

REDUX

A Predictable State Container for JavaScript Apps

REDUX IS SIMPLE

- **It's just a couple of functions - you could write it yourself**
- **Simplicity is powerful**
- **All signals to change application state go through one channel**
- **Encourages/enforces programming tactics of immutability and function purity**

THREE PRINCIPLES

- **Single source of truth**
- **State is read-only**
- **Changes can be requested through actions and are made with pure functions**
 - No side effects (AJAX, mutations)
 - Same input => same output, no matter what!

BACK TO REACT

WHY NOT COMPONENT STATE?



```
      <Main />
     /      \
<Sidebar/>  <Player/>
```



```
      <Main />
    /      \
<Sidebar/>  <Player/> this.state = { currentSong }
```



```

      /      \
    <Sidebar/> <Player/> this.state = { currentSong }
      /
    <MiniPlayer/>
```




```

      /      \
    <Sidebar/> <Player/>
      /
    <MiniPlayer/>

    <Main />   this.state = { currentSong }

```

**SOLUTION: KEEP ALL STATE
IN ONE PLACE**

```

                                <Main /> this.state = { everything }
                                /      \
                                <Sidebar/> <Player/>
                                /      \      |      \
                                <MiniPlayer/> <Navigation/> <Controls/> <ProgressBar/>
                                |
                                <Controls/>
                                /      \
                                <Pause/> <Play/>

```

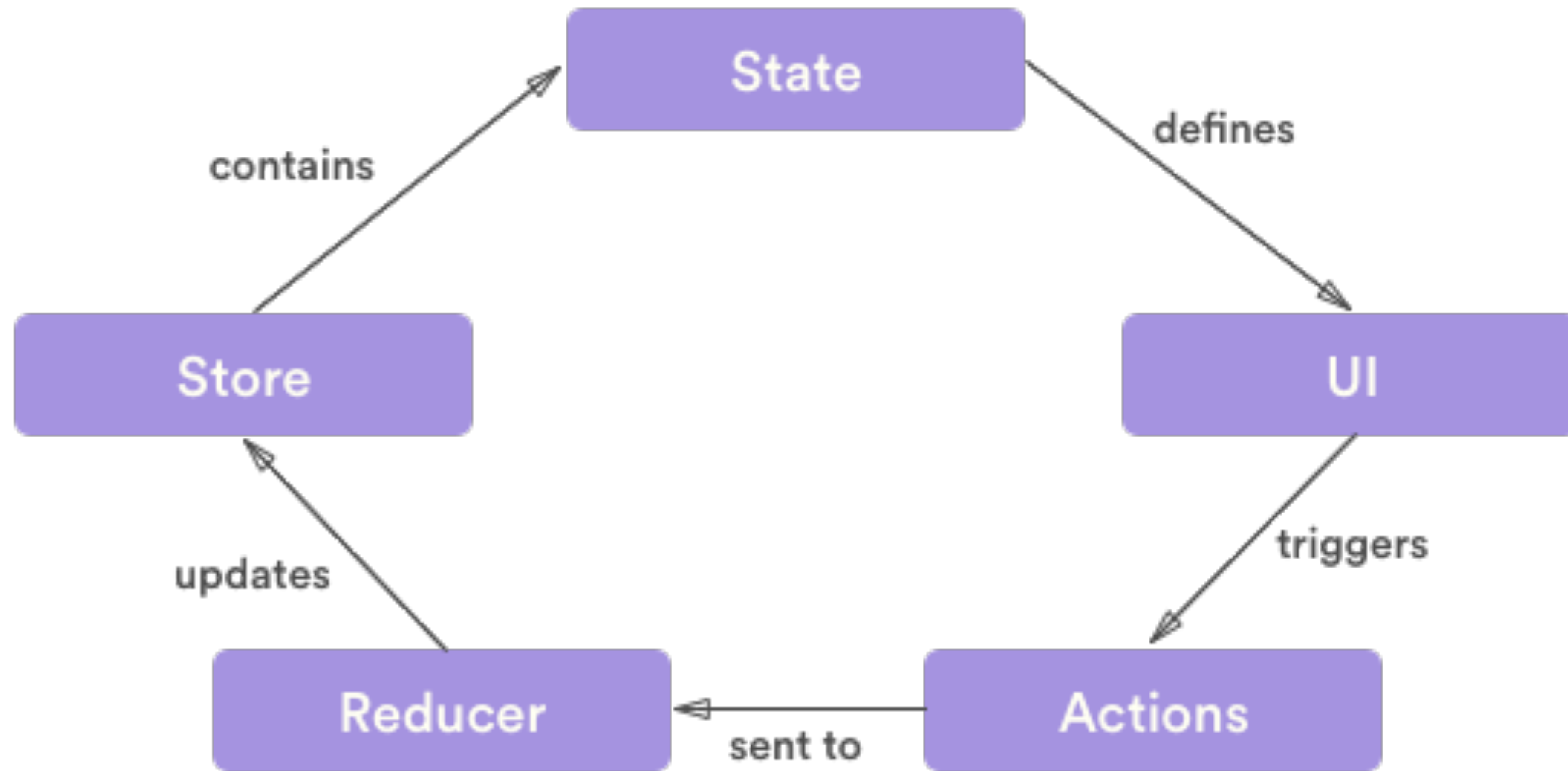
**PROBLEM: PASSING PROPS
ALL THE WAY DOWN IS A PAIN**

SOLUTION....



REACT WITH REDUX

- **React does not try to solve the problem of application state or architecture**
- **Redux is a good fit for a component-system based on unidirectional data flow**
 - “State at the top, passed down as a prop”
- **Components dispatch actions to Redux store, new state is produced, components listen and update**



http://www.theodo.fr/uploads/blog//2016/03/ui_workflow.png

CONNECTING COMPONENTS

- **Your components shouldn't be aware of Redux - Just use props & render.**
- **Wrap your component into a Container (or Connected) component**
- **Connected component acts as a bridge:**
 - Subscribe to the Redux Store
 - Syncs only the parts of the store state we want
 - Maps the dispatches
 - Passes everything down as props to your original component

Demo

Store

State {}

State

```
{  
  albums: [],  
  artists: [],  
  selectedAlbum: {},  
  // etc..  
}
```

