СБЕРБАНК ТЕХНОЛОГИИ

ES6 + ReactJS

https://goo.gl/dq8EoO

Где резюме?

СБЕРБАНК ТЕХНОЛОГИИ

Ударники труда:

1) Малеванный
2) Истамов
3) Чернявский
4) Комягин
5) Коджаев
6) Байда
7) Жмурин
8) Нешин
9) Смолянинова
10) Плешаков

Мои вопросы

СБЕРБАНК ТЕХНОЛОГИИ

1) Почему вы решили учить JAVA?
2) Ещё немного потрепаться про популярность языков и опенсорс?
3) Кто знает идеальную модель взаимодействия бэка и фронта?
4) Как выглядит ваше апи?
5) Почему я задаю эти вопросы?
6) Почему я решил выучить JS?

План занятия

СБЕРБАНК ТЕХНОЛОГИИ

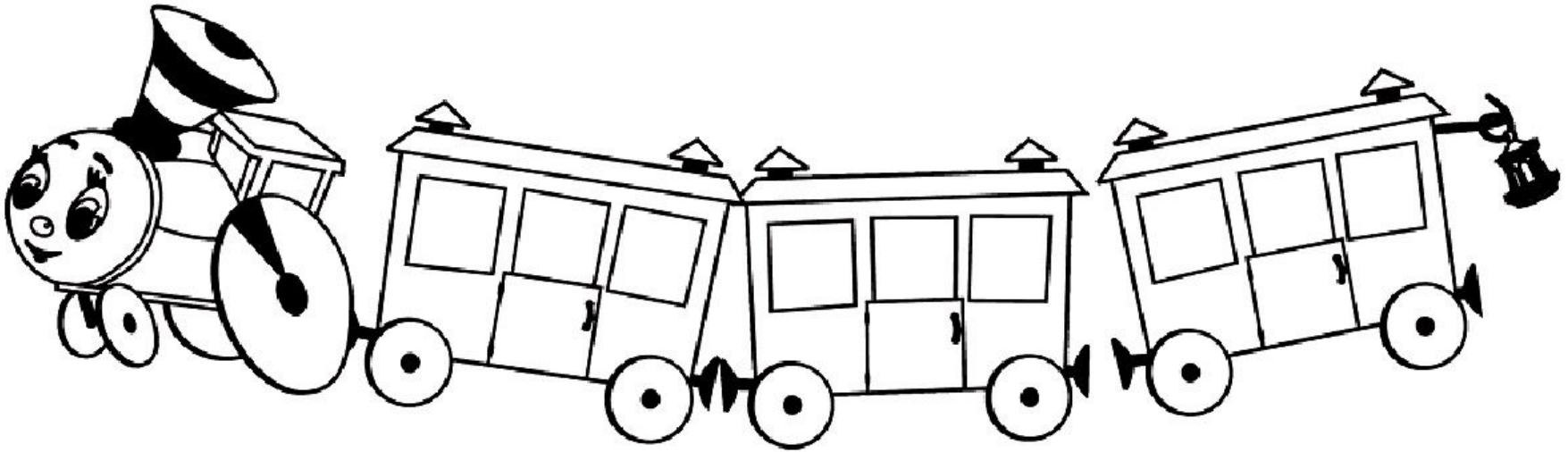
1) Асинхронность и параллельность
2) Асинхронность и событийность
3) Сборщики
4) React + coding
5) Чистые функции

