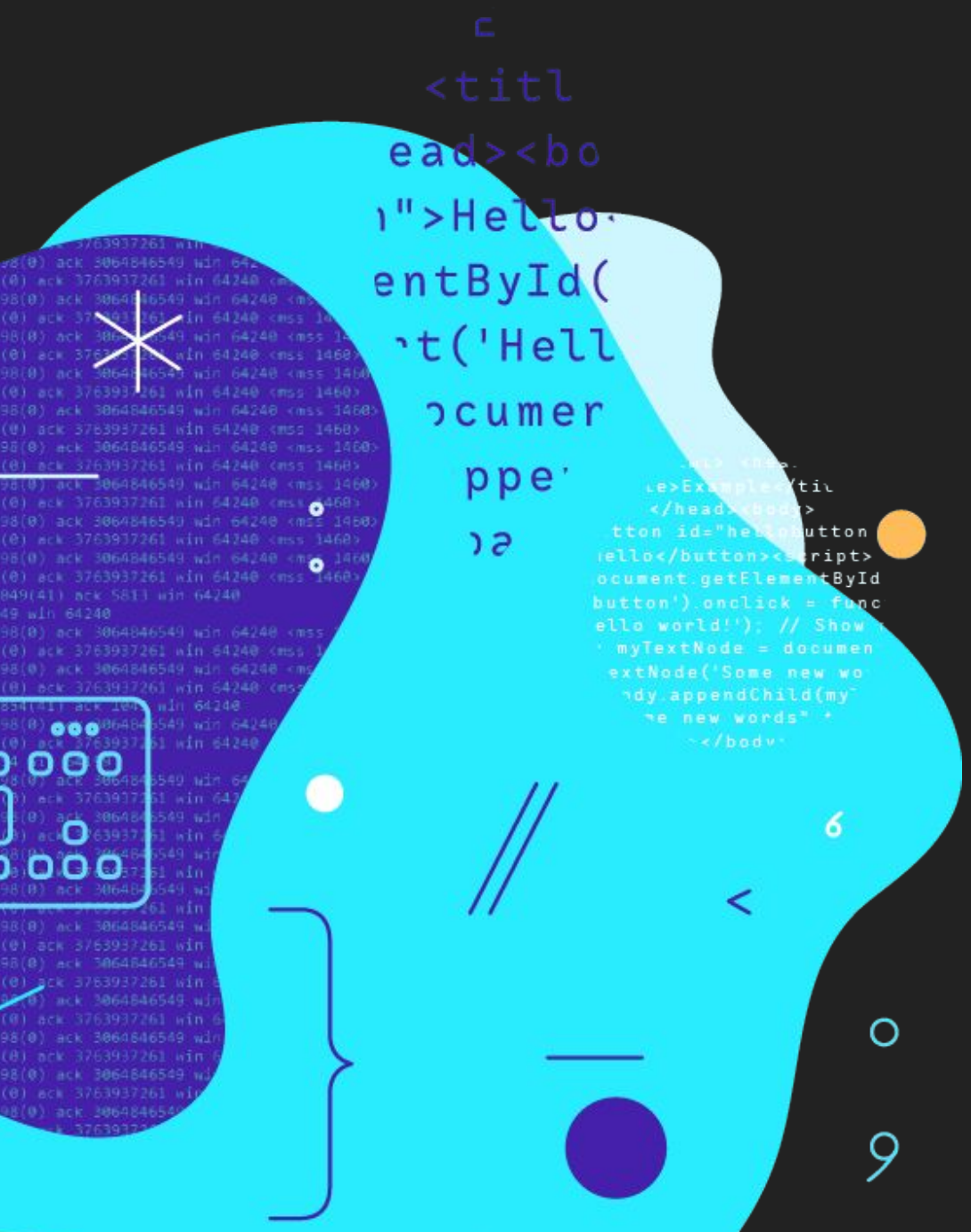


PHASE 3
WEEK 1

DAY 4

ELBRUS/BOOTCAMP

План

1. Redux, Three Principles
2. React Redux, React Redux Hooks
3. Action Types & Action Creators & Root Reducer
4. Redux DevTools

Redux

Redux

Redux — это предсказуемый контейнер состояния для JavaScript приложений.

Состояние системы — это сохранённая информация о предыдущих событиях или действиях пользователя.

npm install redux

Redux vs. useReducer vs. useState

`useState` — хук React для работы с состоянием одного компонента.

`useReducer` — хук React для работы с централизованным состоянием.

`Redux` — отдельная библиотека для работы с централизованным состоянием приложения. Позволяет держать бизнес-логику приложения отдельно от React.