Store

State {}

Our React Components

Playlists

Albums

Artists

Store

State {}

store.subscribe
function () {}

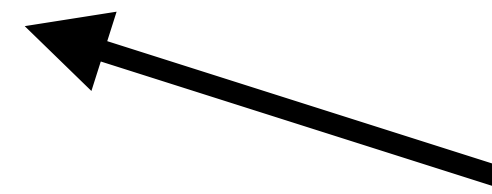
Our React Components

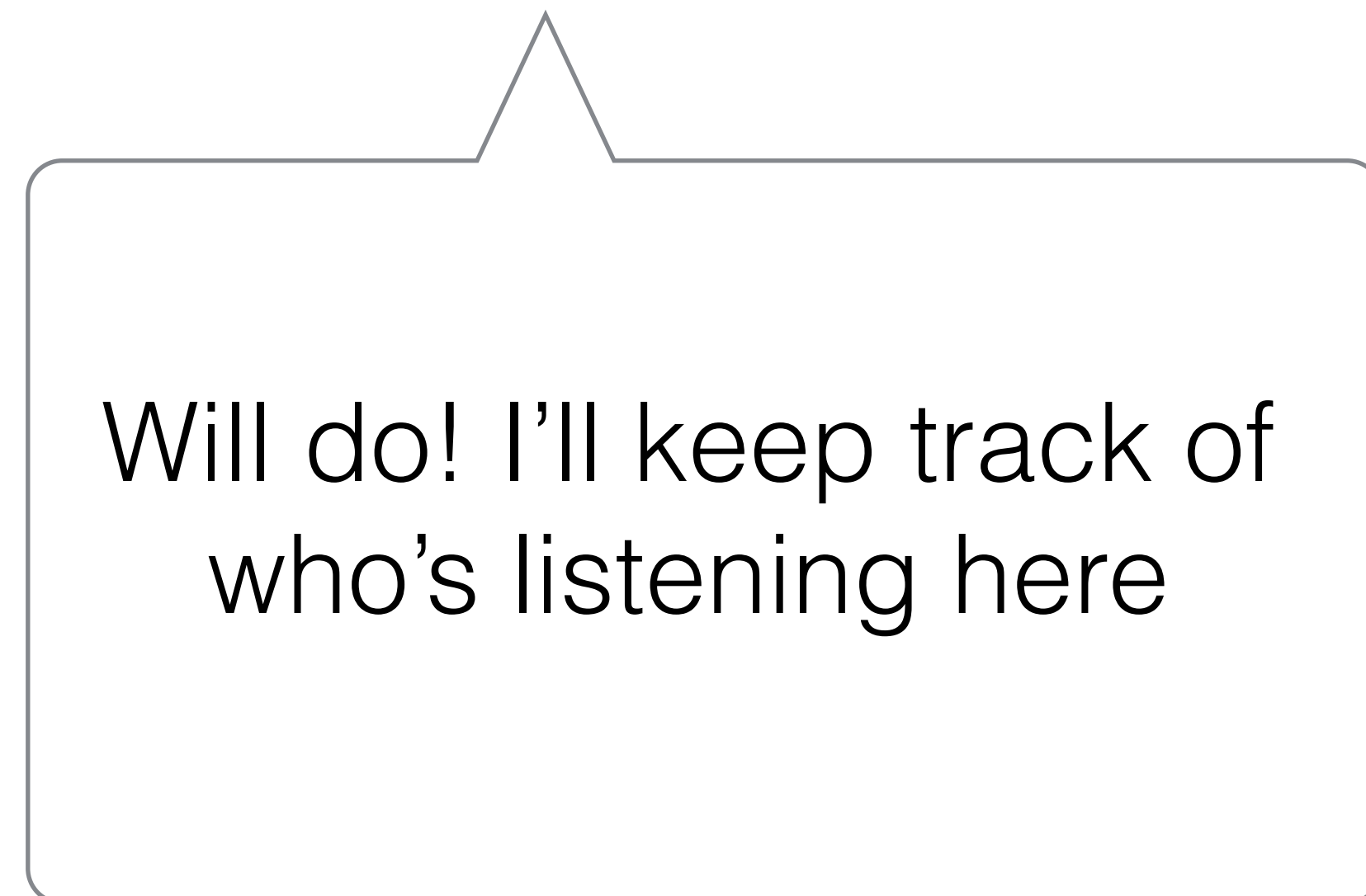
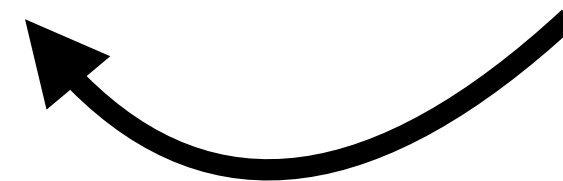
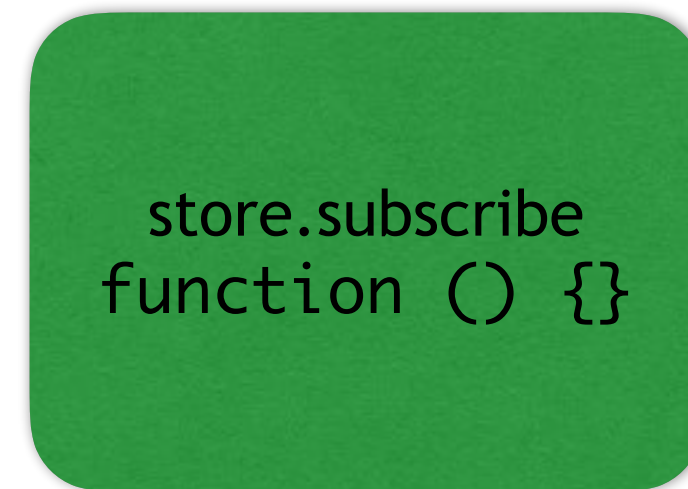
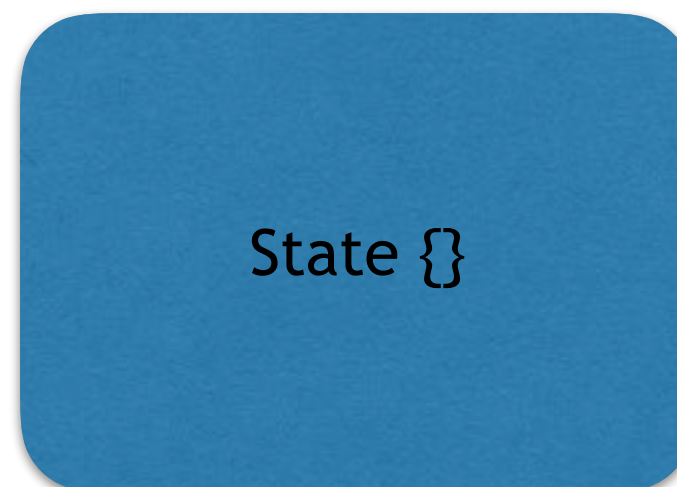
Playlists

