Лекция 10
Redux

Разработка интернет приложений

Канев Антон Игоревич

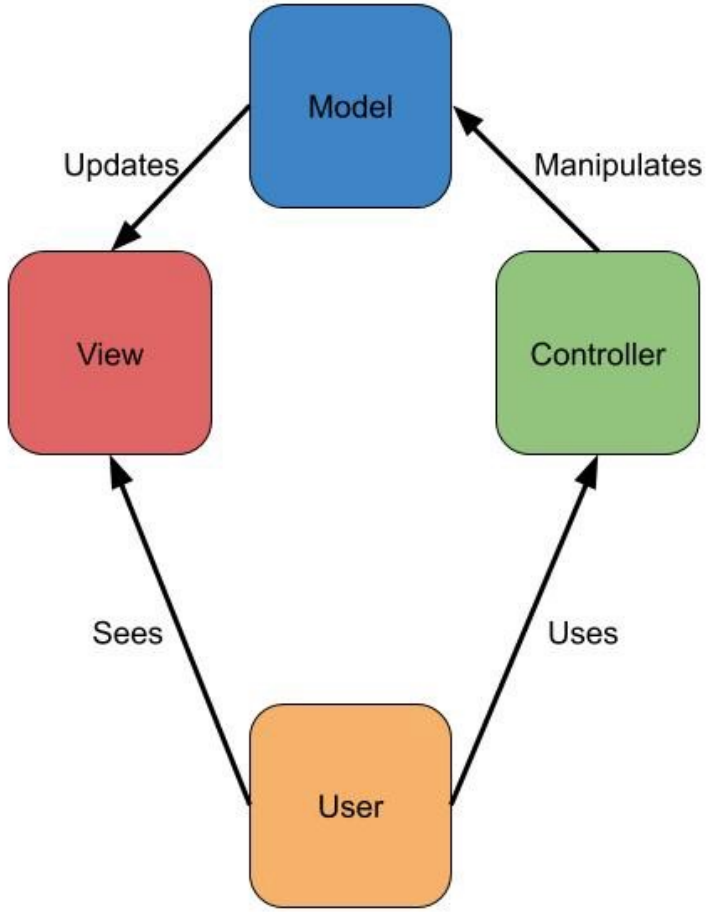
Доработки фронтенда

Авторизация

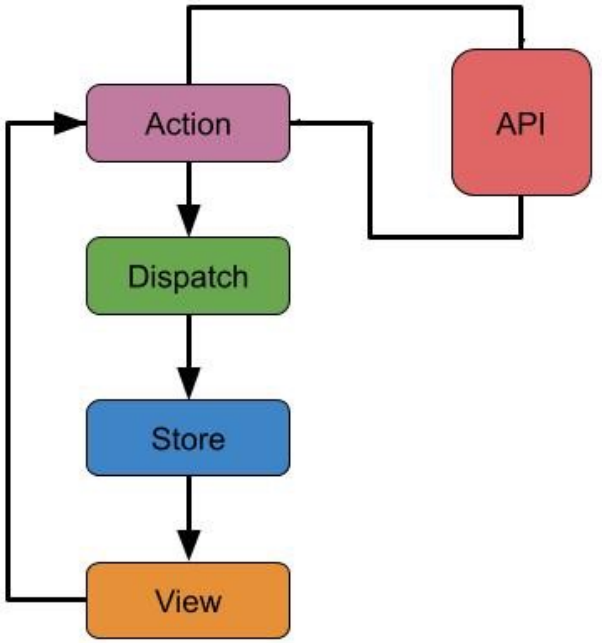
- Добавить окна регистрации и аутентификации
- Добавить логику проверки авторизации пользователя – после успешной авторизации меняется **состояние приложения**
- Авторизованный пользователь может разлогиниться
- Авторизованному пользователю доступен больший объем функционала в зависимости от его роли

