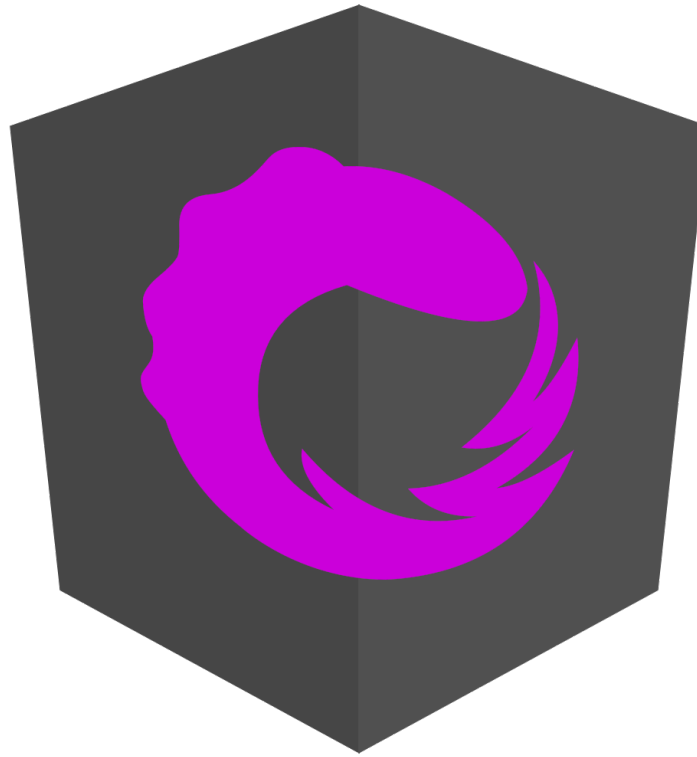


Testing with

*Mock*



in NgRx v8

Store



John Crowson



@john\_crowson



# Agenda

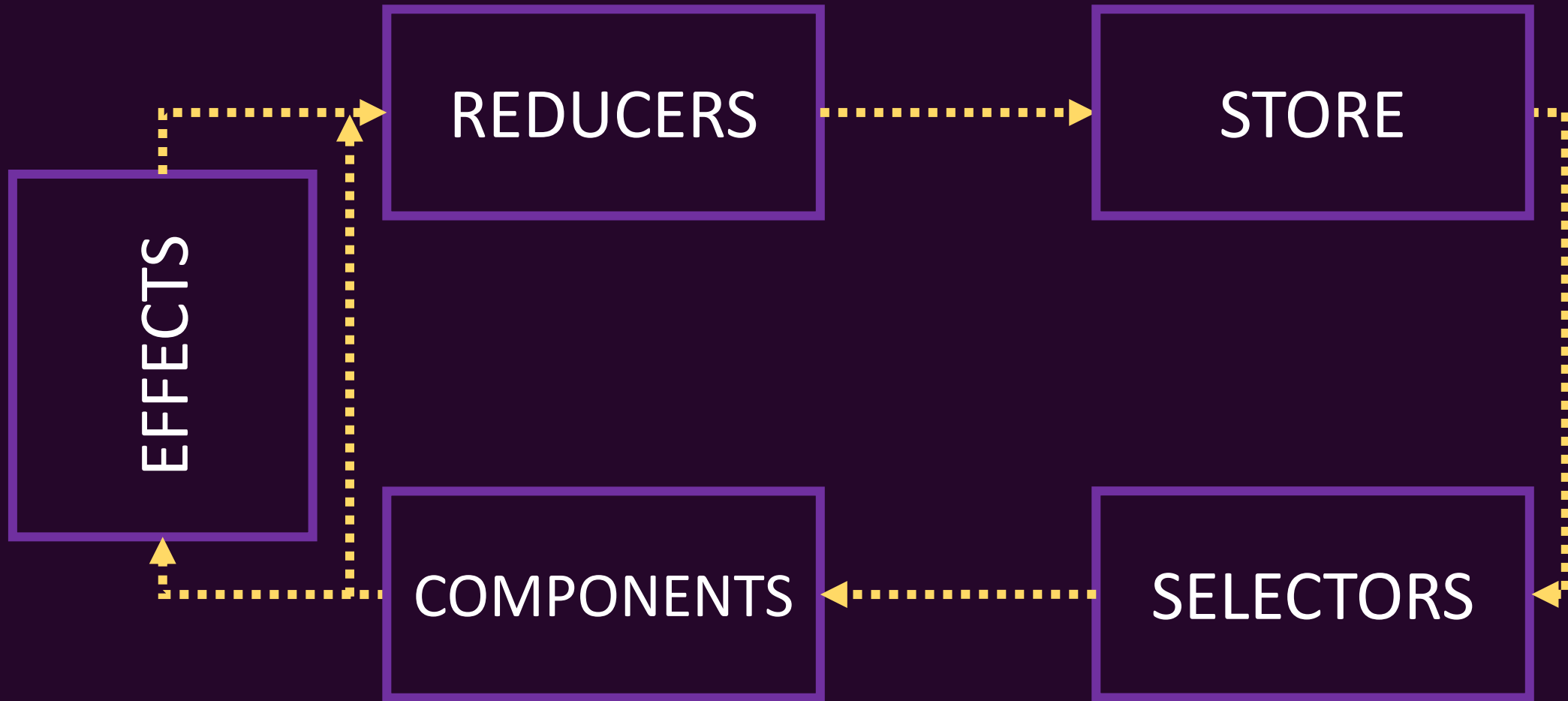
- Testing Angular classes that inject the NgRx store

```
export class AngularClass {  
  constructor(private store: Store<fromAuth.State>) {}  
}
```