Albums

Artists

Hey Store, when
state changes, let us
know, and we'll
re-render





Our React Components



Store

State {}

store.dispatch
function () {}

Listeners []

store.subscribe
function () {}

Now - if you want to
change something, send
an "action" to my
"dispatch" method

Our React Components

Playlists

Albums

Artists

Action

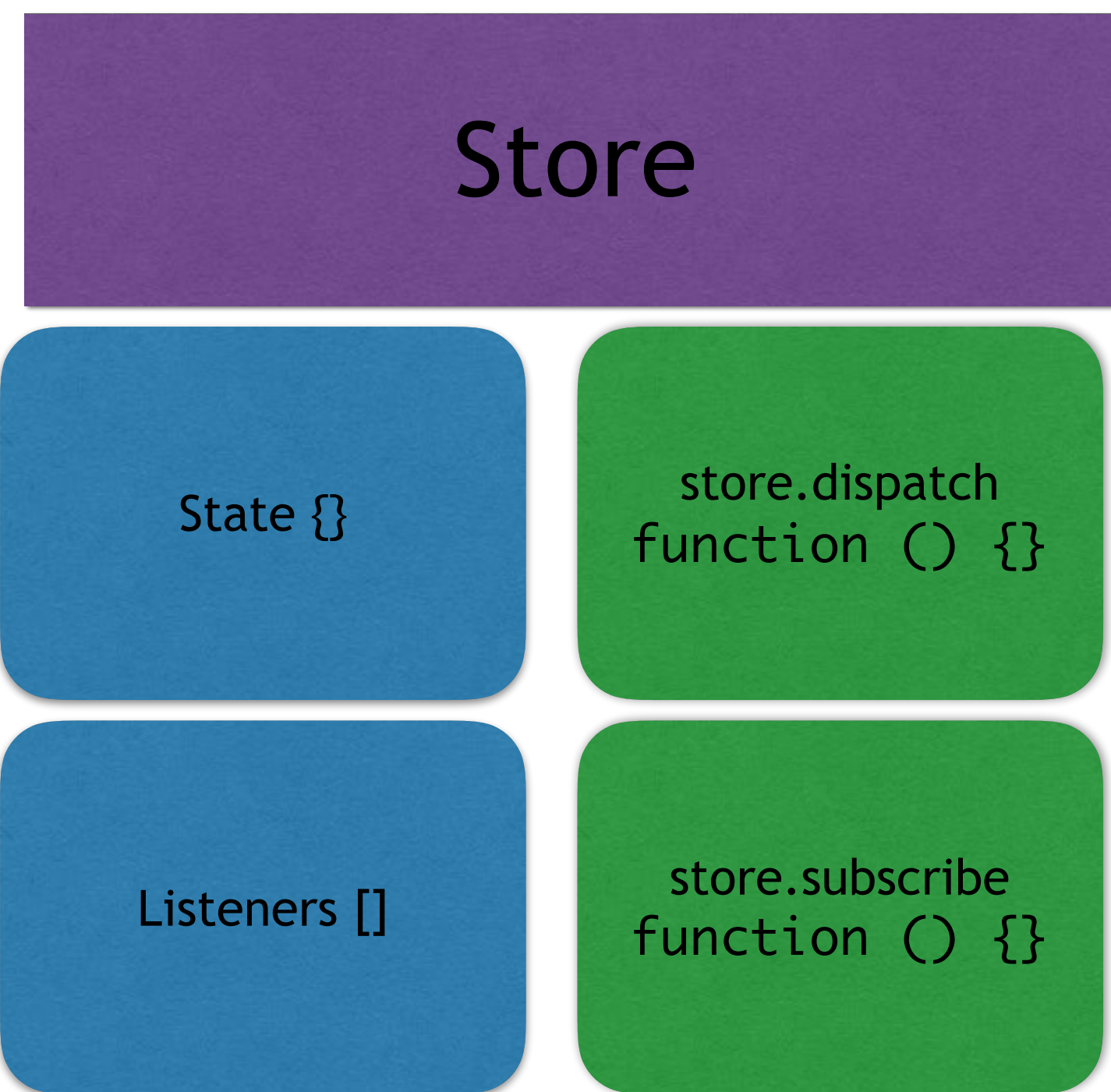
```
{  
  type: ADD_ALBUM,  
  newAlbum: {  
    id: 6,  
    name: "4:44",  
    artist: "Jay-Z"  
  }  
}
```

Action

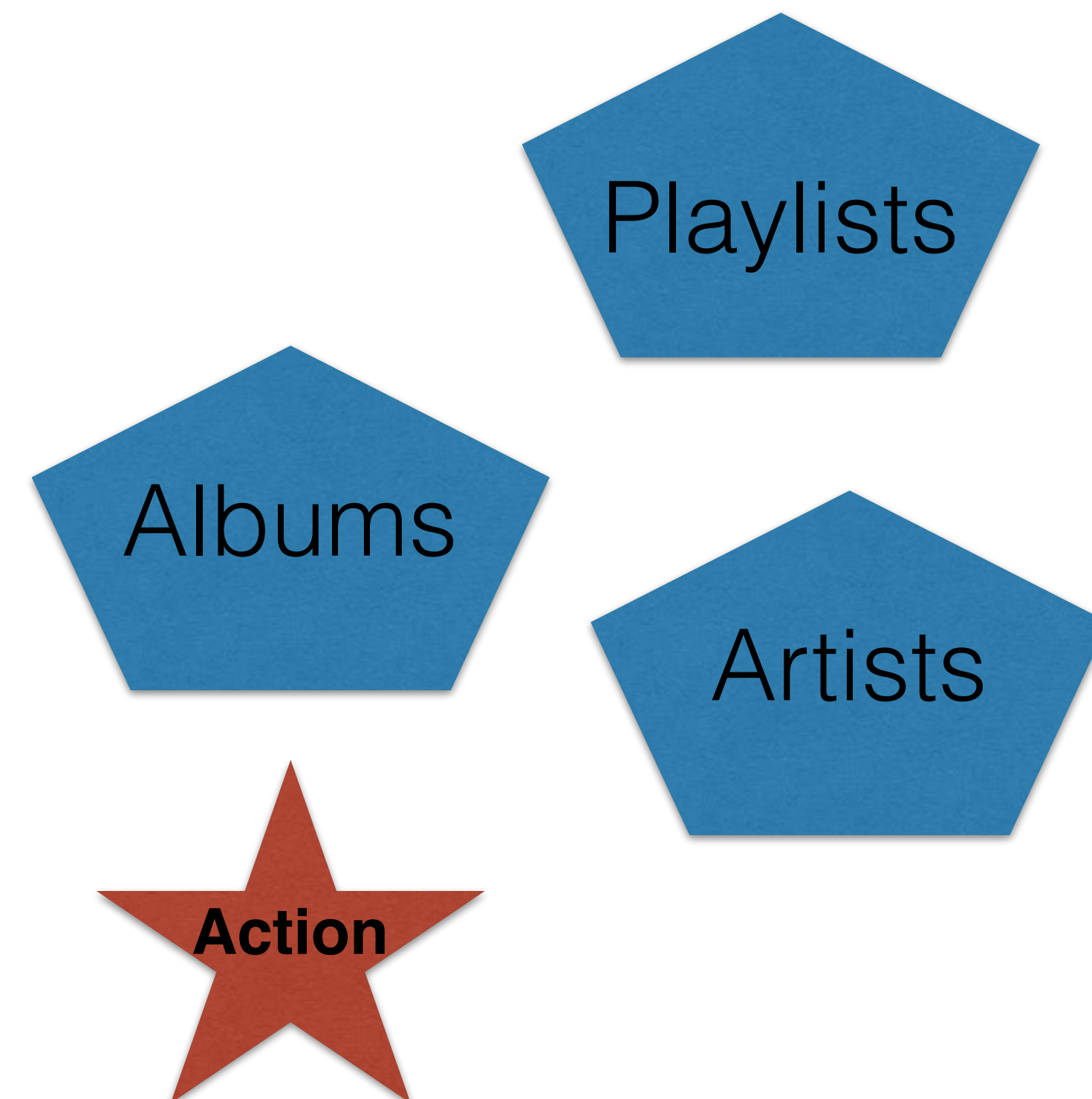
```
{  
  type: ADD_ALBUM, ← “Action Type”  
  newAlbum: {  
    id: 6,  
    name: “4:44”,  
    artist: “Jay-Z”  
  }  
}
```