DevOps

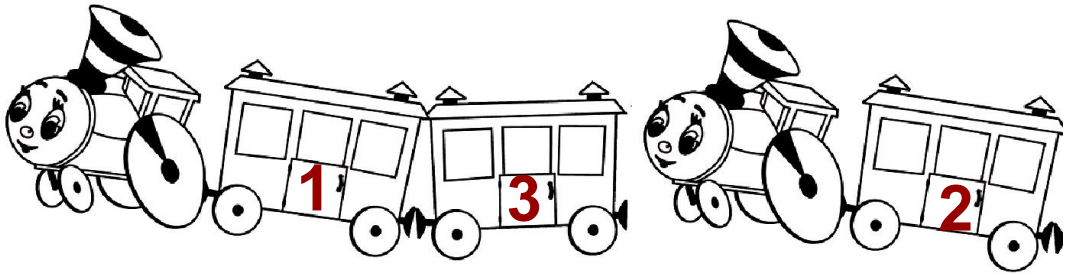
СБЕРБАНК ТЕХНОЛОГИИ

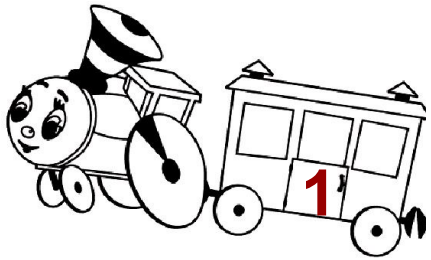
Асинхронность и параллельность

СБЕРБАНК ТЕХНОЛОГИИ



Асинхронность и параллельность





Событийность и асинхронность

СБЕРБАНК ТЕХНОЛОГИИ

Событийность и асинхронность

СБЕРБАНК ТЕХНОЛОГИИ

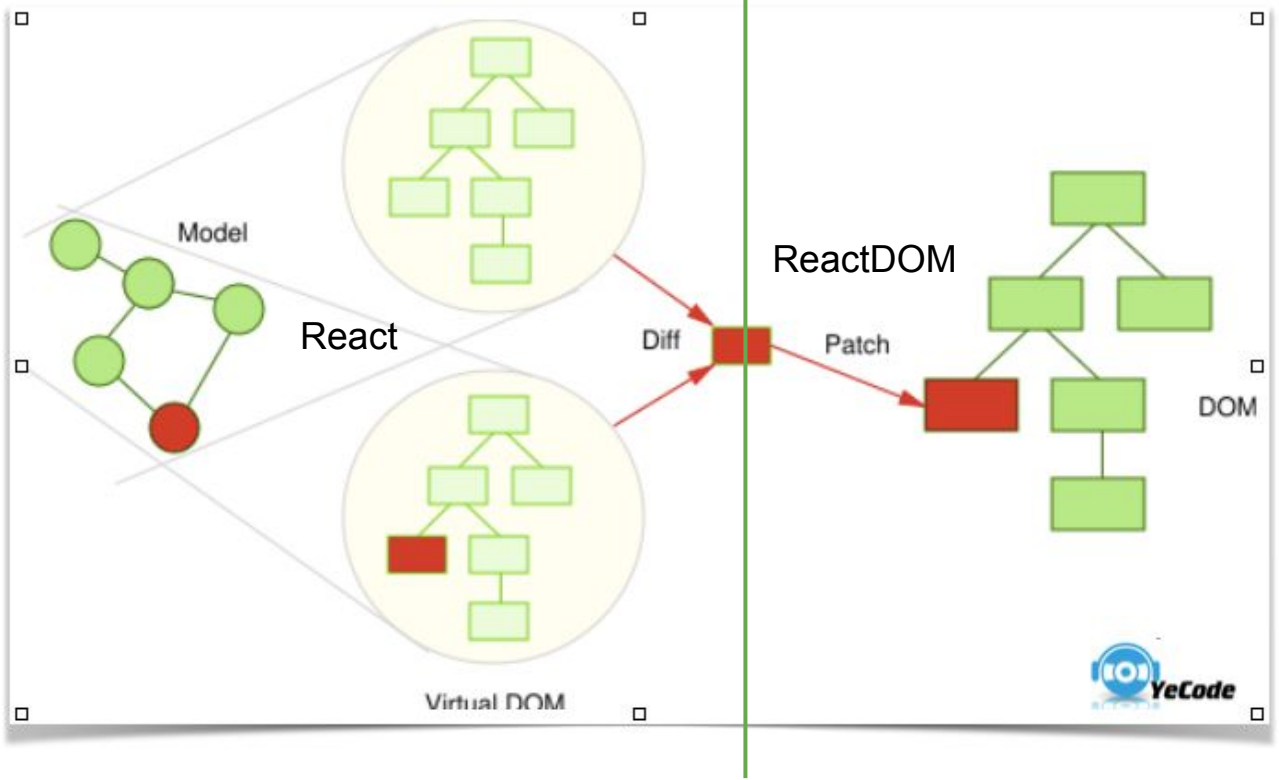
Шу Ха Ри

СБЕРБАНК ТЕХНОЛОГИИ

React и ReactDOM

Объединяй

и властвуй



Сборка фронтенда

СБЕРБАНК ТЕХНОЛОГИИ

0. Set Up a Webpack Development Environment

npm init

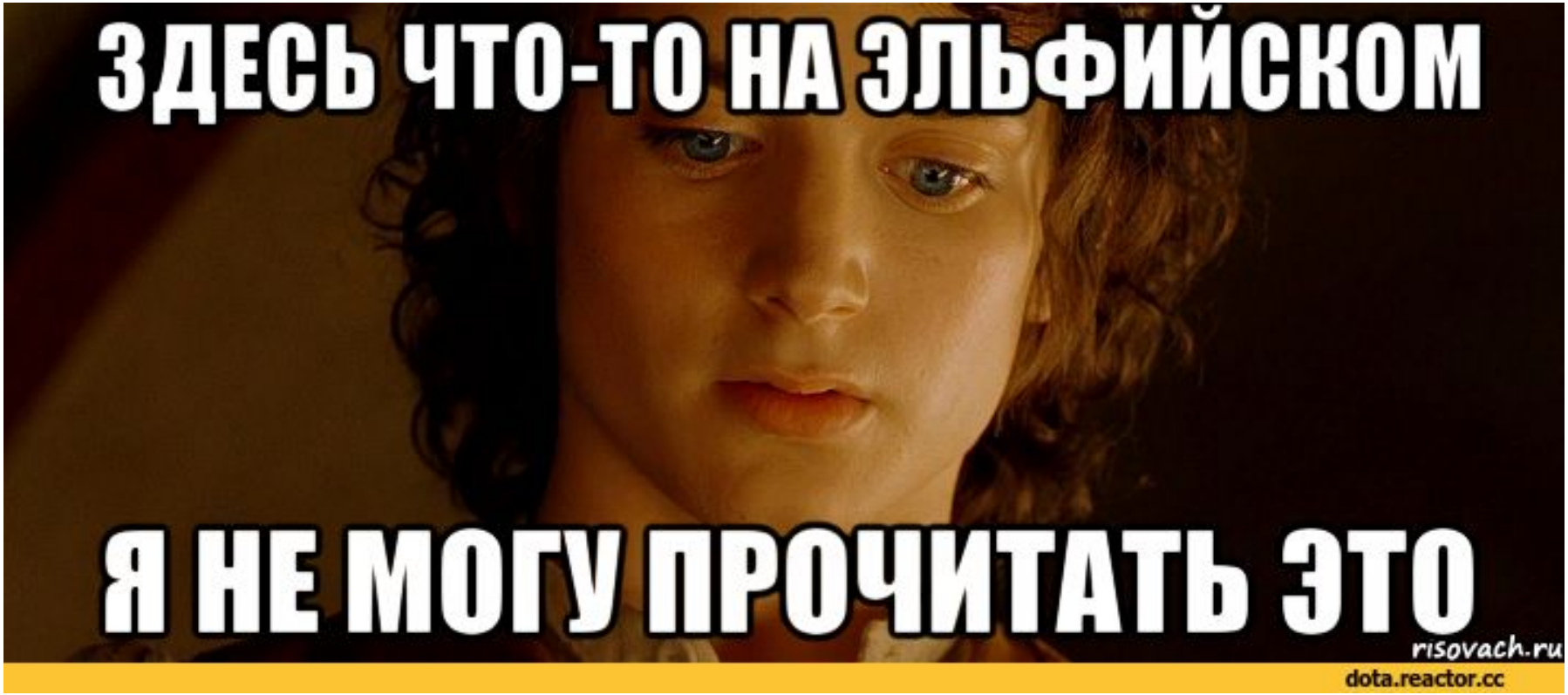