Немного теории

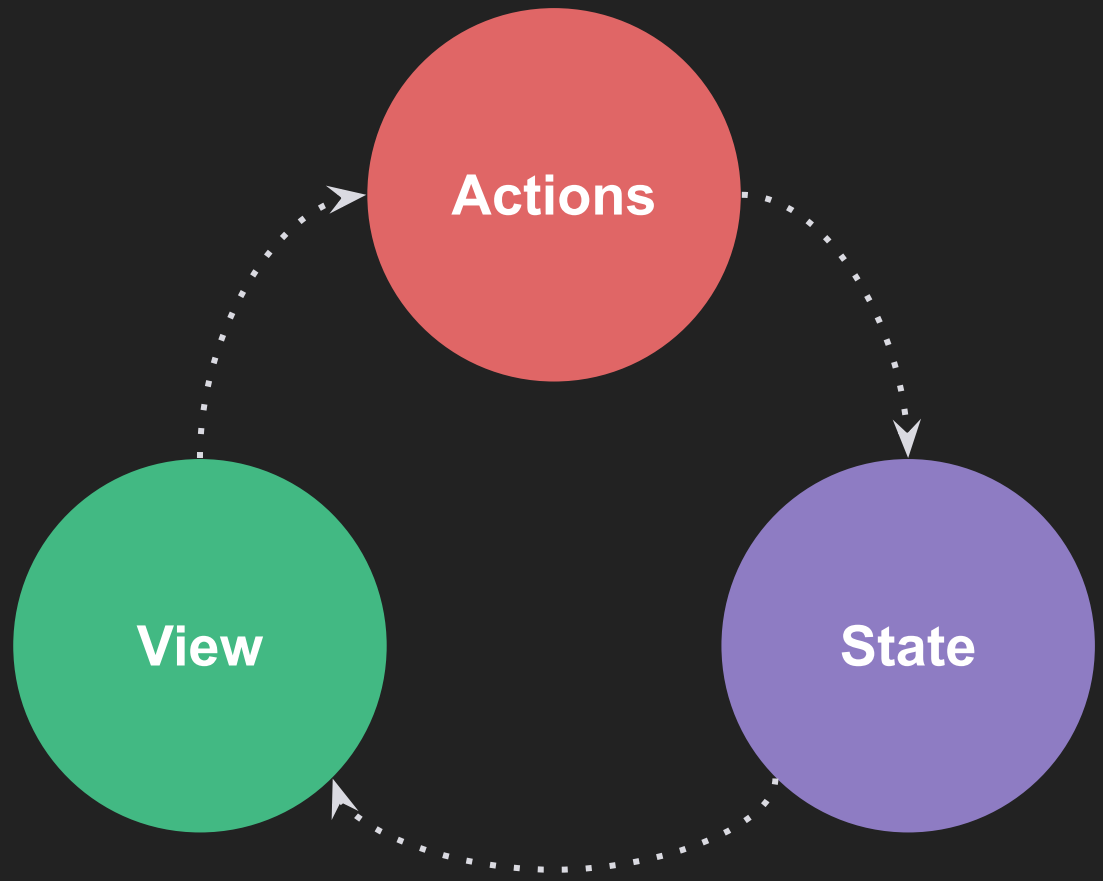
`Redux` — реализация паттерна программирования `Flux`.

`Паттерн программирования` — типовое решение часто встречающейся проблемы.

`Паттерн Flux` — одно из решений проблемы отрисовки UI согласно меняющимся данным.

Однонаправленный поток данных

Однонаправленный поток данных (one way data flow) — основная идея паттерна Flux и библиотеки Redux.



Store & State, хранилище и состояния

```
// функция для создания store, контейнера состояний
import { createStore } from 'redux';

// функция reducer с описанием действий, где state по-умолчанию 0
const reducer = (state = 0, action: Action): number => {
 switch (action.type) {
   case 'counter/increment':
     return state + 1;
   case 'counter/decrement':
     return state - 1;
 }
 return state;
}

const store = createStore(reducer); // формирование store
console.log(store.getState()); // вывод текущего состояния в консоль
```

Action

- Чтобы изменить state, нужно выполнить действие (action).
- Action — это объект, у которого есть ключ type, payload (опционально) и error (опционально)
- Action ничего не делает — только описывает, что нужно сделать.Саму работу совершает reducer.
- Чтобы выполнить action, нужно его отправить в store
- Инициатор действия — dispatch

```
// Тип действия
type Action =
 | { type: 'counter/increment' }
 | { type: 'counter/decrement' };

// описать действие (опционально)
const action: Action = {
 type: 'counter/increment',
};

// отправить действие
store.dispatch(action);
```