- Component Unit Testing in NgRx v6
- Improved Unit Testing with MockStore (v7)  
& Mock Selectors (v8)



# NgRx Data Flow



# Unit Testing



## Unit Testing Goal:

Condition **mock state** to assert component behavior

## Documentation in NgRx v6:

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



# Unit Testing in v6

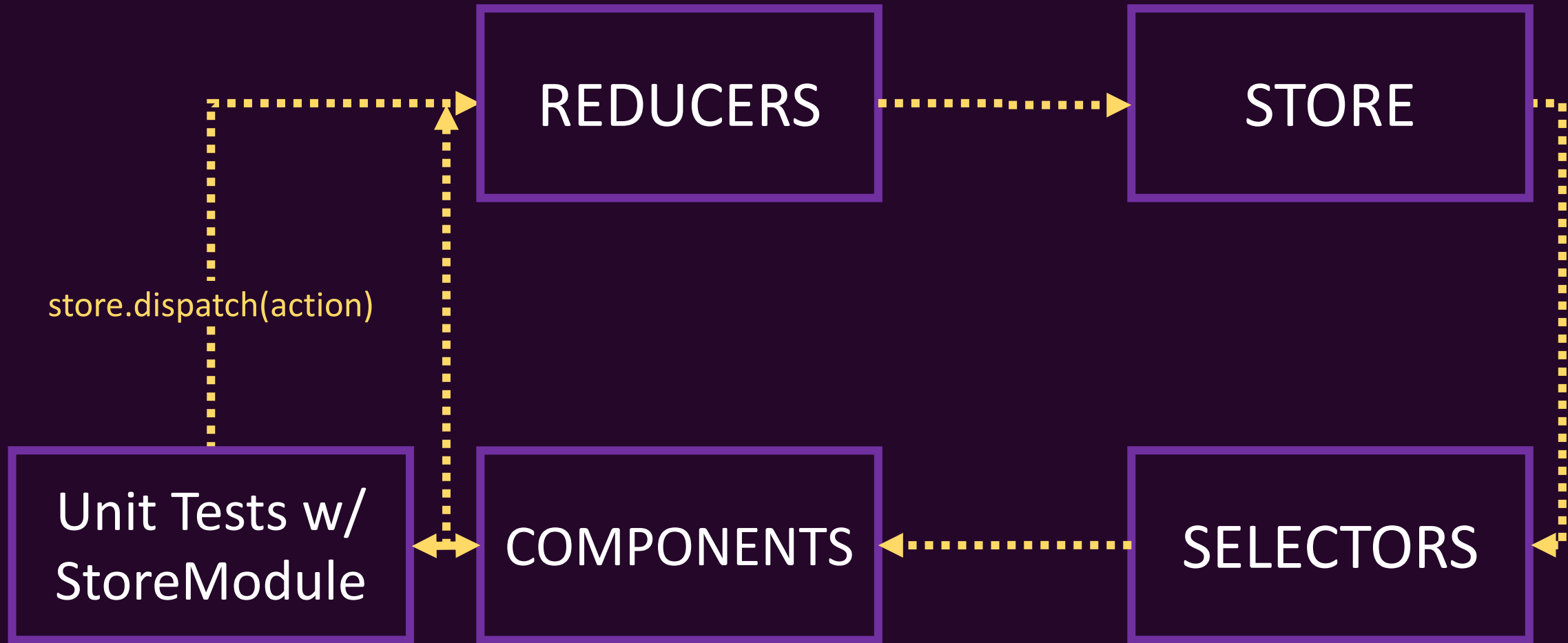
```
TestBed.configureTestingModule({
  // ...
  imports: [ StoreModule.forRoot({ feature: featureReducer }) ]
  // ...
});

it('should display a list of all items after the data is loaded', () => {
  store.dispatch(new LoadSaleItemsSuccess({ items: [1, 2] }));
  store.dispatch(new LoadClearanceItemsSuccess({ items: [3, 4] }));
  store.dispatch(new LoadNewItemSuccess({ items: [5, 6] }));
  // ...

  component.allItems$.subscribe(allItems =>
    expect(allItems.length).toBe(6)
  );
});
```