npm install --save-dev babel-core babel-preset-es2015 babel-preset-react

npm install --save-dev react react-dom

npm install --save-dev webpack webpack-dev-server

Дальше будет на эльфийском

СБЕРБАНК ТЕХНОЛОГИИ



Подключение реакт

СБЕРБАНК ТЕХНОЛОГИИ

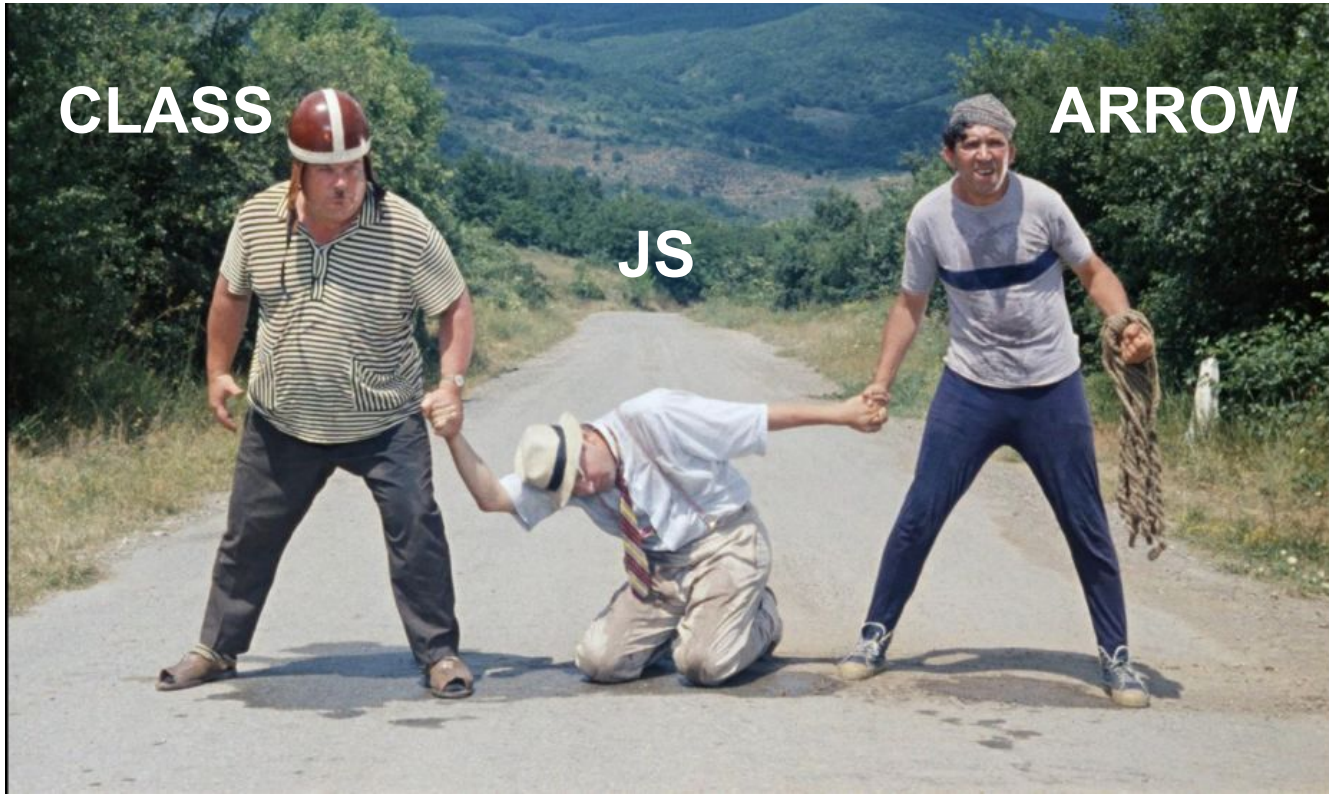
```
<!DOCTYPE html>
<head>
  <title>School</title>
</head>
<body>
  <div id="app"></div>
  <!-- вместо index подключим сборку -->
  <script src="index.js"></script>
</body>
</html>
```

```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(<div>Hello</div>,
document.getElementById('app'));
```

