Intro to @ngrx

What is @ngrx?

A framework for building **reactive applications** in Angular

What is @ngrx/store?

A controlled **state container**

Reactive Application

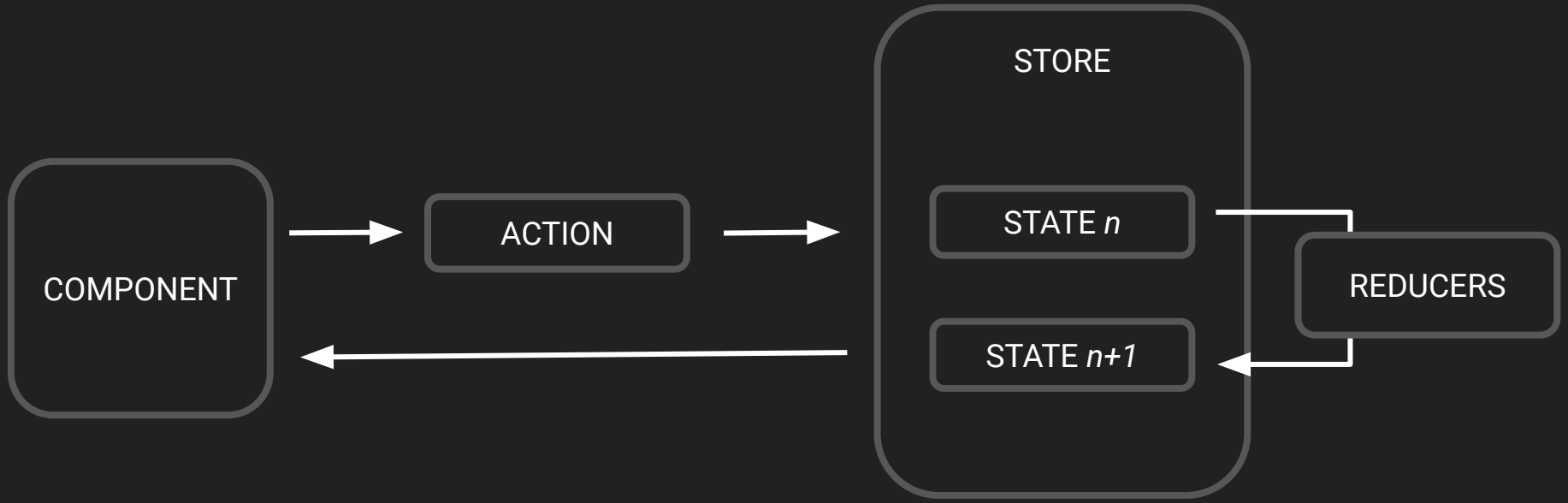
Relies on asynchronous message-passing. Components react to changes instead of asking for them.