# Unit Testing in v6



# Unit Testing in v6

**Wanted:**

Simple way to mock the NgRx Store



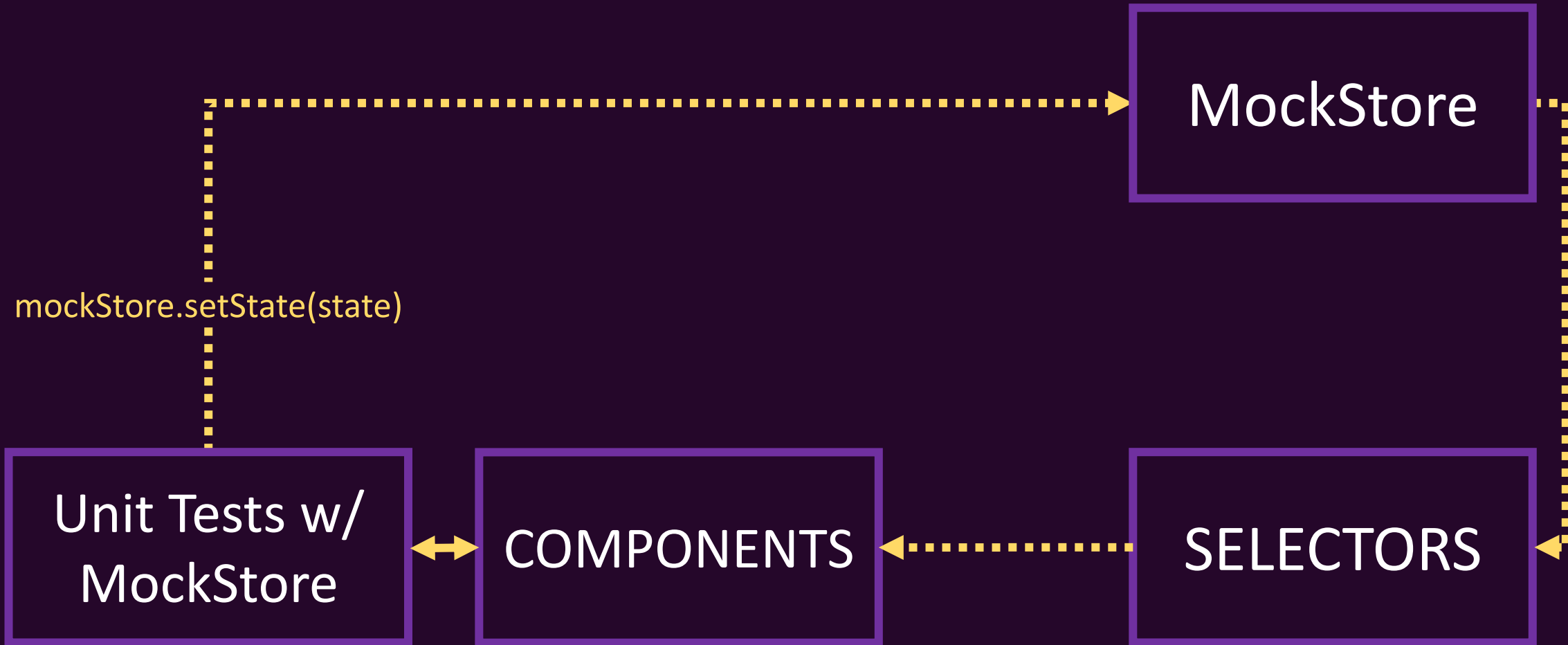
@john\_crowson

```
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



# 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.get(Store);
  // ...
});
```

```
it('should display a list of all items'
  + ' after data is loaded', () => {

  component.allItems$.subscribe(allItems =>
    expect(allItems.length).toBe(0)
  );

  // ...
  const loadedState = {
    items: {
      saleItems: [1, 2],
      clearanceItems: [3, 4],
      newItems: [5, 6]
    }
  } as fromItems.State;

  mockStore.setState(loadedState);

  component.allItems$.subscribe(allItems =>
    expect(allItems.length).toBe(6)
  );
});
```