1. Все любят троицу

СБЕРБАНК ТЕХНОЛОГИИ



1. Виды компонент

СБЕРБАНК ТЕХНОЛОГИИ

```
// class
class App extends
React {
 render() {
   return (
     <div>Hi World</div>
   )
 }
}
```

```
//origin js
const App = function
App() {
 return
React.createElement(
   'div',
   null,
   'Hi World'
 );
};
```

```
// arrow
const App = () => (
 <div>Hi World</div>
);
```

СБЕРБАНК ТЕХНОЛОГИИ

Первое правило React

Не возвращайте более одного тега из реакт компонента!

2. Типы атрибутов и параметры по умолчанию

СБЕРБАНК ТЕХНОЛОГИИ

```
// arrow
const App = (props) => (
 <div>Hi {props.target}</div>
);

App.propTypes = {
 target: React.PropTypes.string.isRequired
};

export default App;
```

3. Состояния компонента и обновление

СБЕРБАНК ТЕХНОЛОГИИ

```
class App extends Component {
 constructor(props) {
   super(props);
   this.state = { count: 0 };
   this.update = this.update.bind(this);
 }
 update() {
   this.setState({ count: this.state.count + 1 });
 }
render() {...}
```

4. Внутренние компоненты

СБЕРБАНК ТЕХНОЛОГИИ

```
const Widget = (props) => (
   <div>Hi {props.target} {props.message}</div>
);
const App = () => (
 <div>
   <Widget ... />
   <Widget ... />
   <Widget ... />
</div>
);
```

5. Связь компонентов и DOM

СБЕРБАНК ТЕХНОЛОГИИ

```
update(e) {
 this.setState({ width: ReactDOM.findDOMNode(this.refs.widget).offsetWidth })
}

render() {
 return (
    <div ref="widget">{this.state.width}</div>
 );
}
```

! Плохой пример в eggheads !

6. Работа с детьми

СБЕРБАНК ТЕХНОЛОГИИ

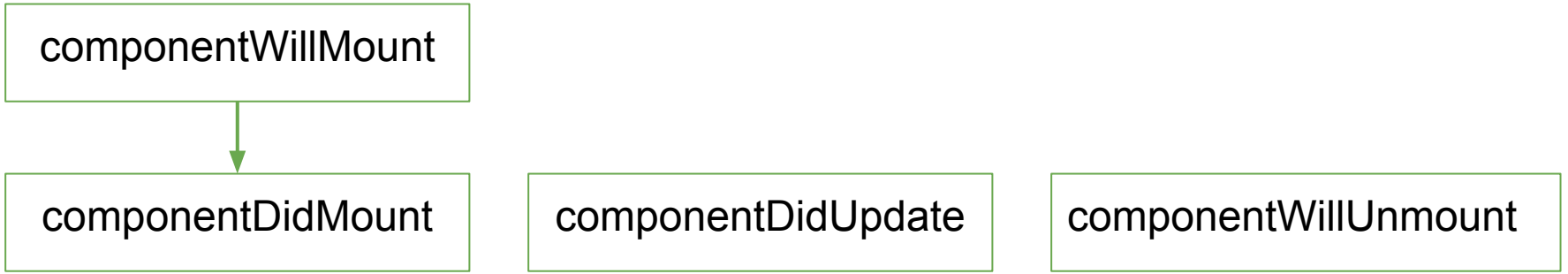
```
const Widget = (props) => (
   <div style={{color:’red’}}>{props.children}</div>
);

const App = (props) => (
  <Widget>
    <div>Hi {props.target} {props.text}</div>
    <div>bye{props.target} {props.text}</div>
  </Widget>
 );
```