- Поля поиска, сложные формы с дополнительными полями и тд

MVC vs Flux

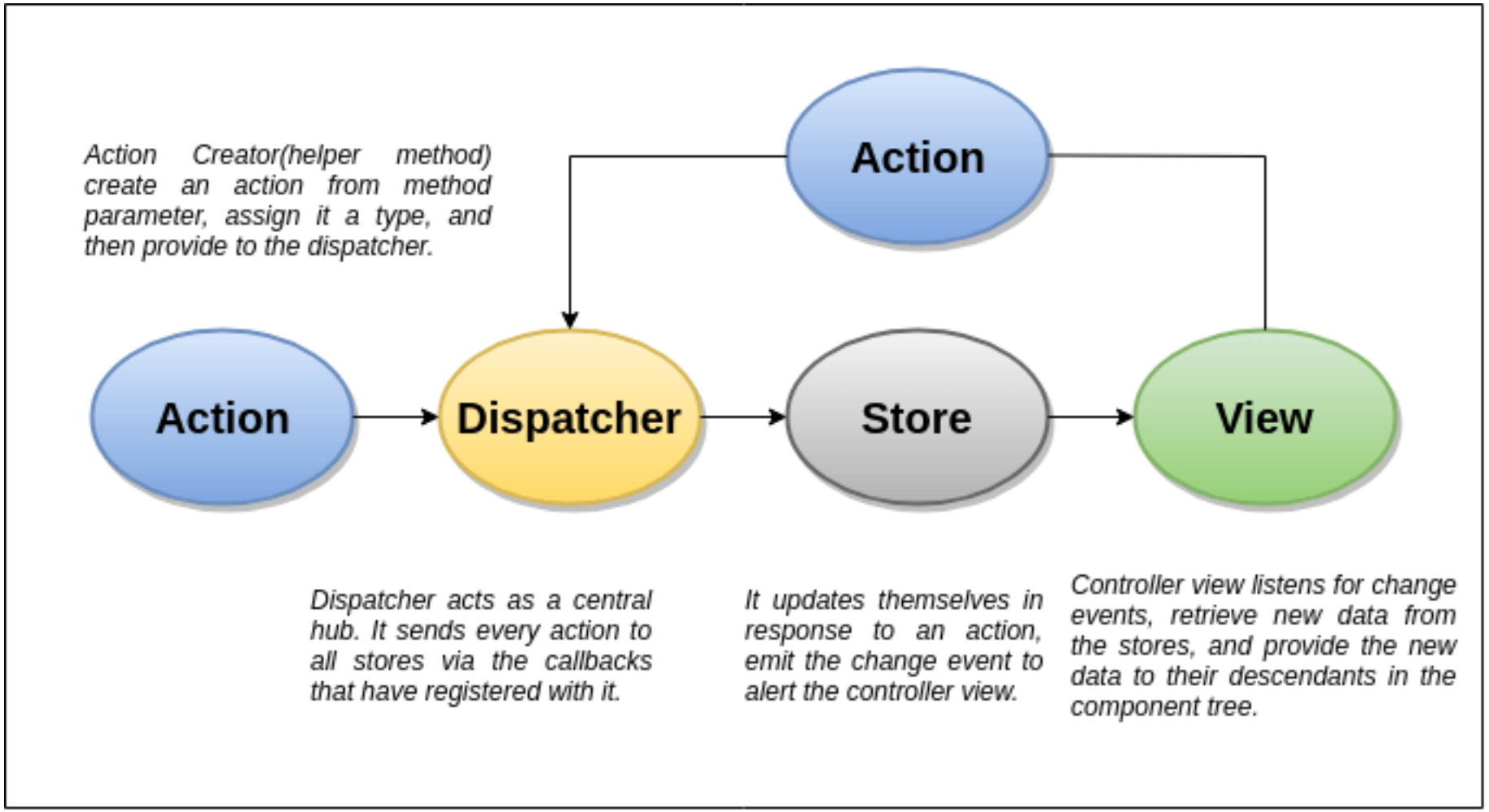


MVC base Model

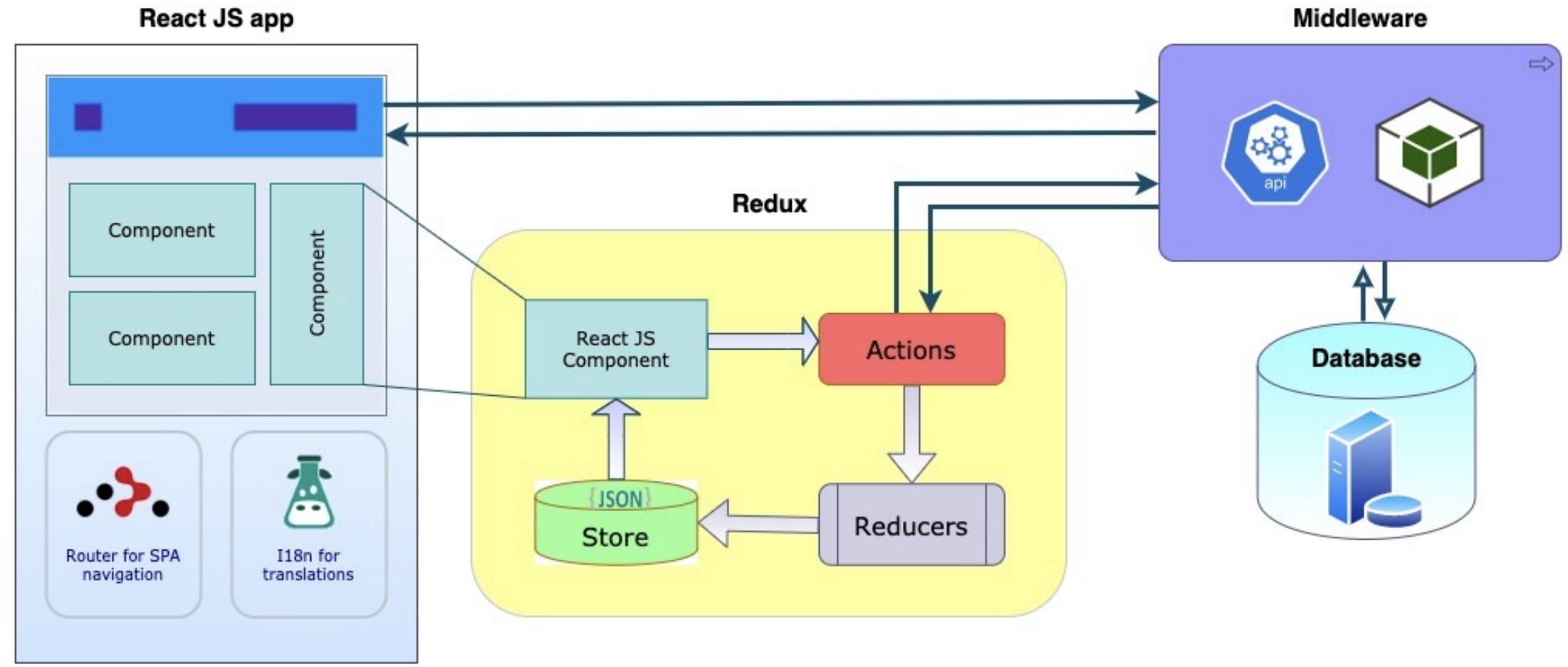


React's MVC model

Flux

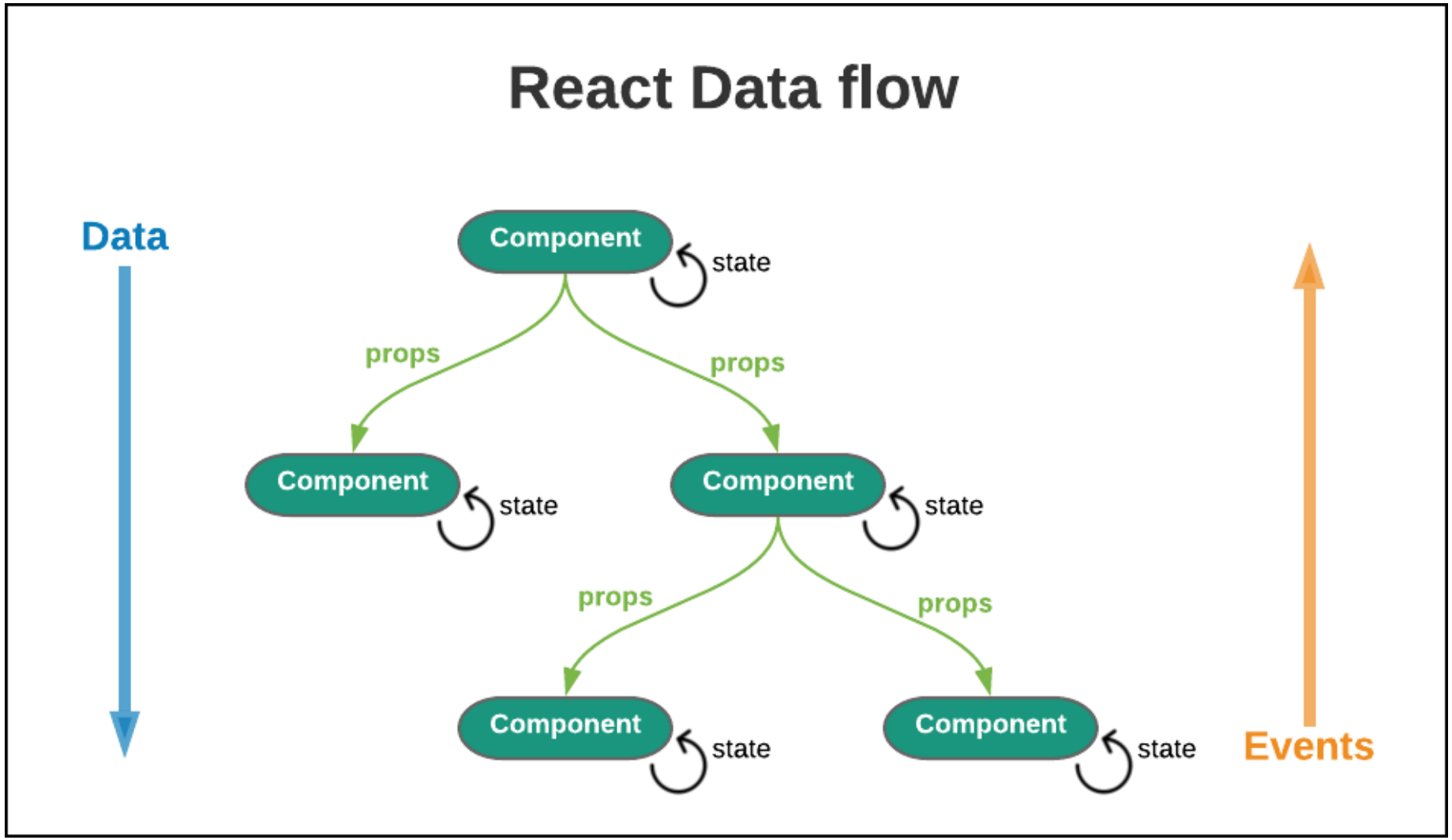

Action Creator(helper method) create an action from method parameter, assign it a type, and then provide to the dispatcher.
Action
Action
Dispatcher
Store
View
Dispatcher acts as a central hub. It sends every action to all stores via the callbacks that have registered with it.
It updates themselves in response to an action, emit the change event to alert the controller view.
Controller view listens for change events, retrieve new data from the stores, and provide the new data to their descendants in the component tree.