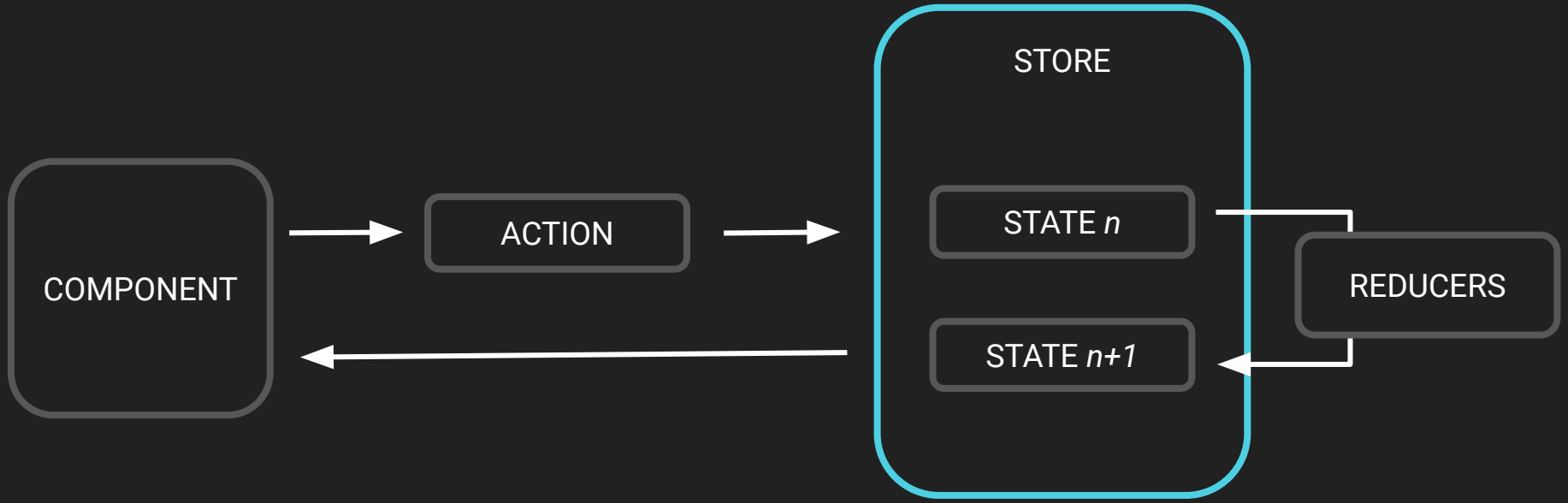
State Container

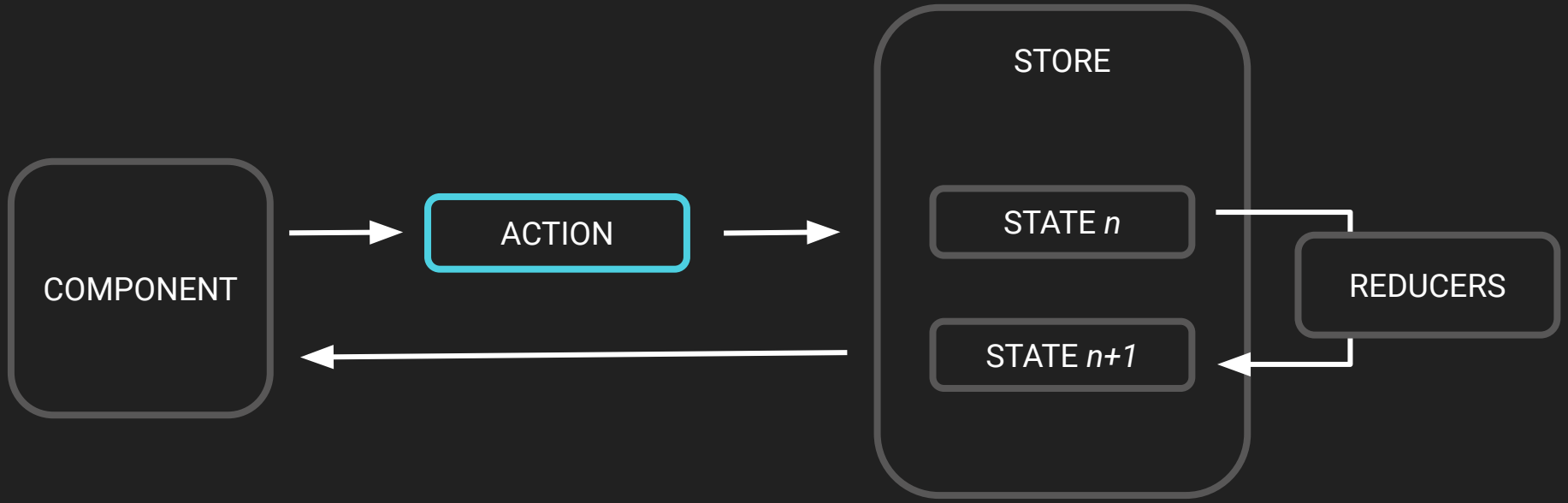
Has a finite number of states, but can only be in one state at a time. It transitions between states based on external inputs.