Жизненный цикл компонент

7. Жизненный цикл простой

СБЕРБАНК ТЕХНОЛОГИИ



7. Жизненный цикл простой

СБЕРБАНК ТЕХНОЛОГИИ

```
class Widget extends Component {
 componentWillMount(){ console.log('перед добавлением в DOM') }
 componentDidMount(){ console.log('после добавления в DOM') }
 componentDidUpdate(){ console.log('после обновления')}
 componentWillUnmount(){ console.log('перед удалением из DOM') }

render() {
   return ( <div>HI</div>);
 }
}
```

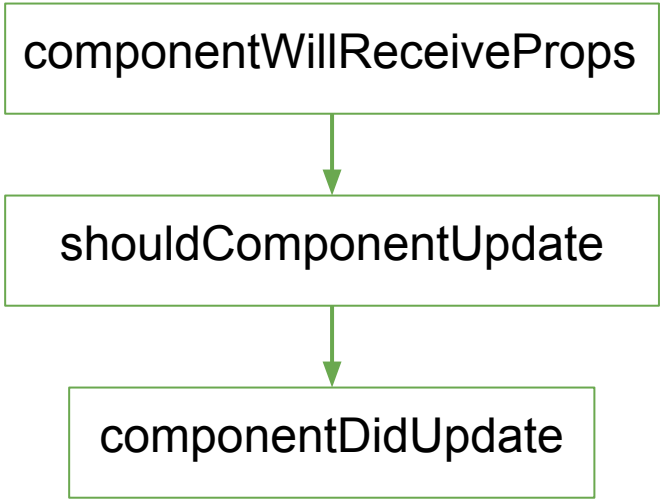
СБЕРБАНК ТЕХНОЛОГИИ

Второе правило React

Не забываю убирать листенеров

8. Жизненный цикл расширенный

СБЕРБАНК ТЕХНОЛОГИИ



8. Жизненный цикл расширенный

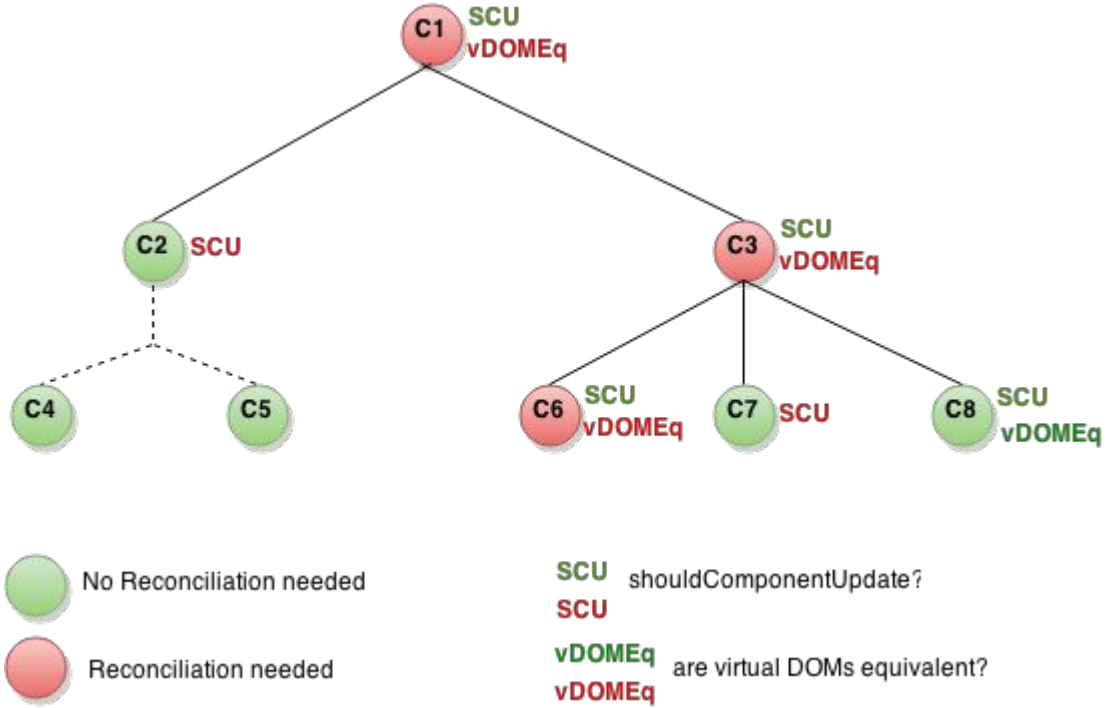
СБЕРБАНК ТЕХНОЛОГИИ

```
class Widget extends Component {
 componentWillReceiveProps(){ console.log('обработка props') }
 shouldComponentUpdate(){ console.log('отменяет render'); return true; }

render() {
   return ( <div>Hl</div>);
 }
}
```