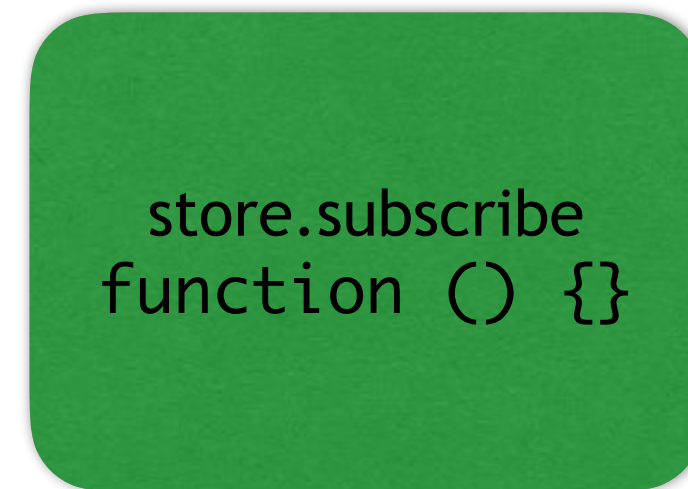
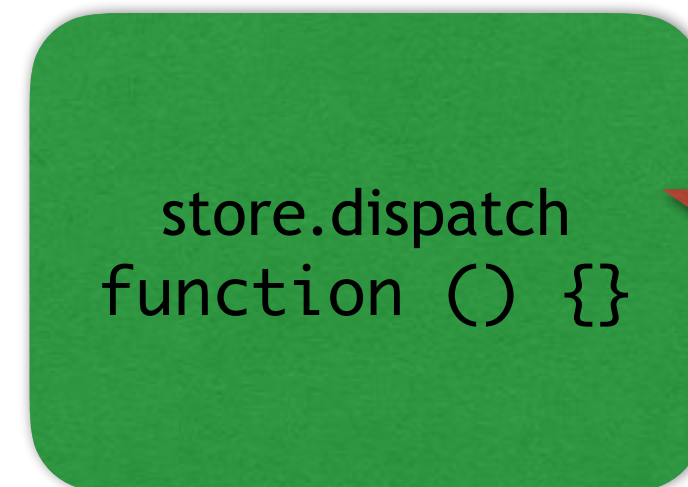
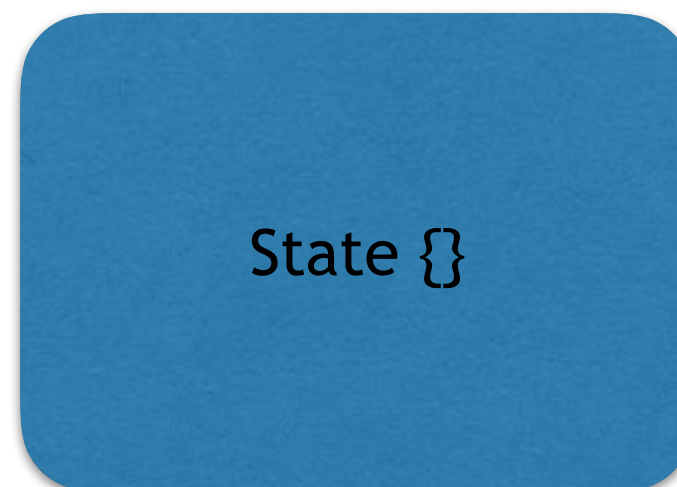
store.dispatch