Reducer, example code

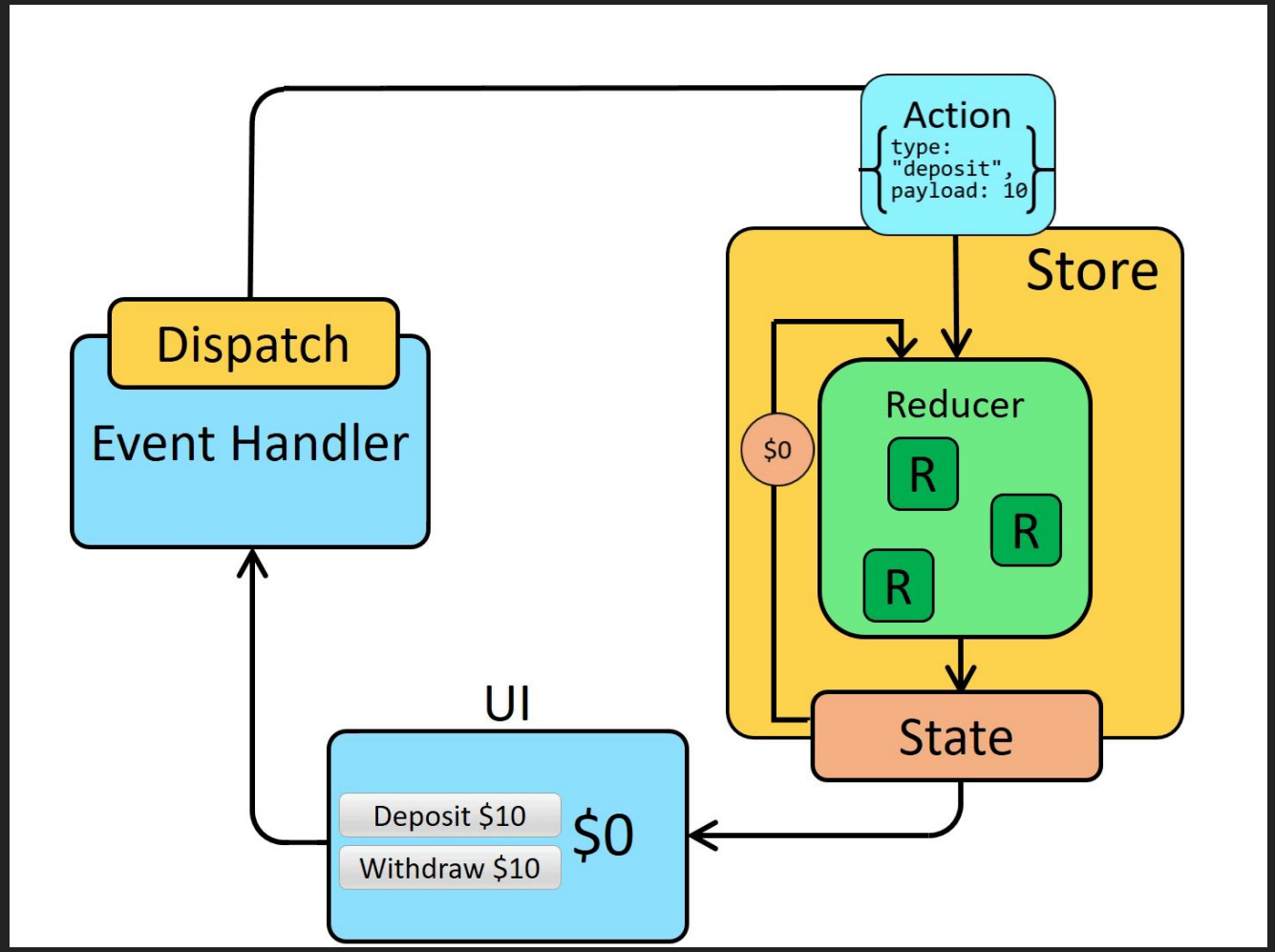
```ts
// types/Student.ts
export default interface Student {
 id: string;
 name: string;
 age: number;
}

// types/State.ts
export default interface State {
 title: string;
 students: Student[];
}

// types/Action.ts
type Action = {
 type: 'students/updateStudent';
 payload: Partial<Student>;
};
export default Action;
```

```ts
// reducer.ts
const studentReducer = (
 state: State = {
   title: 'Students',
   students: [],
 },
 action: Action
): State => {
 switch (action.type) {
   case 'students/updateStudent':
     return {
       ...state,
       students: [...state.students].map((student) => {
         if (student.id === action.payload.id) {
           return {
             ...student,
             ...action.payload,
           };
         }
         return student;
       }),
     };
   default: return state;
 }
 };
```

Data flow in Redux



См. анимацию и пояснения здесь:

https://redux.js.org/tutorials/essentials/part-1-overview-concepts#redux-application-data-flow

Three Principles

Three principles, три принципа

Single source of truth — store всегда один и он является источником истины всего приложения.