8. SCU

СБЕРБАНК ТЕХНОЛОГИИ

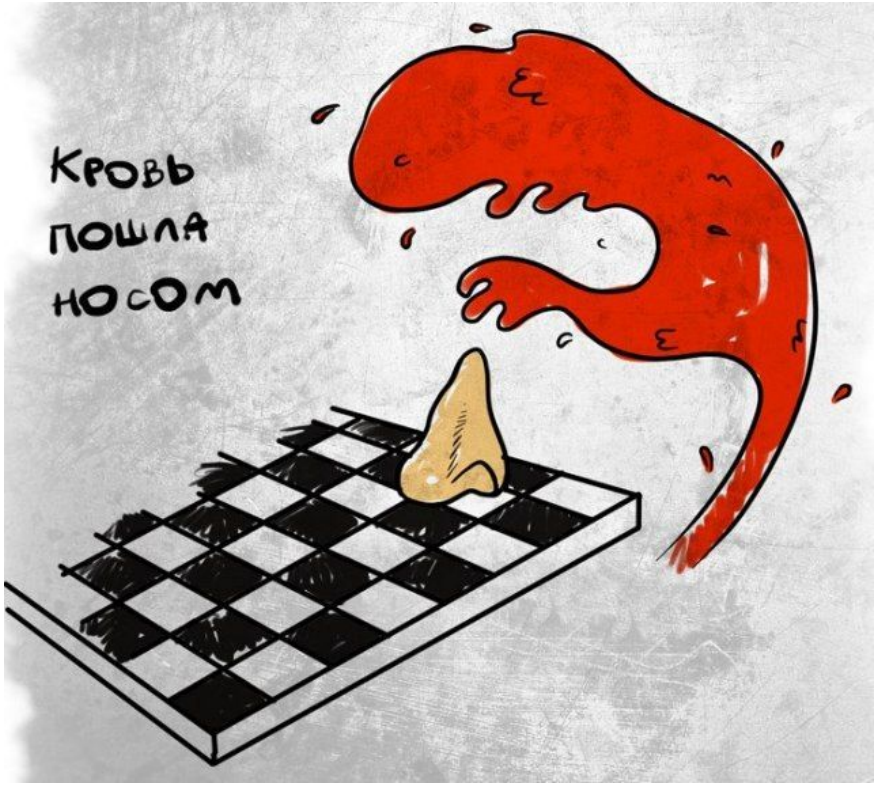


СБЕРБАНК ТЕХНОЛОГИИ

Третье правило React

Не забывай про о SCU

9. Зачем нам НОС



9. Зачем нам High Order Component

СБЕРБАНК ТЕХНОЛОГИИ

```
let Widget = (props) => (
   <div style={{color:’red’}}>{props.children}</div>
);
function greenHOC(WrappedComponent) {
 return (props) => (
   <div style={{backgroundColor: 'green'}}>
     <WrappedComponent {...props}/>
   </div>
 )
}
Widget = greenHOC(Widget);
```