The ONLY way to send
an action to the
store, so that it
can update the state

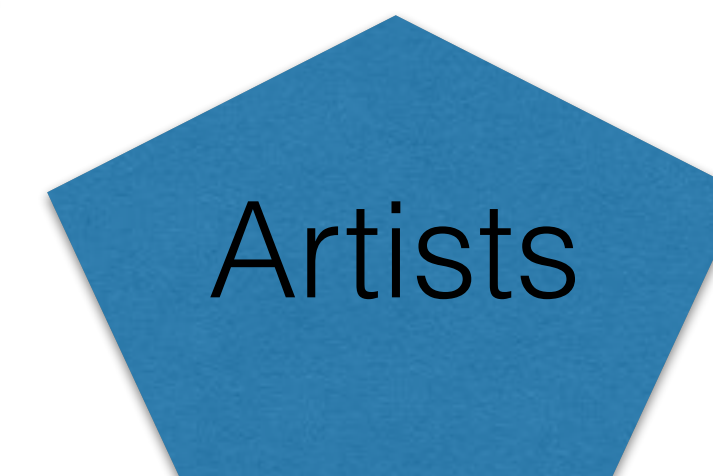


Our React Components

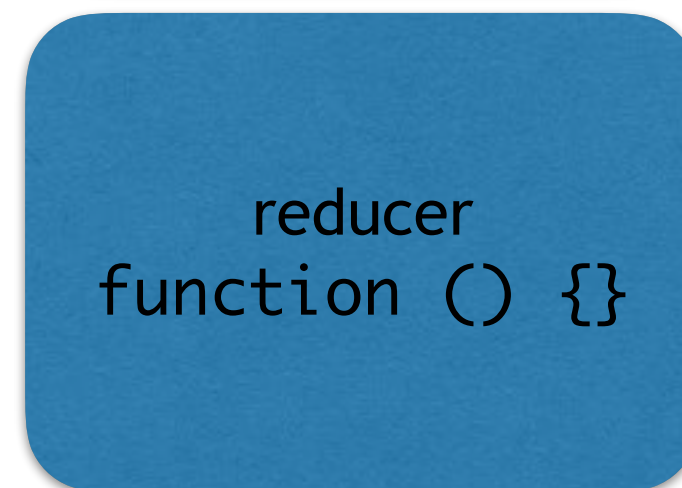
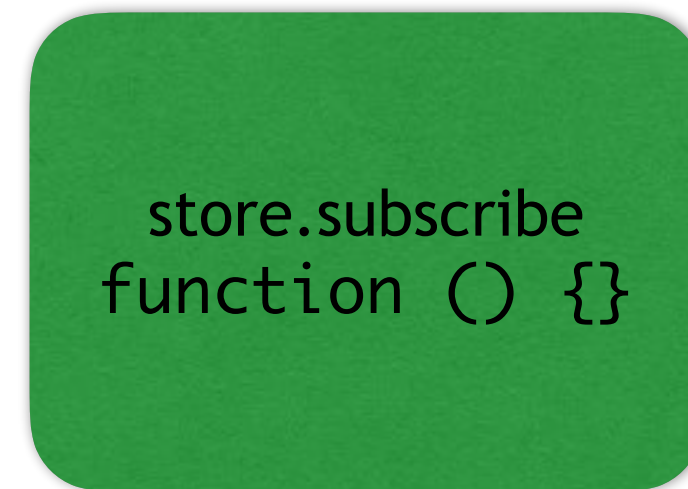
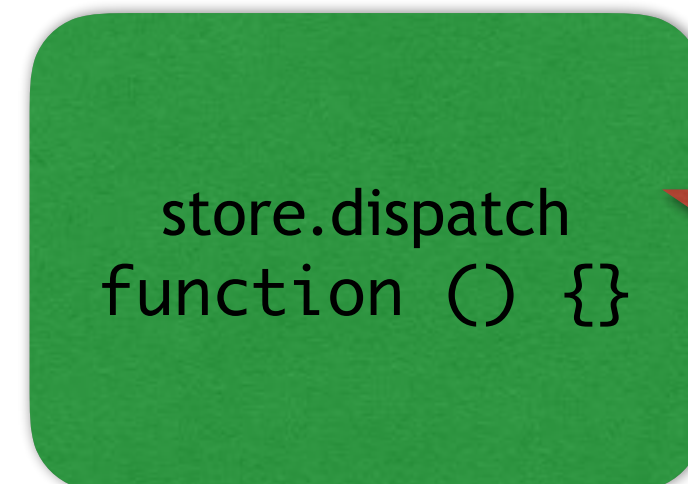
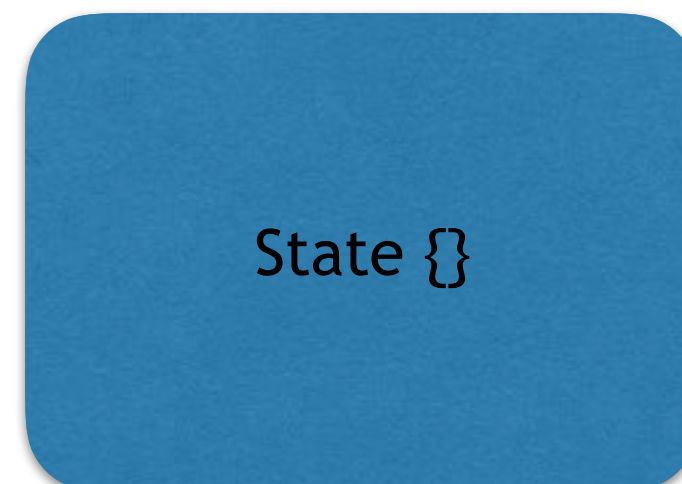




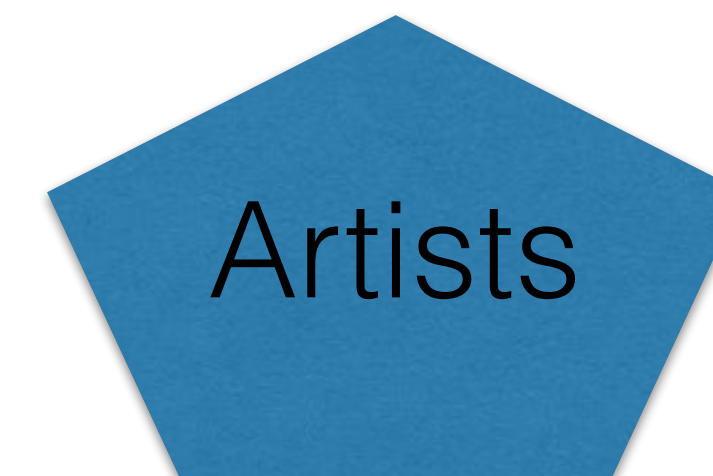
Our React Components



Wait a moment - I've got
this action...but how
do I know **how** to
change the state?



Our React Components



Oh yeah, you gave
me this!