State is read-only — нельзя изменять state напрямую, для этого нужно вызвать action.

Reducer is a pure function — повторные вызовы редьюсера с одними и теми же аргументами возвращают одно и то же. Текущий state не мутируется, а клонируется (copy).

React Redux

React Redux

`React Redux` — официальный пакет сборка под React, имеющий свои хуки.

`React Redux` позволяет вашим компонентам React считывать данные из хранилища Redux и отправлять действия в хранилище для обновления состояния.

```
npm install react-redux
```

React Redux: Provider

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

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root'),
);
```

React Redux Hooks: useSelector

useSelector

Хук из React Redux для доступа к состоянию или его ключам.

```
const contactsSelector = (state: State): Contact[] => state.contacts;
const contacts = useSelector(contactsSelector);
```

React Redux Hooks: useDispatch

useDispatch

Хук для доступа к методу dispatch.

```
const dispatch = useDispatch()
```

useSelector & useDispatch, example code

```
import { useSelector, useDispatch } from 'react-redux';
import { selectUser } from '../selectors';

export default function List(): JSX.Element {
 const dispatch = useDispatch();
 const user = useSelector(selectUser);

 return (
   <div className="user-card">
     <div>Name: ${user.name}</div>
     <div>Age: ${user.age}</div>
     <button
       type="button"
       onClick={() => dispatch({ type: 'users/removeUser', payload: user.id })}
     >
       remove
     </button>
   </div>
 );
}
```

Store, scheme of work

Store, example code

```
import { createStore } from 'redux';
import { composeWithDevTools } from 'redux-devtools-extension';
import reducer from './rootReducer';

// формирование контейнера состояний (store)
const store = createStore(reducer,
    // эта функция нужна для того, чтобы отлаживать Redux через панель инструментов
разработчика
    composeWithDevTools()
);

// экспортируем контейнер
export default store;

// RootState - это тип нашего глобального состояния
export type RootState = ReturnType<typeof store.getState>;
```

Action Types
Action Creators
Root Reducer

Actions Type

Типы для `action` мы представляем в виде union-типа (соединяем через “или”):

```ts
// types/Action.ts
type Action =
  | { type: 'users/addUser'; payload: User }
  | { type: 'users/deleteUser'; payload: UserId }
  | { type: 'users/updateUser'; payload: Partial<User> };
```

Action Types, example code

```ts
// reducer.ts
import State from './types/State.ts';
import Action from './types/Action.ts';

export default (
 state: State = {},
 action: Action
) => {
 switch (action.type) {
   case 'users/addUser':
   // ...
 }
 return state;
};
```

```ts
// SomeComponent.tsx
import { useDispatch } from
'react-redux';

const dispatch = useDispatch();

dispatch({
  type: 'users/addUser',
  payload: data,
});
```

Action Creators, example code

```ts
// actionCreators.ts
import Action from 'types/Action.ts';

export const addUser = (user: User):
Action => ({
 type: 'users/addUser',
 payload: user,
});
```

```ts
// использование AC
import { addUser } from
'./actionCreators.ts';

dispatch(addUser(user));
```

Root Reducer, example code

```ts
// rootReducer.ts
import { combineReducers } from
'redux';
import clientReducer from
'./features/client/reducer';
import userReducer from
'./features/user/reducer;

const rootReducer = combineReducers({
 clients: clientReducer,
 users: userReducer,
});

export default rootReducer;
```

```ts
// store.ts
import { createStore } from 'redux';
import { composeWithDevTools } from
'redux-devtools-extension';
import rootReducer from './rootReducer';

const store = createStore(rootReducer,
composeWithDevTools());

export default store;

export type RootState = ReturnType<typeof
store.getState>;
```

Redux DevTools

Redux DevTools

Инструмент, упрощающий разработку с Redux.

Для работы с Redux DevTools:

1. установите расширение для Chrome:
   https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=en
2. подключите NPM-пакет:
   https://www.npmjs.com/package/redux-devtools-extension

Установка: `npm install -D @redux-devtools/extension`

Redux DevTools — подключение пакета

```
import { createStore } from 'redux';
import { composeWithDevTools } from 'redux-devtools-extension';
import reducer from './rootReducer';


const store = createStore(
   reducer,
   composeWithDevTools() // подключение Redux DevTools
);

export default store;

export type RootState = ReturnType<typeof store.getState>;
```

Альтернативы Redux

- MobX
  https://mobx.js.org/README.html
  https://habr.com/ru/post/471048/

- Recoil JS
  https://recoiljs.org/

- React useContext + useReducer
  https://habr.com/ru/post/473070/