is loaded', () => {  
  component.allItems$.subscribe(allItems =>  
    expect(allItems.length).toBe(0)  
  );  
  
  const loadedState = {  
    items: {  
      saleItems: [1, 2],  
      clearanceItems: [3, 4],  
      newItems: [5, 6]  
    }  
  };  
  
  mockStore.setState(loadedState);  
  component.allItems$.subscribe(allItems =>  
    expect(allItems.length).toBe(6)  
  );  
});
```

Unit Testing in NgRx v7

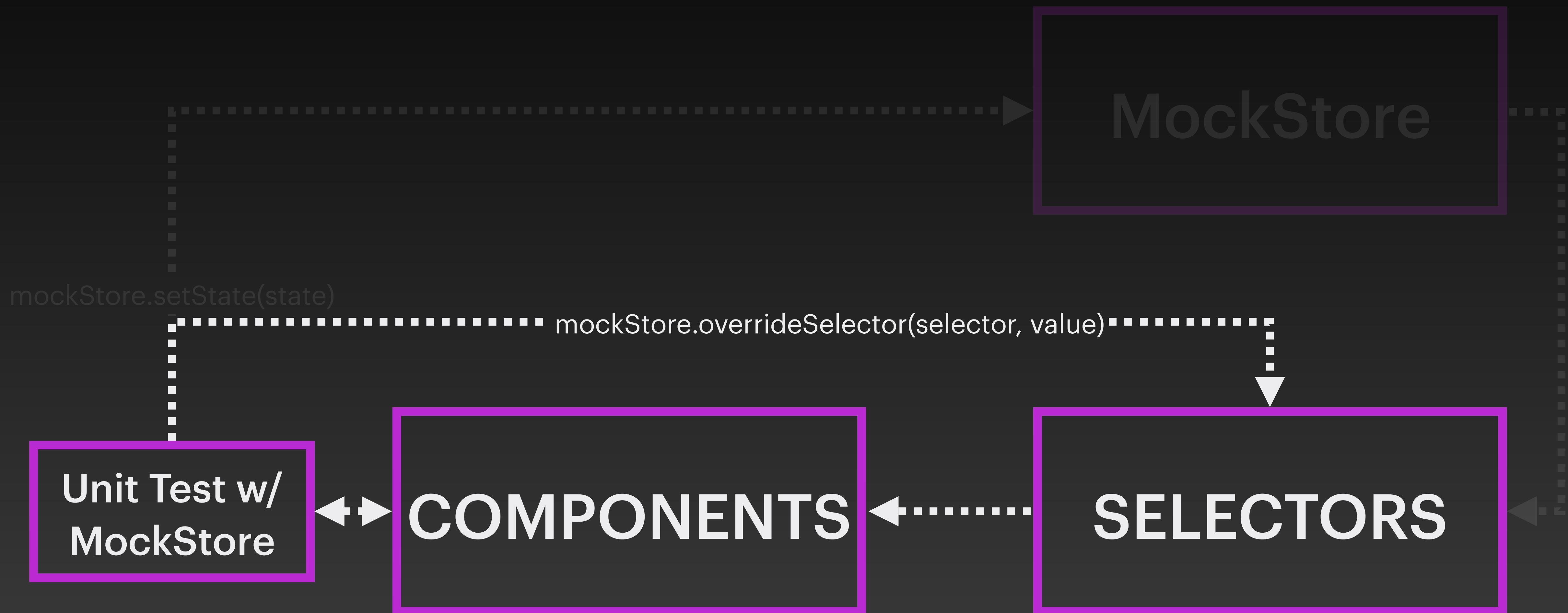


≡

Selectors



Unit Testing in NgRx v8



Mock Selectors

- Condition a mock selector without mocking the entire state
- Test complexity does not increase with selector complexity

```
const selectAllItems = createSelector(  
  selectSaleItems,  
  selectClearanceItems,  
  selectNewItems,  
  (saleItems, clearanceItems, newItems) =>  
    [...saleItems, ...clearanceItems, ...newItems]  
);
```

- Conditioning steps:
 1. Provide provideMockStore() in the testing module
 2. Mock selectors with mockStore.overrideSelector(selector, value)

Unit Testing with Mock Selectors

```
import { provideMockStore, MockStore } from '@ngrx/store/testing';
let mockStore: MockStore<fromItems.State>;
let mockSelector:
  MemoizedSelector<fromItems.State, number[]>;
beforeEach(() => {
  TestBed.configureTestingModule({
    // ...
    providers: [ provideMockStore() ]
  });
  mockStore = TestBed.inject(MockStore);
  mockSelector = mockStore.overrideSelector(
    fromItems.selectAllItems,
    []
  );
});
it('should display a list of all items' + ' after the data is loaded', () => {
  component.allItems$.subscribe(allItems =>
    expect(allItems.length).toBe(0)
  );
  mockSelector.setResult([1, 2, 3, 4, 5, 6]);
  mockStore.refreshState();
  fixture.detectChanges();
  component.allItems$.subscribe(allItems =>
    expect(allItems.length).toBe(6)
  );
});
```

Mock Selector Notes

- Mock selectors that don't need to be updated can be setup in provideMockStore:

```
providers: [ provideMockStore(  
  {  
    selectors: [  
      {  
        selector: fromItems.otherSelector,  
        value: 'value that won`t need to change'  
      }  
    ]  
  }  
) ]
```

- Mock selectors are automatically reset after each test.
 - Uses MockStore.resetSelectors() internally
- Supports all types of selector usage:

- `store.select('saleItems');`
- `store.select(fromItems.selectSaleItems);`
- `store.pipe(select('saleItems'));`
- `store.pipe(select(fromItems.selectSaleItems));`

Verifying Dispatched Actions w/ MockStore



```
// Component
fetchData() {
  store.dispatch(fetchSaleItems());
}
// Component Test
it('fetchData should trigger a fetchSaleItems action', () => {
  component.fetchData();
  const expected = cold('a', { a: fetchSaleItems() });
  expect(store.scannedActions$).toBeObservable(expected);
});
```

Why Should I Use...

StoreModule

- Integration testing

Mock Store

- Triggers state change automatically
- Grouping test case state together can be more readable

Mock Selectors

- Most isolated
- Only mock what you need
- Eliminates selector complexity

Closing Remarks

Limit the Dependency on Store

Container components only!

Use Package-Specific Testing Libraries

Like NgRx MockStore! Ever tried mocking store.pipe(select(...))? 😬

Use Schematics

```
ng generate container ComponentName --testDepth unit
```

```
ng generate container ComponentName --testDepth integration
```

Not on NgRx v7+ Yet?

Good news...

Checkout NgRx-MockStore

The MockStore library from `@ngrx/store/testing`, made available in a standalone package to accommodate testing `@ngrx/store` v5 and v6.

Installation

This library should be installed as a `devDependencies` separately from `@ngrx/store`:

```
npm install --save-dev ngrx-mockstore
```

Usage

See usage of `provideMockStore` and `MockStore` in the [NgRx store testing documentation](#). However, instead of importing from `@ngrx/store/testing`, import from `ngrx-mockstore`.

Install

```
> npm i ngrx-mockstore
```

Weekly Downloads
364 

Version 1.0.0 **License** MIT

Unpacked Size 82.7 kB **Total Files** 29

Issues 1 **Pull Requests** 6

Start Small: Scuba Dive in a Pool

Mock Store > StoreModule

References

- <https://ngrx.io/guide/store/testing>
- <https://ngrx.io/guide/schematics/container>
- <https://JohnCrowson.com/assets/angularday2020.pdf>
- Questions?
 - @John_Crowson

Thank You!