React

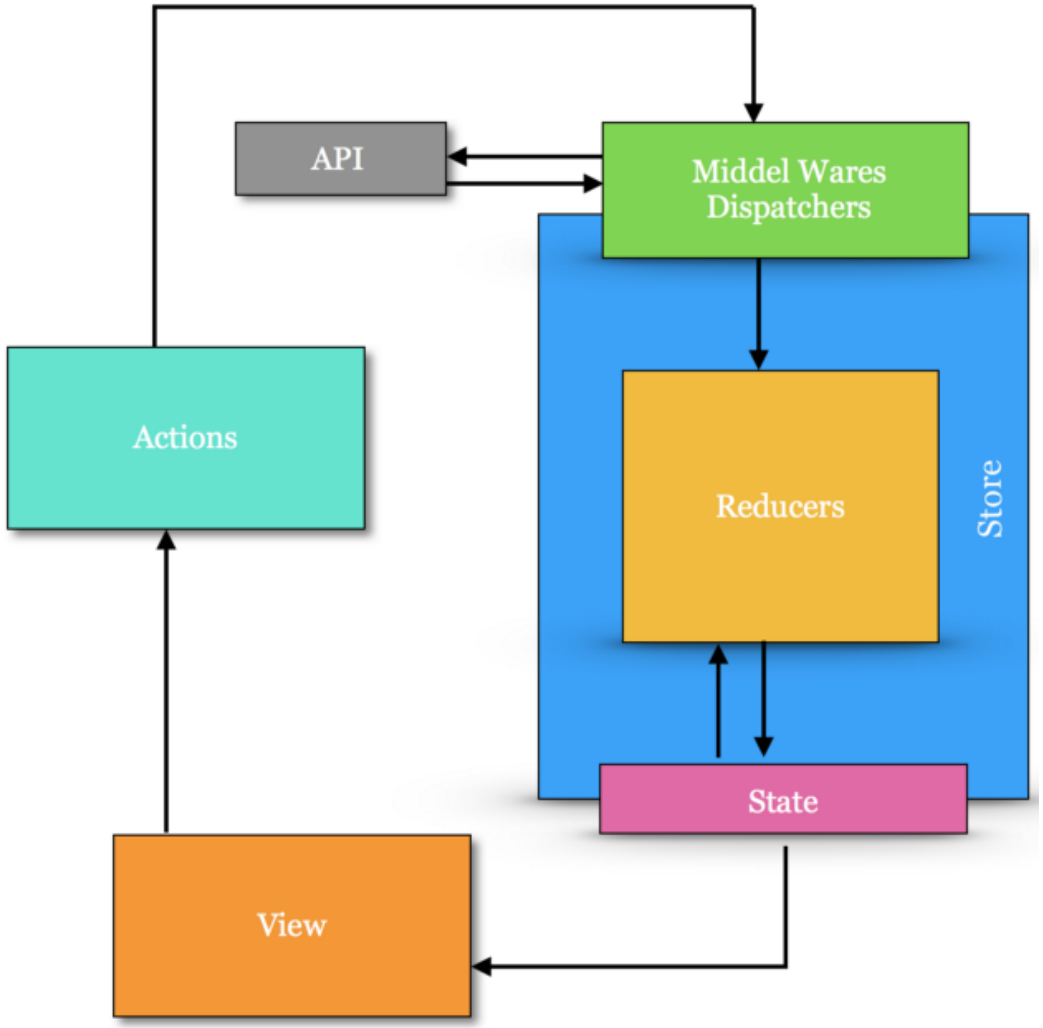


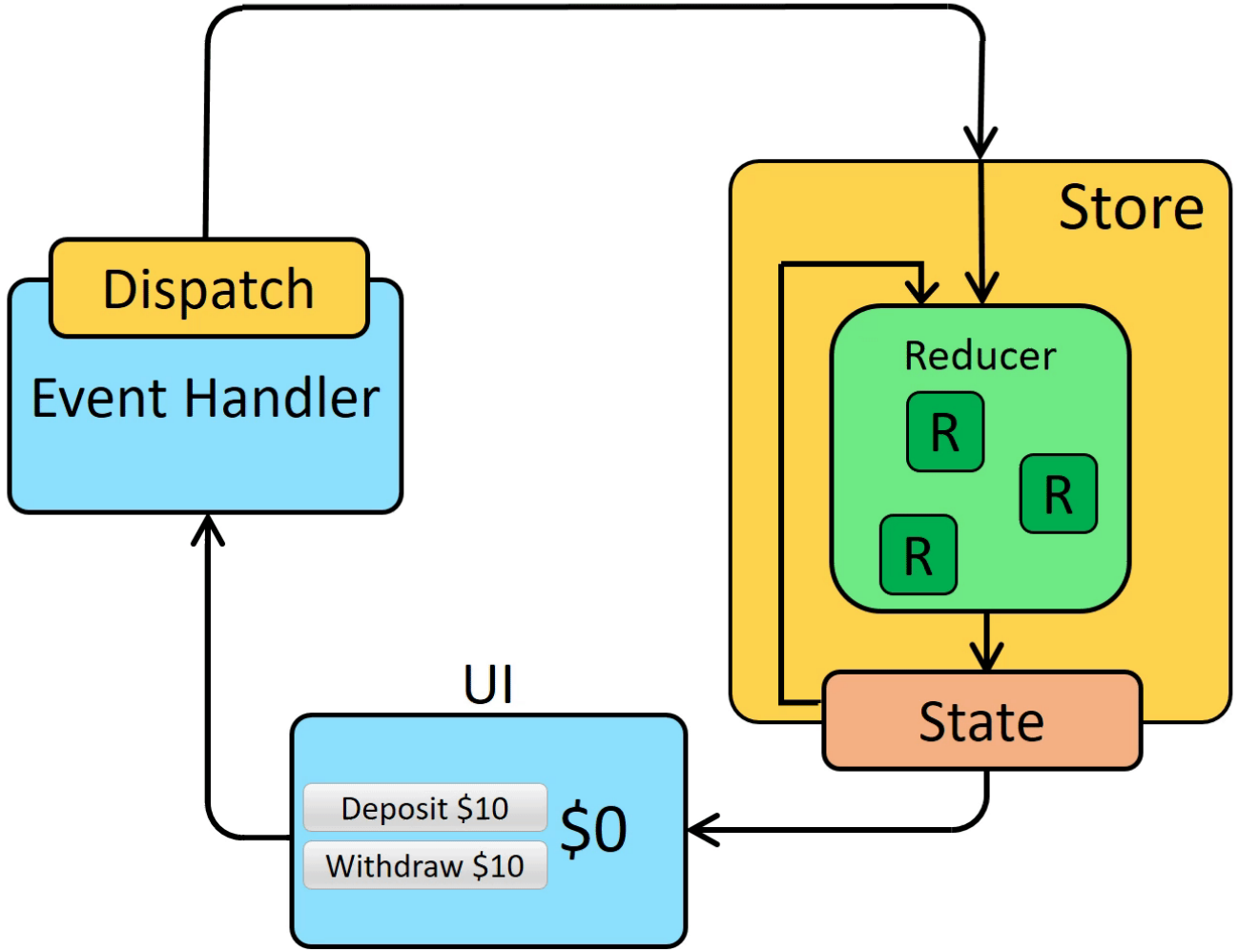
https://react.dev/community/team

Поток данных и сообщений React



React + Redux


