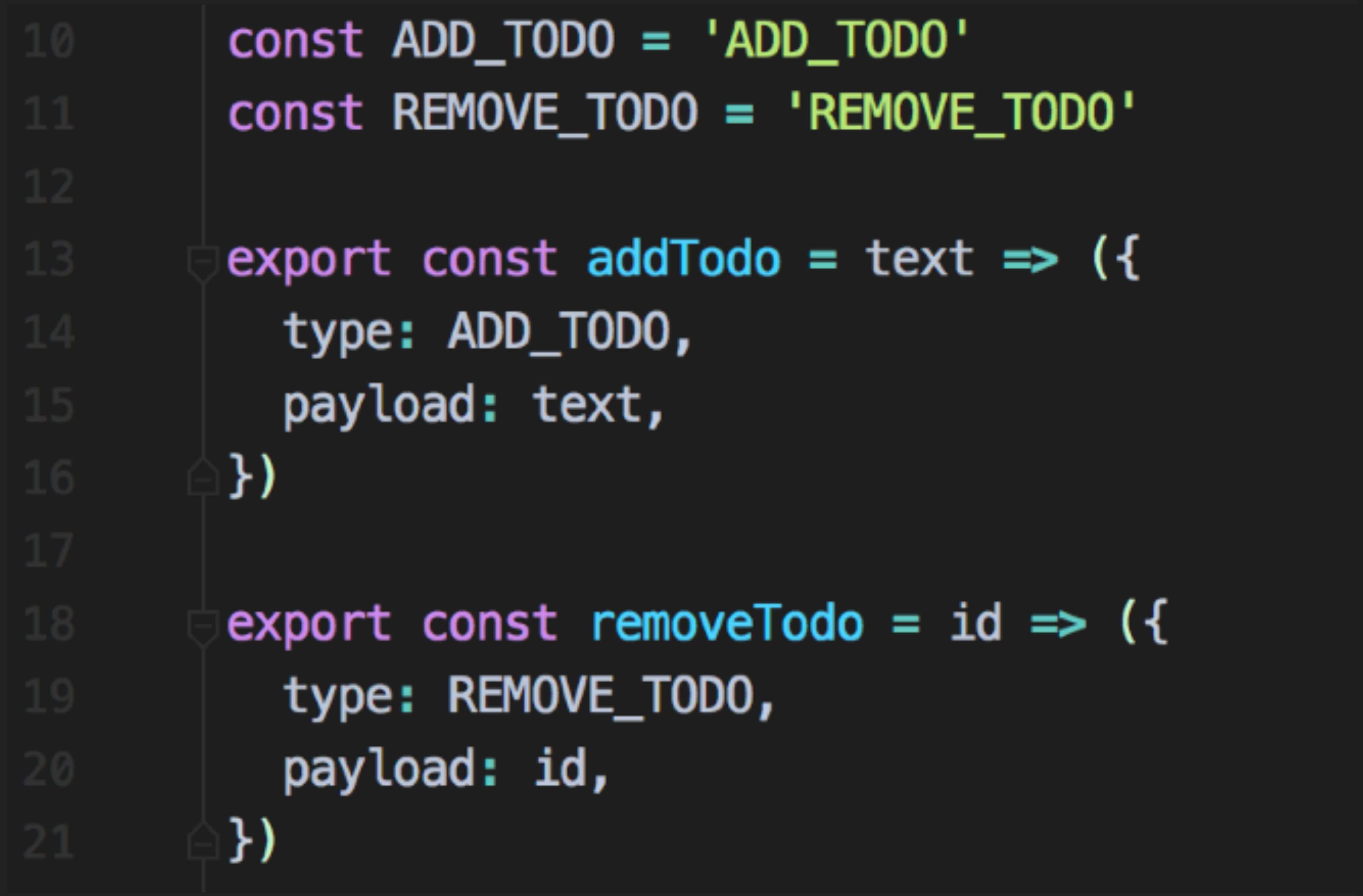
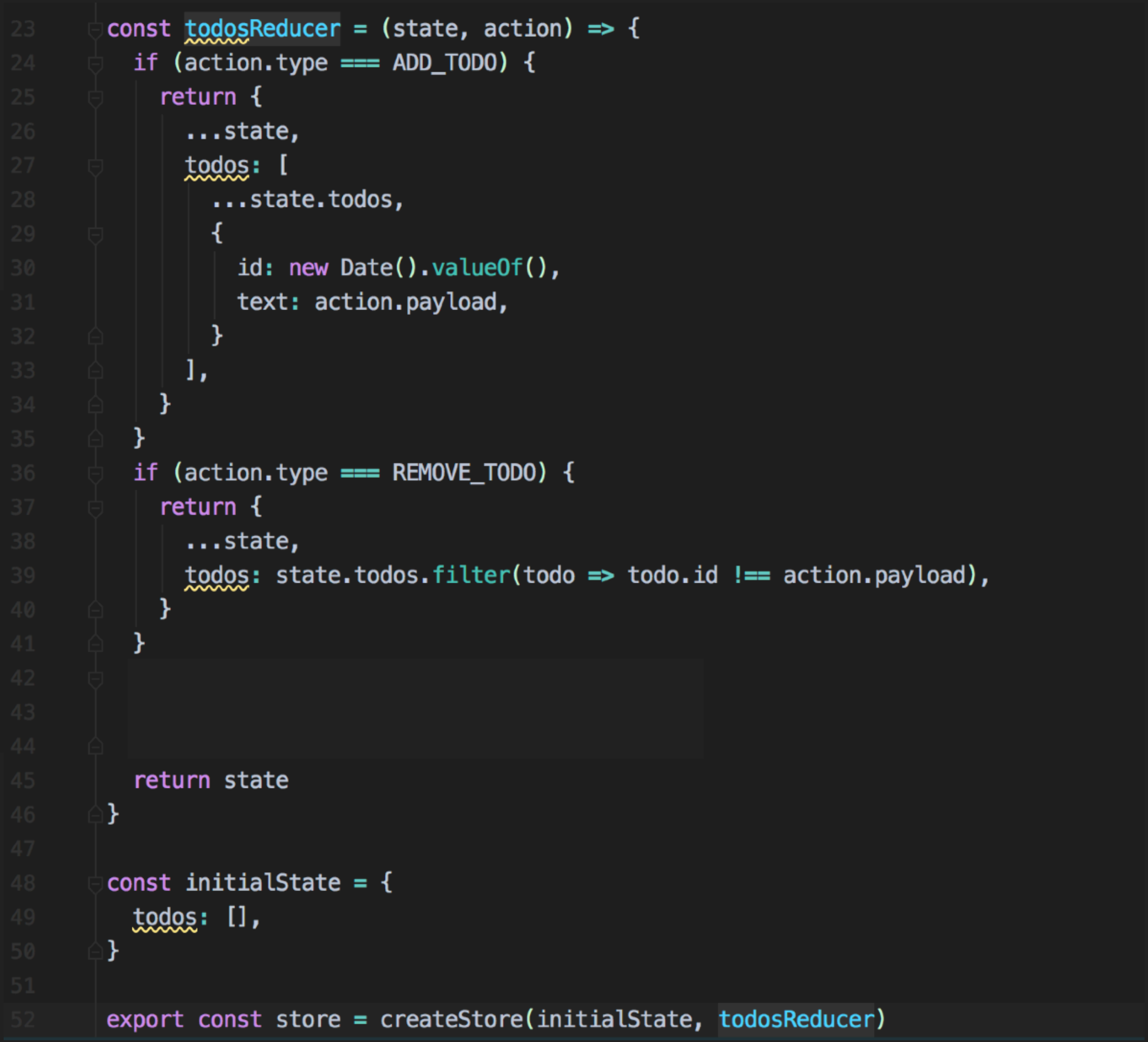
2. Создадим todos store



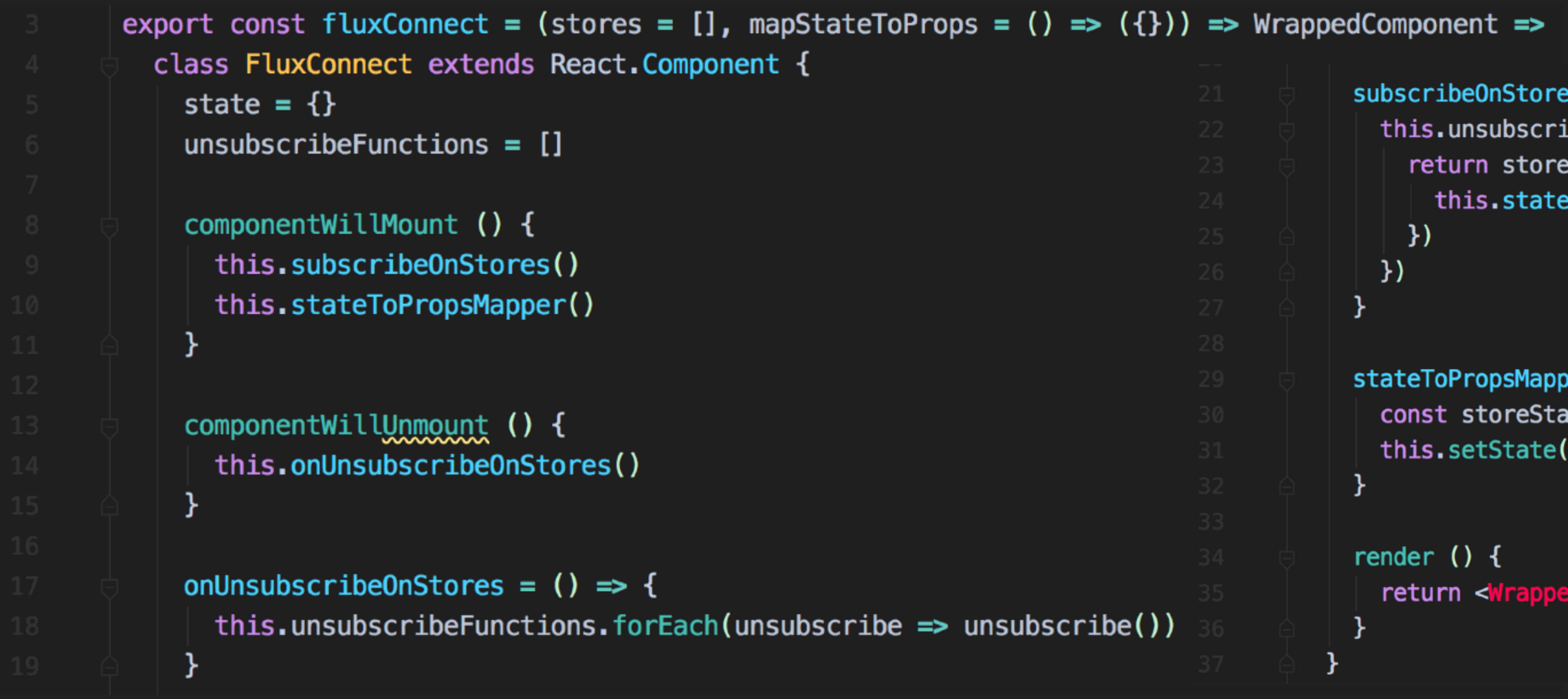
```
const ADD_TODO = 'ADD_TODO'
const REMOVE_TODO = 'REMOVE_TODO'

export const addTodo = text => ({
  type: ADD_TODO,
  payload: text,
})

export const removeTodo = id => ({
  type: REMOVE_TODO,
  payload: id,
})
```



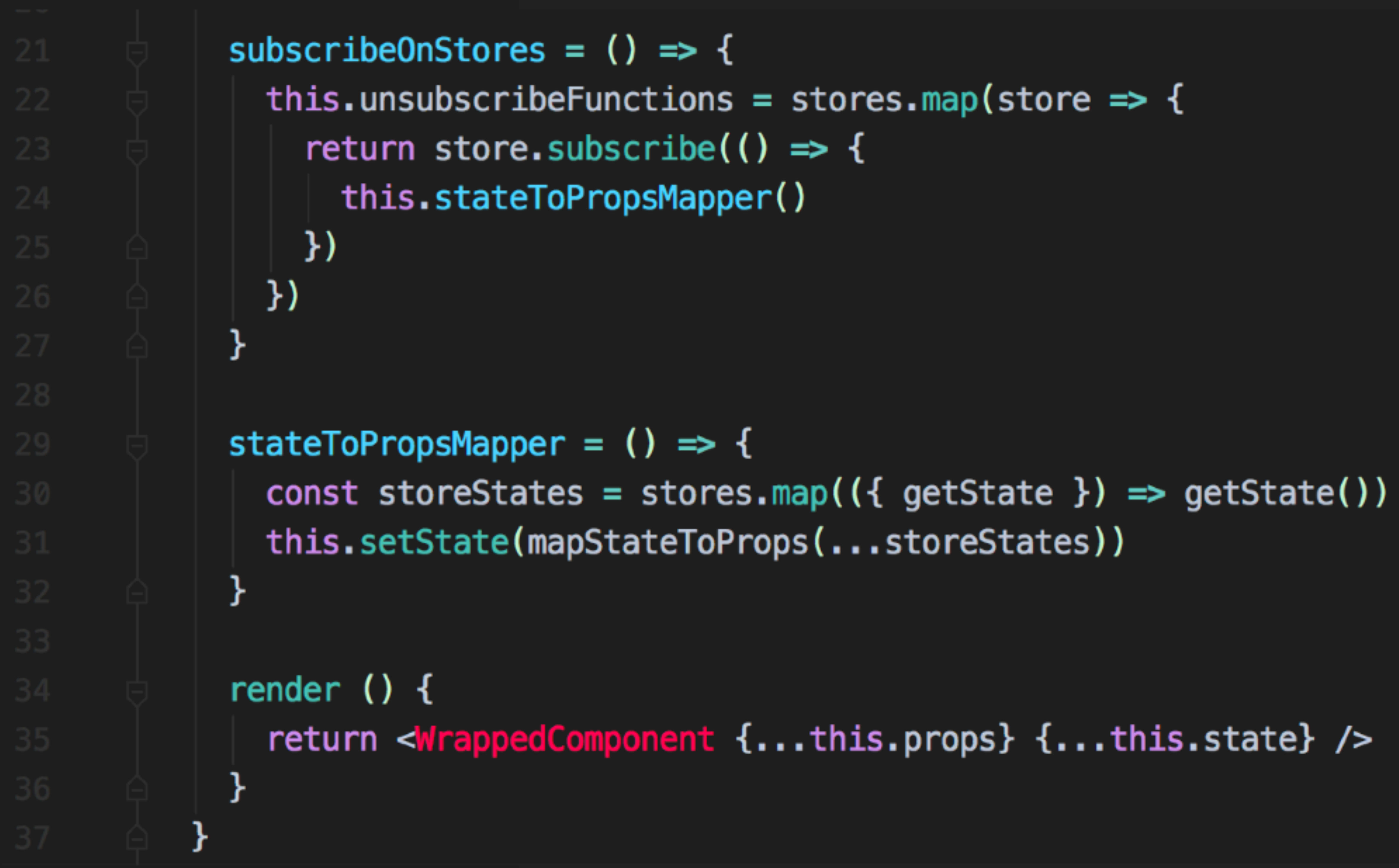
```
const todosReducer = (state, action) => {
  if (action.type === ADD_TODO) {