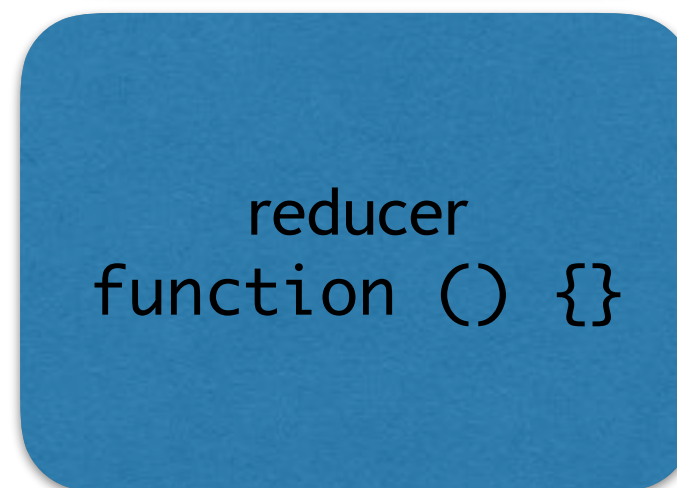
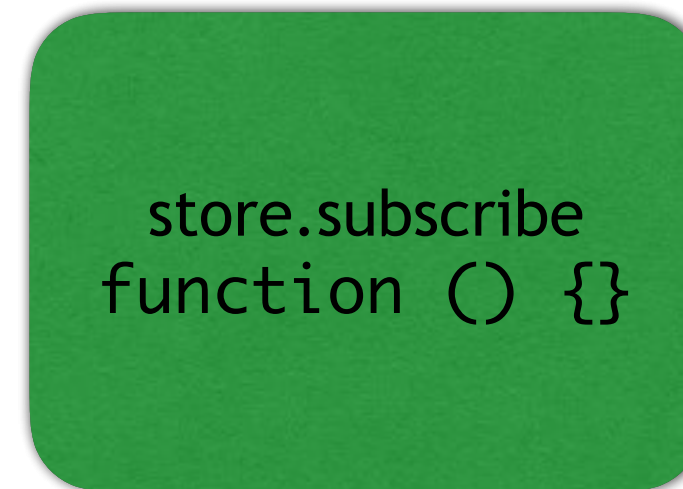
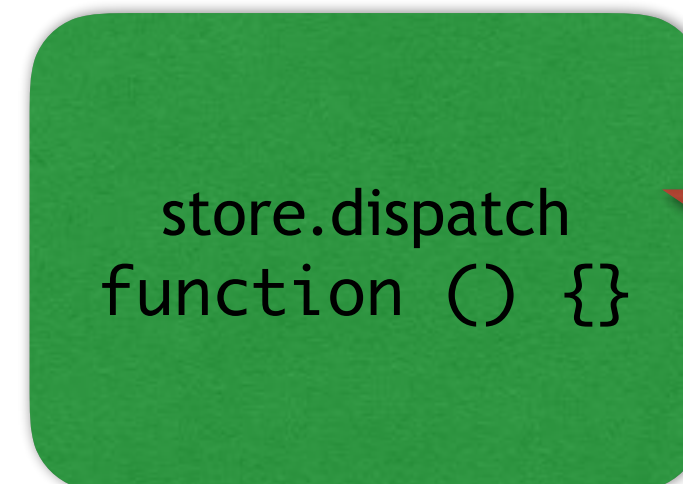
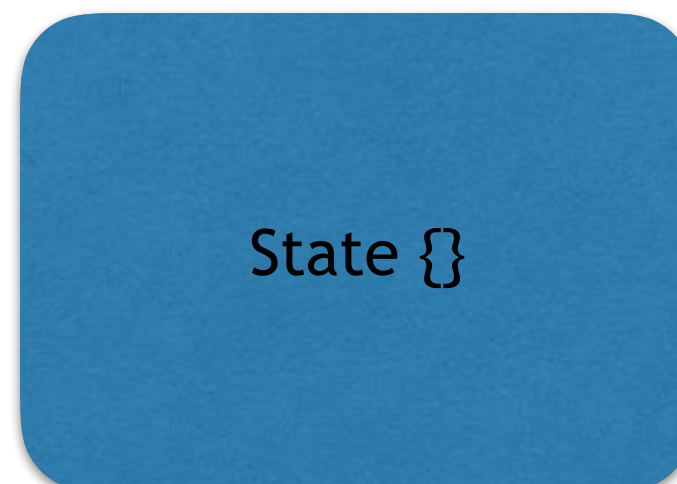
A white speech bubble with a grey border and a pointer directed at the reducer box.

Reducer

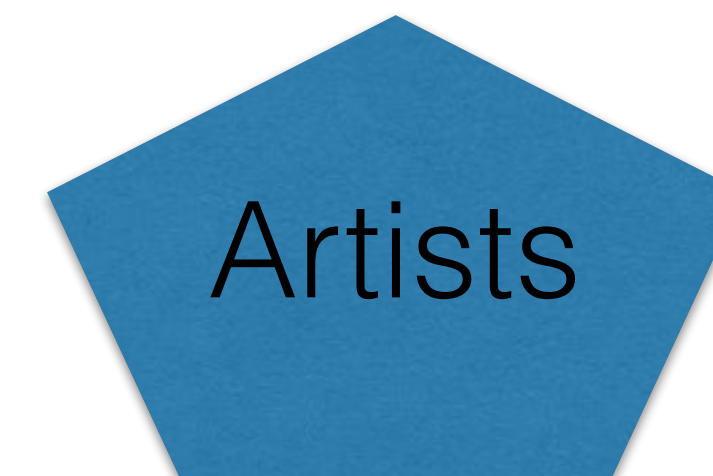
```
function reducer (prevState, action) {  
  
  // returns a NEW state object  
  // based on the previous state  
  // and the action  
  
}
```

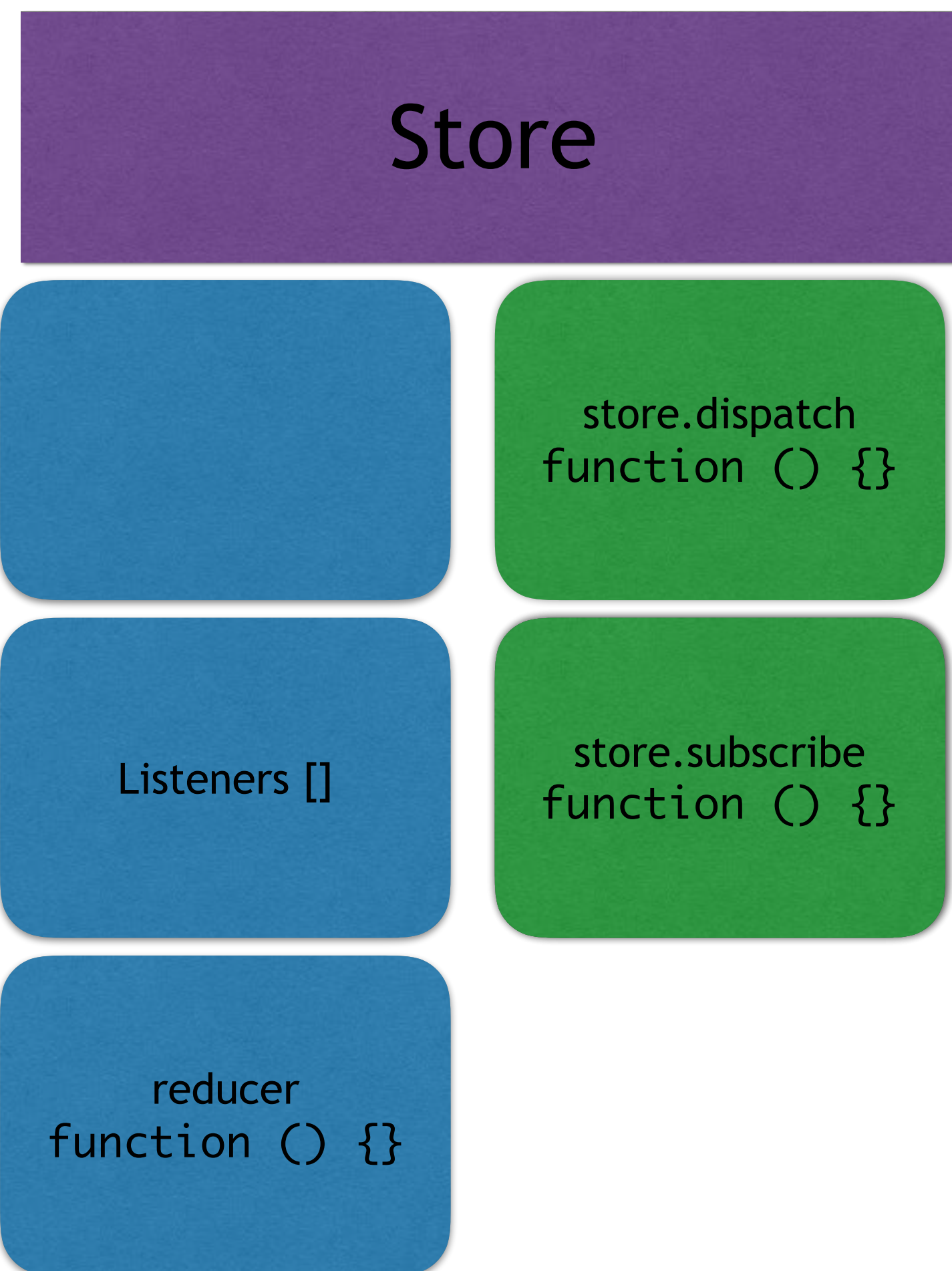
Reducer

```
function reducer (prevState, action) {  
  
  const newState = Object.assign({}, prevState)  
  
  switch (action.type) {  
    case ADD_ALBUM:  
      newState.albums = newState.albums.concat(action.newAlbum)  
      return newState  
    default:  
      return prevState  
  }  
  
}
```