Reactive State

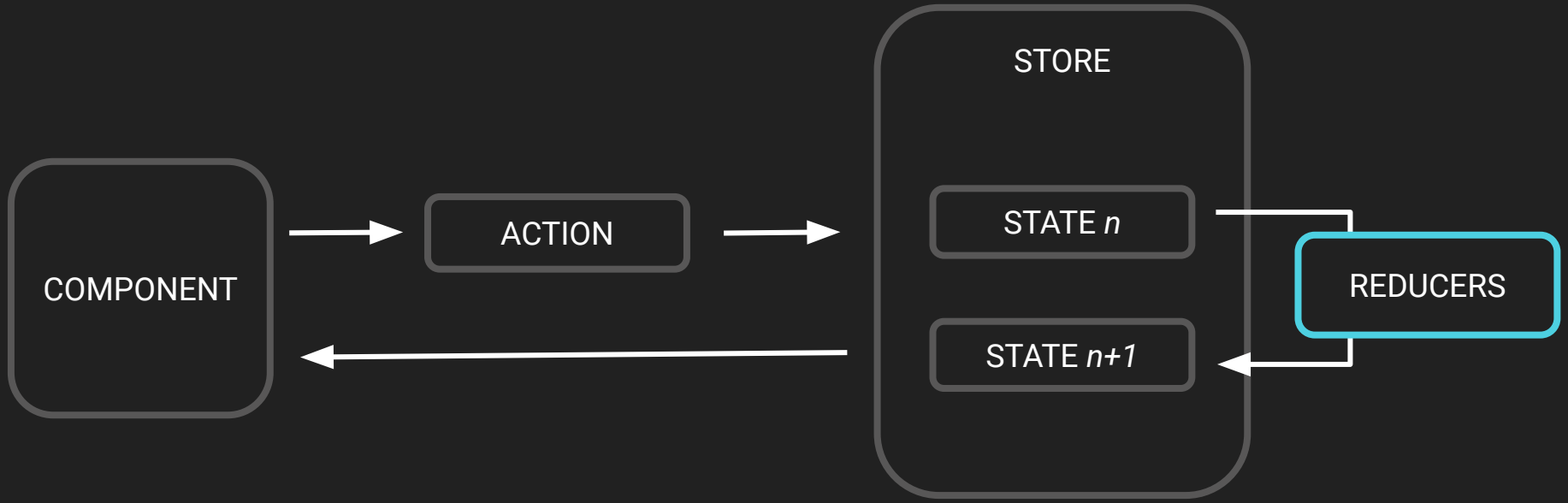
- State is immutable
- Messages can cause a transition between states


COMPONENT
ACTION
STORE
STATE n
STATE n+1
REDUCERS


COMPONENT
ACTION
STORE
STATE n
STATE n+1
REDUCERS

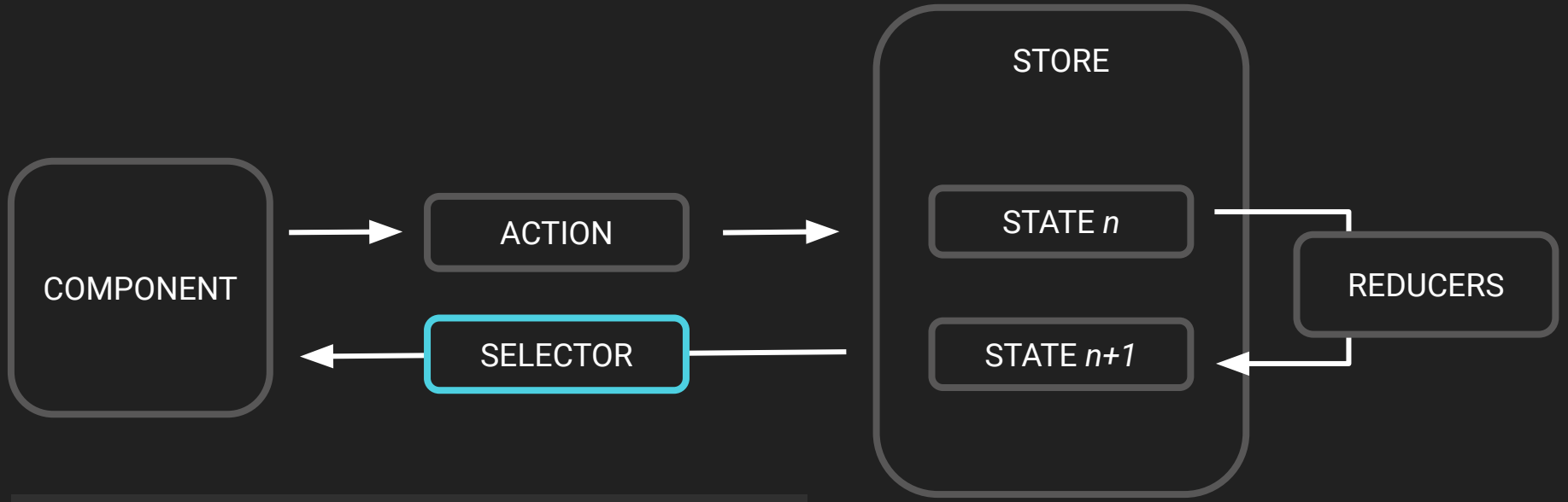



```
export const removeItem = createAction(
  '[Cart] Remove Item',
  props<{ productId: number }>()
)
```



