- ngrx-data: no longer have to write actions/reducers/selectors again…

- Reducers
  - Should always be pure. Input = Output

- Selectors
  - Should always be pure.
  - Can use Snapshot testing.

- Actions

- Side Effects
  - Can use Jasmine Marbles to verify input actions

- Components
  - Given set of inputs = same outputs
  - Has no global observable dependencies: (modifying global vars, global state, rendering other side effect components)
    1. create component fixture
    2. beforeEach: async() + await fixture.compile({book});
    3. Should Compile: expect(fixture).toMatchSnapshot();

- Container Tests
  - Store aware components
  - Verify they work correctly
  - Verify they dispatch the right actions

NgRx Entity

- Provides set of the functions for CRUD operations

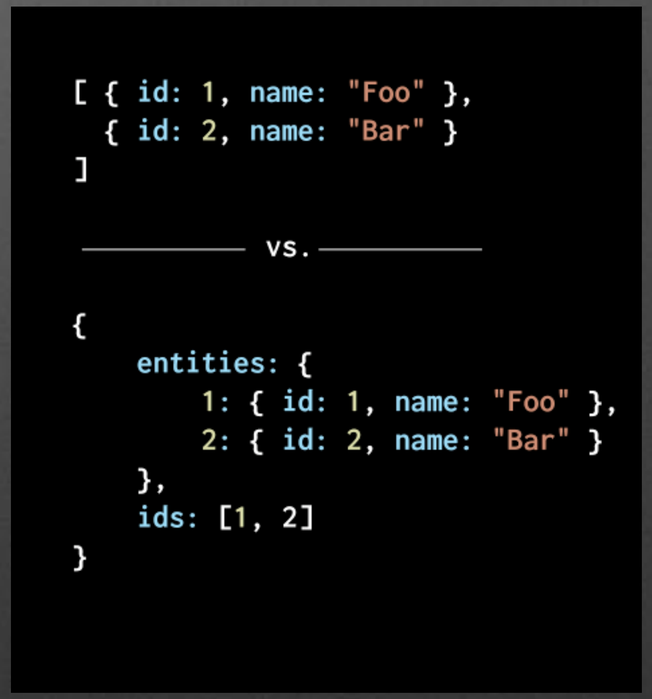
- @ngrx/entity lets you create entity adapters for different kinds of entities. Using an entity adapter, you can quickly write reducer operations and automatically generate selectors.

```
interface Book {
  id: string;
  title: string;
}

import { createEntityAdapter } from '@ngrx/entity';

const bookAdapter = createEntityAdapter<Book>();

export const {
selectIds,  selectEntities,  selectAll,  selectTotal,
} = bookAdapter.getSelectors();
```



NgRx

- ngrx-data: no longer have to write actions/reducers/selectors again…
- Reducers
  - Should always be pure. Input = Output
- Selectors
  - Should always be pure.
  - Can use Snapshot testing.
- Actions
- Side Effects
  - Can use Jasmine Marbles to verify input actions
- Components
  - Given set of inputs = same outputs
  - Has no global observable dependencies: (modifying global vars, global state, rendering other side effect components)
    1. create component fixture
    2. beforeEach: async() + await fixture.compile({book});
    3. Should Compile: expect(fixture).toMatchSnapshot();
- Container Tests
  - Store aware components
  - Verify they work correctly
  - Verify they dispatch the right actions