API
Middel Wares
Dispatchers
Actions
Reducers
Store
State
View

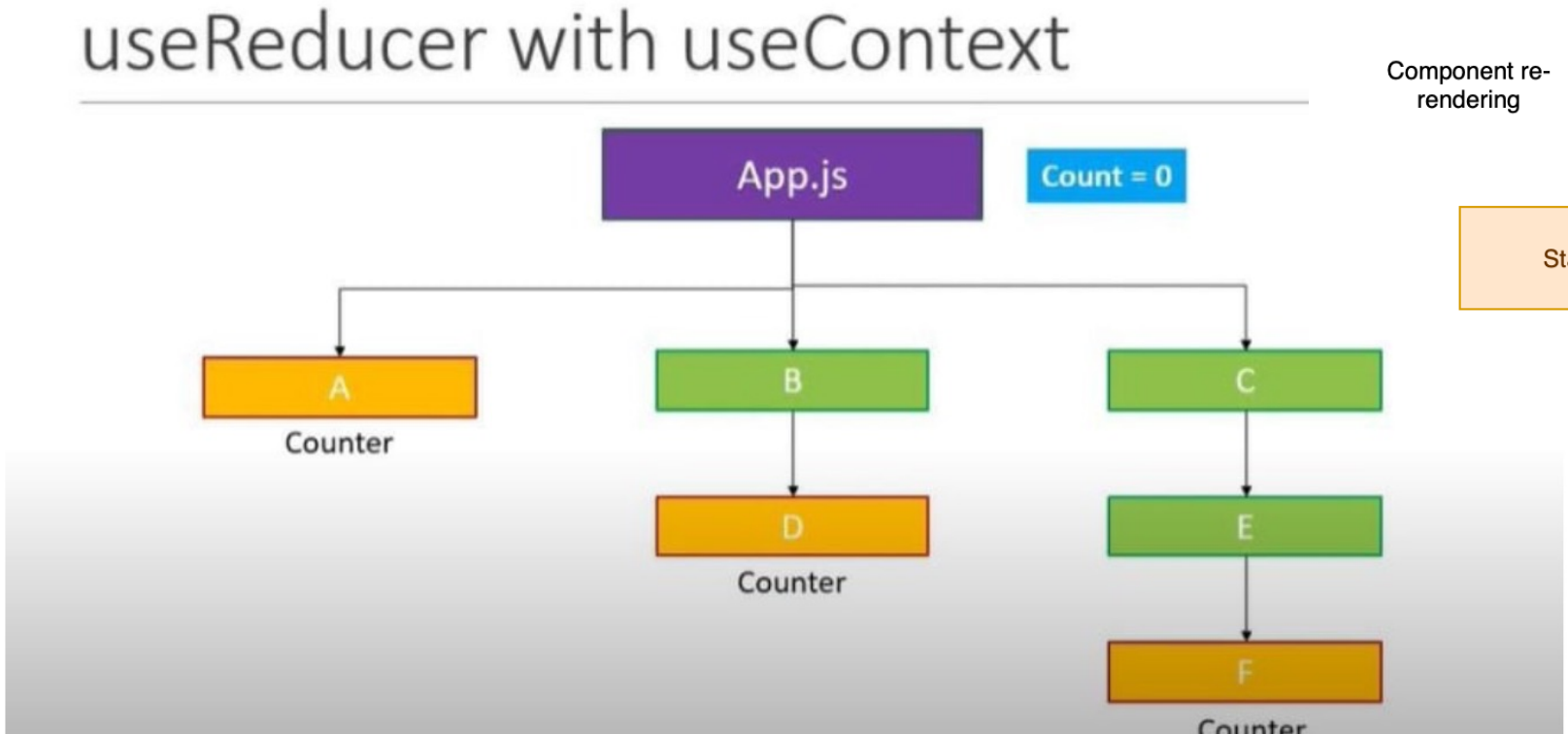

Dispatch
Event Handler
Store
Reducer
R
R
R
State
UI
Deposit $10