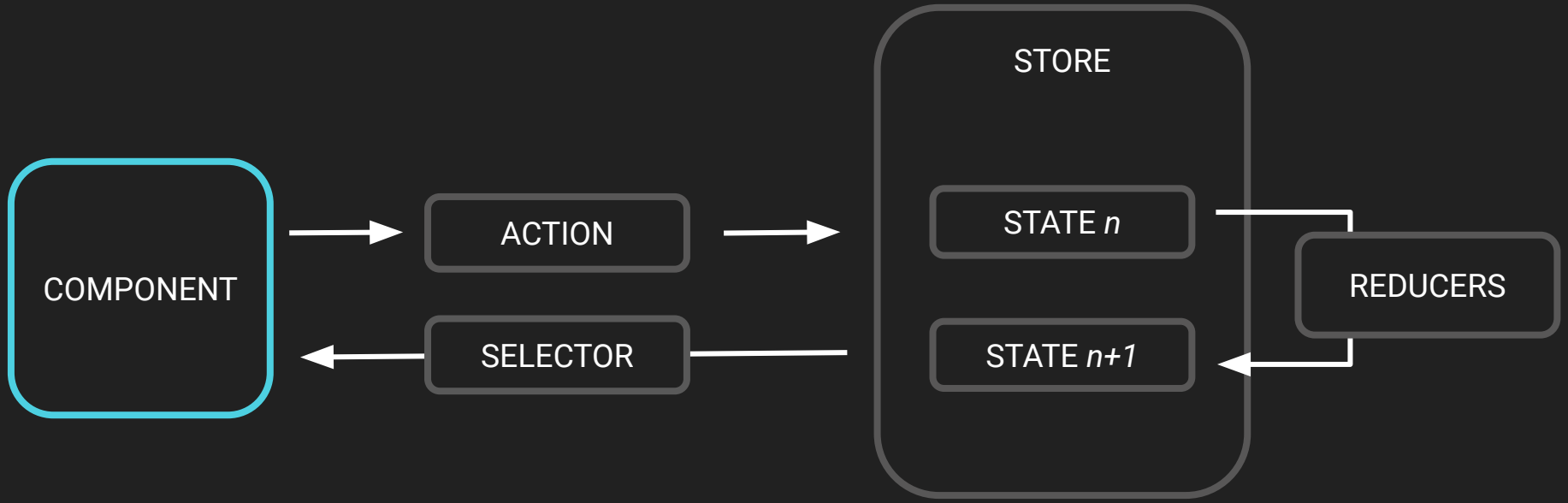

```
const cartReducer = createReducer(
  initialState,

  on(removeItem, (state, { productId }) => newState)
);
```




```
const getCartState = createFeatureSelector('cart');

export const getCartItems = createSelector(
  getCartState,
  (state: State): App.CartItem[] => state.items)
);
```


COMPONENT
ACTION
SELECTOR
STORE
STATE n
STATE n+1
REDUCERS

Gettin’ Started

Design credit: https://seedlipdrinks.com/uk/

Recap

- Our application state is all together
- Components are no longer responsible for composing data structures based on state
- Our services are no longer responsible for managing, updating, or selecting state

When should I use it?

- **Shared**: state is accessed by many components
- **Hydrated**: state is persisted and rehydrated
- **Available**: state needs to be available through route transitions
- **Retrieved**: state must be retrieved using a side effect
- **Impacted**: state is impacted by actions from other sources

QUESTIONS?

Resources

- Official Docs
- Version 8 improvements
- Architecture Guide
- Where to Initiate Data Load
- Parameterized Selectors
- Selectors are more Powerful than you Think

Thank you!

Repo and slides will be made available later in the week!