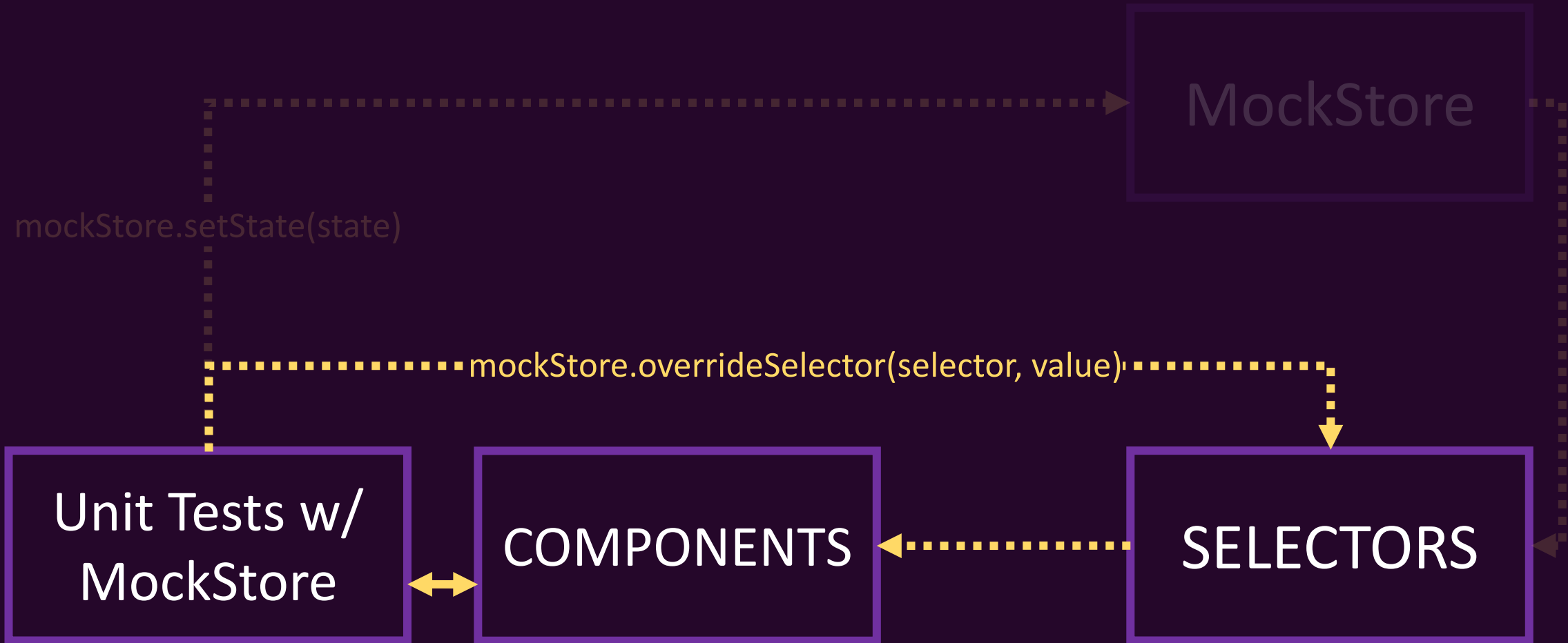
# Mock Selectors

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

```
export const selectAllItems = createSelector(
  selectSaleItems,
  selectClearanceItems,
  selectNewItem,
  (saleItems, clearanceItems, newItem) =>
    [...saleItems, ...clearanceItems, ...newItem]
);
```

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

# Unit Testing with Mock Selectors



# Unit Testing with Mock Selectors

```
import { provideMockStore, MockStore }
  from '@ngrx/store/testing';
// ...
let mockStore: MockStore<fromItems.State>;

beforeEach(() => {
  TestBed.configureTestingModule({
    // ...
    providers: [
      provideMockStore()
    ]
  });

  mockStore = TestBed.get(Store);
  mockStore.overrideSelector(
    fromItems.selectAllItems,
    []
  );
  // ...
});
```

```
it('should display a list of all items'
  + ' after data is loaded', () => {

  component.allItems$.subscribe(allItems =>
    expect(allItems.length).toBe(0)
  );

  // ...
  mockStore.overrideSelector(
    fromItems.selectAllItems,
    [1, 2, 3, 4, 5, 6]
  );

  triggerStateChange()

  component.allItems$.subscribe(allItems =>
    expect(allItems.length).toBe(6)
  );
});
```

# Why Should I Use...

## StoreModule

- Integration testing

## MockStore

- 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



# References & Thanks

- <https://ngrx.io/guide/store/testing>
- <https://ngrx.io/guide/schematics/container>
- <https://JohnCrowson.com/assets/angular-denver.pdf>
- NgRx Team
- Questions?
  - @john\_crowson



# Appendix



# Selector Support

```
this.store.select('orders');
```

```
this.store.select(fromOrders.getOrders);
```

```
this.store.pipe(select('orders'));
```

```
this.store.pipe(select(fromOrders.getOrders));
```