    return {
      ...state,
      todos: [
        ...state.todos,
        {
          id: new Date().valueOf(),
          text: action.payload,
        }
      ],
    }
  }
  if (action.type === REMOVE_TODO) {
    return {
      ...state,
      todos: state.todos.filter(todo => todo.id !== action.payload),
    }
  }



  return state
}

const initialState = {
  todos: [],
}

export const store = createStore(initialState, todosReducer)
```

3. Свяжем stores с view

Пишем свой FluxConnect



```
export const fluxConnect = (stores = [], mapStateToProps = () => ({})) => WrappedComponent =>
  class FluxConnect extends React.Component {
    state = {}
    unsubscribeFunctions = []

    componentWillMount () {
      this.subscribeOnStores()
      this.stateToPropsMapper()
    }

    componentWillUnmount () {
      this.onUnsubscribeOnStores()
    }

    onUnsubscribeOnStores = () => {
      this.unsubscribeFunctions.forEach(unsubscribe => unsubscribe())
    }
```



```
    subscribeOnStores = () => {
      this.unsubscribeFunctions = stores.map(store => {
        return store.subscribe(() => {
          this.stateToPropsMapper()
        })
      })
    }

    stateToPropsMapper = () => {
      const storeStates = stores.map(({ getState }) => getState())
      this.setState(mapStateToProps(...storeStates))
    }

    render () {
      return <WrappedComponent {...this.props} {...this.state} />
    }
  }
```