10. Универсальные компоненты

СБЕРБАНК ТЕХНОЛОГИИ

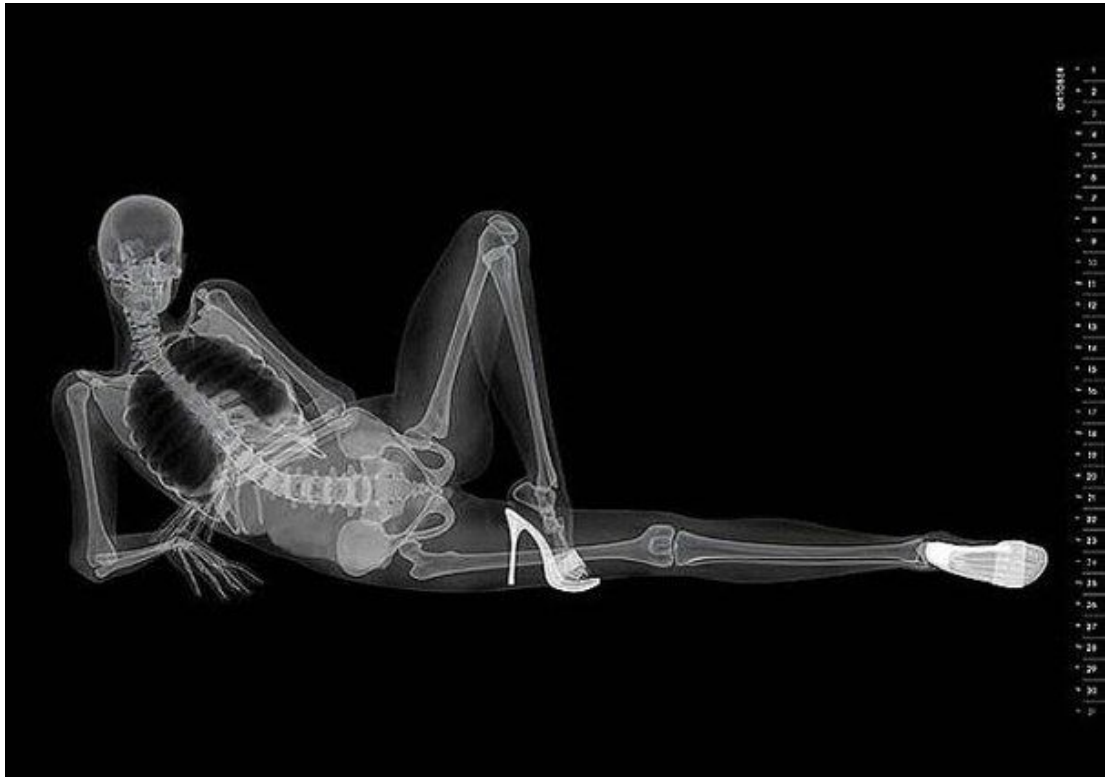
11. Отображение динамических элементов

СБЕРБАНК ТЕХНОЛОГИИ

```
const Widget = (props) => (
 <div >
  {
    props.colors.map((item) => (
      <div key={item} style={{color: item}}/>
    ))
  }
 </div>
);
<Widget colors={[‘red’, ‘green’, ‘blue’]} />
```

12. Что внутри ES5?