Withdraw $10
$0

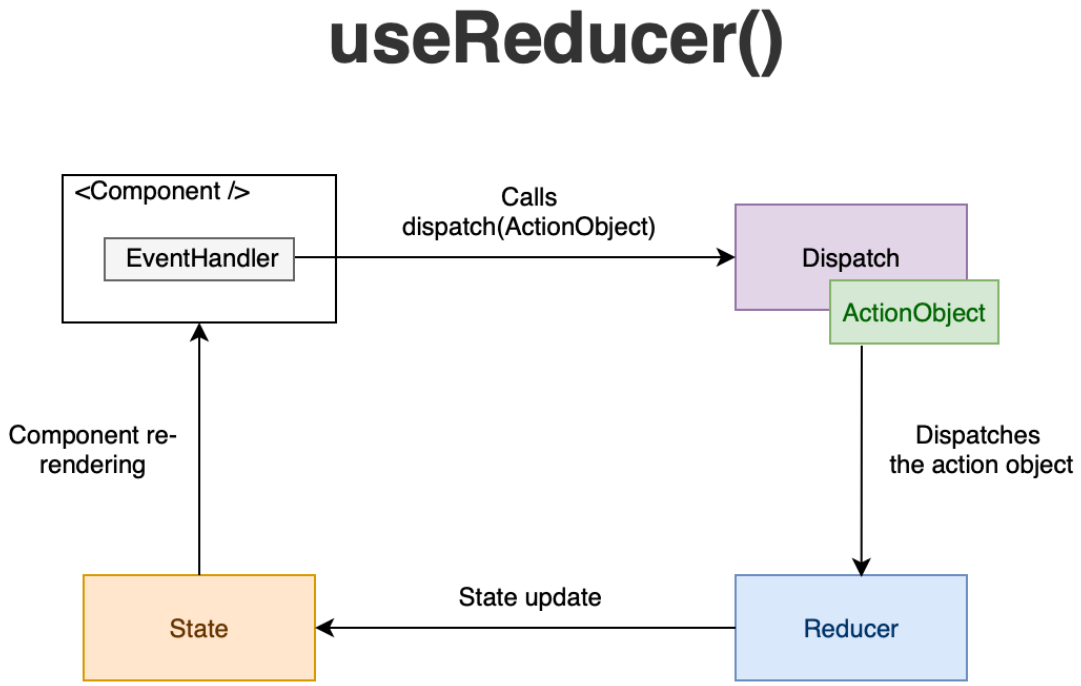
Виды Redux

- Redux классовый
- Redux функциональный через Hooks
- Redux Toolkit – мы используем его в лабораторных