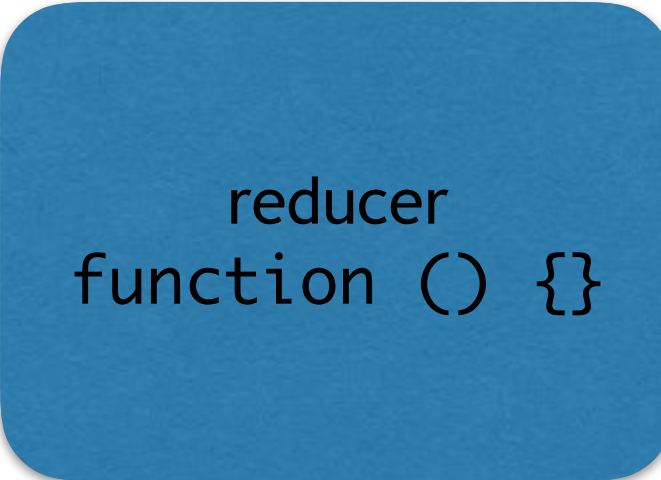
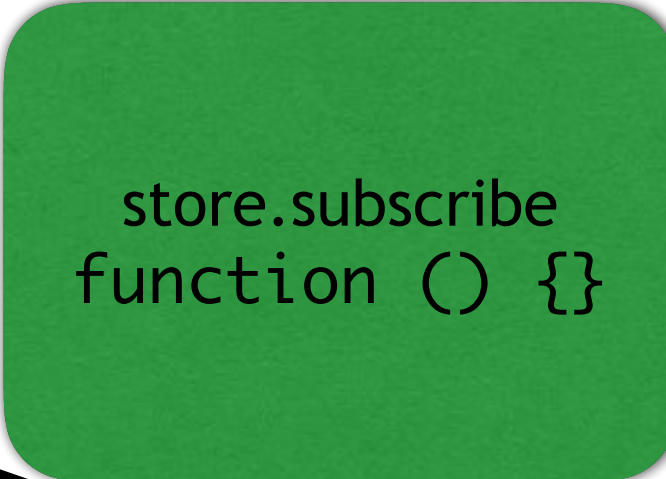
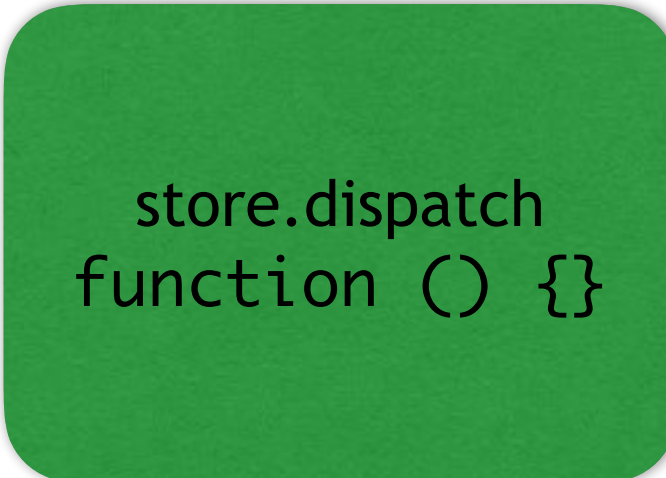
Our React Components





Our React Components

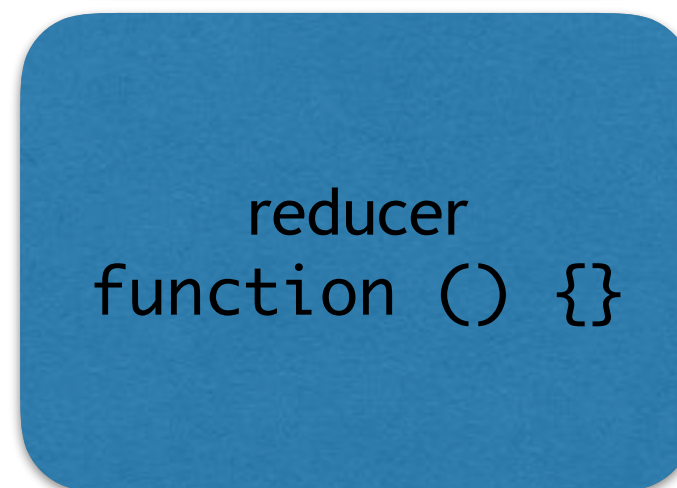
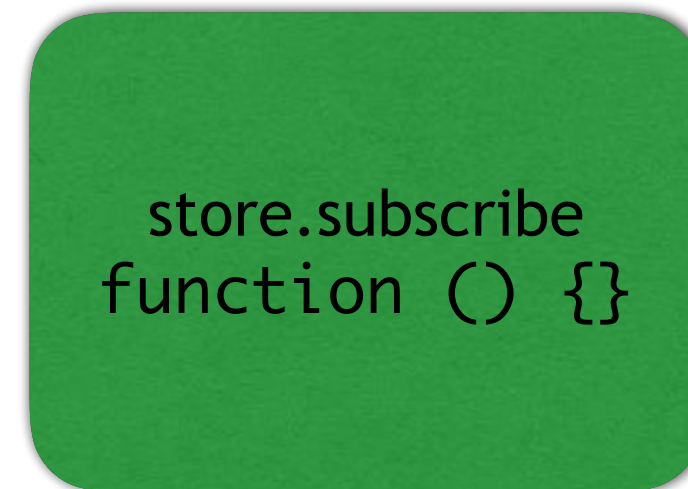
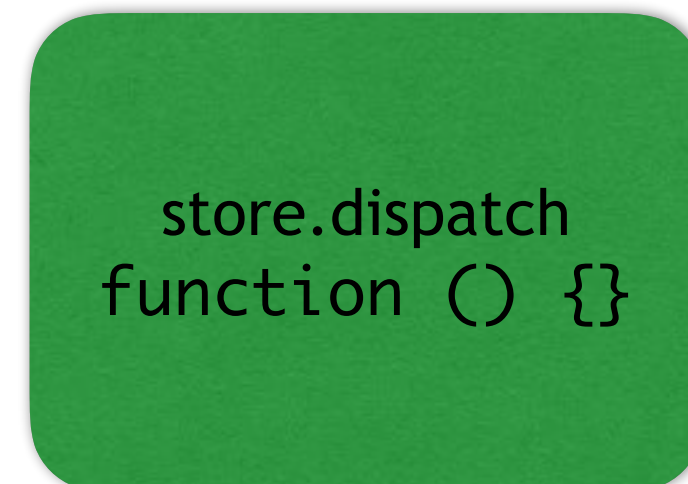