СБЕРБАНК
ТЕХНОЛОГИИ



13. Дебаг в Хроме

СБЕРБАНК ТЕХНОЛОГИИ

React Developer Tools

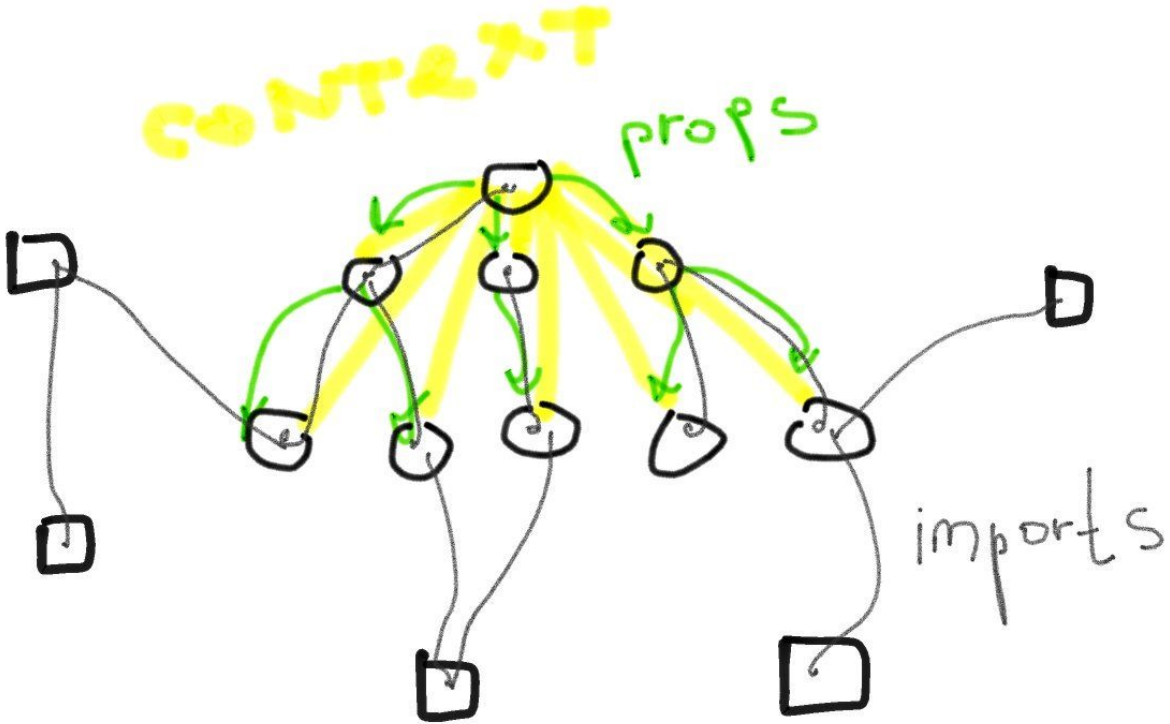
14. Axios

СБЕРБАНК ТЕХНОЛОГИИ

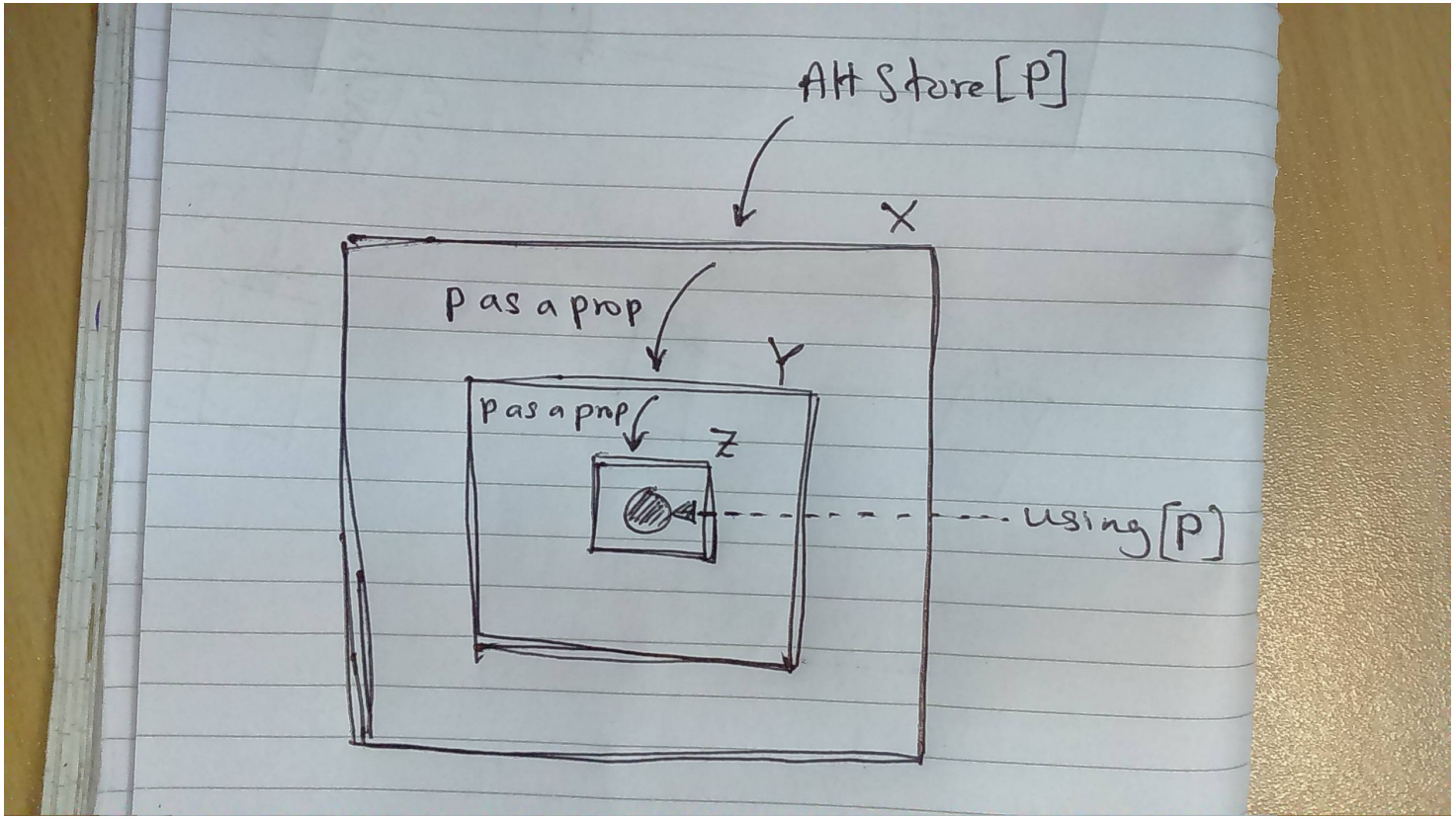
npm install --save axios@0.8.0

```
const root = 'https://jsonplaceholder.typicode.com';
axios.get(root + '/posts/1')
  .then((response) => {
    this.setState({ text: response.data.title });
  }).catch((error) => {
    this.setState({ text: error.message});
  });
```

15. Контекст



15. Контекст



15. Контекст

СБЕРБАНК ТЕХНОЛОГИИ

```
class App extends Component {
  getChildContext() {
   return {color: "white"};
  }
}

App.childContextTypes = {
 color: React.PropTypes.string
};
```

```
const Widget = (props, context) => (
  <div style={{ color: context.color}} >
    Hello
  </div>
);
Widget.contextTypes = {
 color: React.PropTypes.string
};
```

СБЕРБАНК ТЕХНОЛОГИИ

Пример рефакторинга

Когда не нужен “классный” компонент

Вопросы?

СБЕРБАНК
ТЕХНОЛОГИИ

Задание

СБЕРБАНК ТЕХНОЛОГИИ

Написать взаимодействие с рестом

1. componentWillMount + axios.get получаем данные
2. render - отображаем данные (изменяемые в input)
3. компонент 1 onClick - получаем новые данные
4. компонент2 onClick - собираем данные из input и отправляем ajax.post

Чистые функции - почему?

СБЕРБАНК ТЕХНОЛОГИИ

```
function sum(x, y) {
 return x + y;
}
```

```
function add(y) {
 return this.x + y;
}
```

Предсказуемость

Независимость

Простота тестирования

Распараллеливание

Мои вопросы

СБЕРБАНК ТЕХНОЛОГИИ

1. Где flexbox?
2. Масштабируемость

Что дальше?

СБЕРБАНК ТЕХНОЛОГИИ

- ESLint
- ?PostCSS
- FlexBox или MaterialUI
- ReactRouter
- Redux
- TimeMachine
- Тесты
- Документирование