- Помимо Redux есть другие контекстные менеджеры, например MobX
- Также всегда остается стандартный useContext + useReducer

useReducer + useContext





Базовый пример Redux Toolkit

- Опишем пример простого Slice
- Здесь уже все наши основные составляющие: reducer, action, initialState

```
const dataSlice = createSlice({
    name: "data",
    // в initialState мы указываем начальное состояние нашего глобального хранилища
    initialState: {
        Data: []
    },
    // Редьюсеры в слайсах мутируют состояние и ничего не возвращают наружу
    reducers: {
        setData(state, {payload}) {
            state.Data = payload
        }
    }
})

export const useData = () =>
    useSelector((state) => state.ourData.Data)


export const {
    setData: setDataAction
} = dataSlice.actions
```

Хранилище

- Создадим store
- Подключим его в наше приложение

```
import { combineReducers, configureStore } from "@reduxjs/toolkit"
import dataReducer from "./slices/dataSlice"


export default configureStore({
    reducer: combineReducers({
        ourData: dataReducer
    })
})
```

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App'
import store from "./store";
import { Provider } from "react-redux";


const root = ReactDOM.createRoot(
    document.getElementById('root')
);


root.render(
    <React.StrictMode>
        <Provider store={store}>
            <App />
        </Provider>
    </React.StrictMode>
);
```

Сделаем корзину без API

- Опишем slice
- Опишем наши данные и начальные состояние для них
- Состав корзины и ее стоимость

```
const dataSlice = createSlice({
    name: "data",
    initialState: {
        Data: [
            {
                "id": 1,
                "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",
                "price": 109,
                "category": "men's clothing",
                "rating": {
                    "rate": 3.9,
                    "count": 120
                }
            },
            {
                "id":2,
                "title":"Mens Casual Premium Slim Fit T-Shirts ",
                "price":22,
                "category":"men's clothing",
                "rating":{
                    "rate":4.1,
                    "count":259
                }
            }
        ],
        SumShoppingCart: 0,
    },
```

Reducer, action, selector

- Укажем какие action каким reducer соответствуют
- Опишем наши функции reducer для изменения данных в store – указываем варианты для изменения наших данных
- Используем useSelector, чтобы отобразить изменения из нашего store в UI

```
export const {
    setData: setDataAction,
    setSum: setSumAction,
    delSum: delSumAction
} = dataSlice.actions
```

```
reducers: {
    setData(state, {payload}) {  // изменяем состояние на полученные данные
        state.Data = payload
    },
    setSum(state, {payload}) {  // суммируем цены выбранных товаров
        state.SumShoppingCart += payload
    },
    delSum(state) {  // обнуляем сумму выбранных товаров
        state.SumShoppingCart = 0
    }
}
```

```
export const useData = () =>
    useSelector((state) => state.ourData.Data)

export const useSum = () =>
    useSelector((state) => state.ourData.SumShoppingCart)
```

Наш компонент

- useDispatch чтобы инициировать из event (события) изменение в store через action
- useSelector чтобы поймать изменение в store для render UI


```
export default function ShoppingCart(){
    const dispatch = useDispatch()
    const sum = useSum()
    const data = useData()
    return(
        <div>
            <div className="large"> Сумма заказа: { sum }</div>
            {
                data.map((good) =>
                    <div key={good.id}>
                        <p>
                        { good.title }
                        </p>
                        <p> Цена -
                            { good.price }
                        </p>
                        <button onClick={ () => {
                            dispatch(setSumAction( good.price ))
                            }}>
                            Добавить
                        </button>
                    </div>
                )
            }
            <button onClick={() => {
                dispatch(delSumAction())
            }
            }>
                Обнулить
            </button>
        </div>
    )
}
```

Добавим взаимодействие с API

- Добавим вызов axios в useEffect при монтировании страницы с нашим компонентом
- При этом мы только отправляем новые данные в store
- UI описан в отдельном файле и изменения в store в него придут через useSelector

```
import {useEffect} from "react";
import axios from "axios"
import {setDataAction} from "./slices/dataSlice";
import {useDispatch} from "react-redux";

export function GetData() {
    const dispatch = useDispatch()
    async function fetchData() {
        const response = await axios.get('https://fakestoreapi.com/products?limit=5') // получение данных с API
        dispatch(setDataAction(response.data))
    }
    useEffect(() => {
        fetchData()
    }, [])
}
```