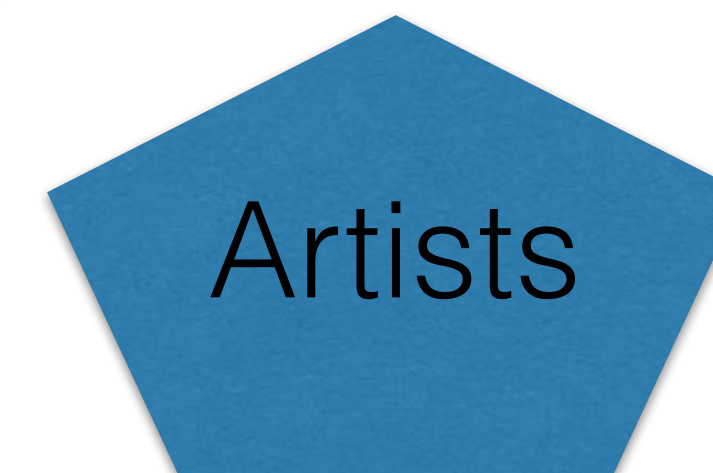
Hey! New state is ready!

Our React Components





Our React Components



Cool! ...can you
give it to us?

A speech bubble indicating a request from the React components to the Store.

Store

NEW STATE {}

store.dispatch
function () {}

Listeners []

store.subscribe
function () {}

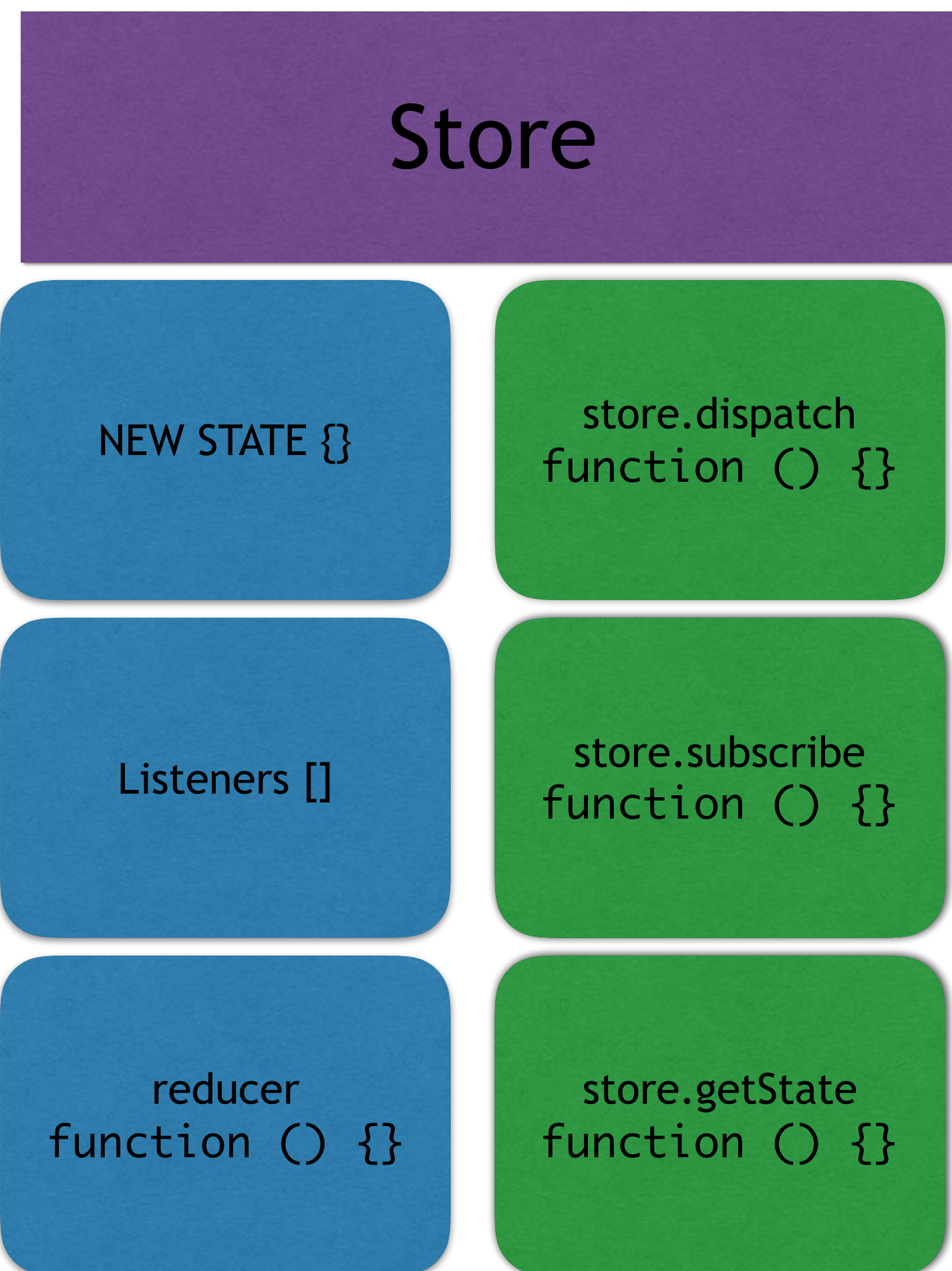
reducer
function () {}

store.getState
function () {}

Oh, yeah. Use this

Our React Components





Our React Components



Re-render with new state!

The Store

External Methods

getState()

subscribe(listener)

dispatch(action)

exposes current state

registers listener

invokes:
reducer(state, action)
to get the new state

Internal

State

Listeners

Your
Reducer

2. invokes all
registered listeners

1. swaps old state with new state

