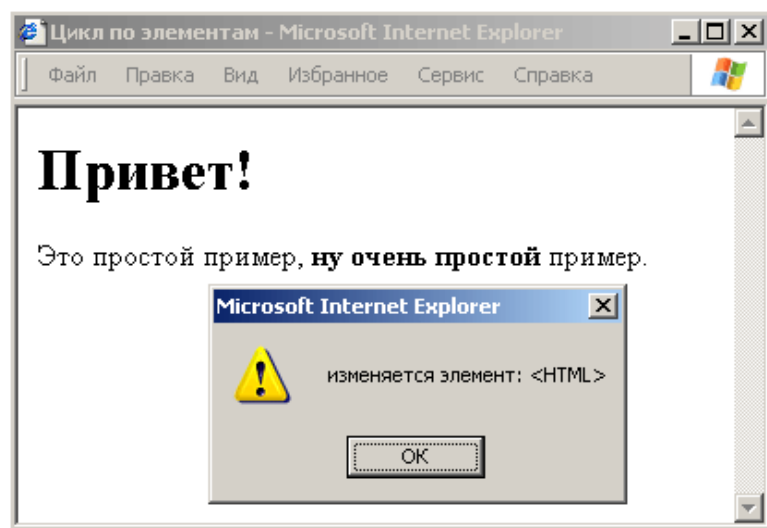


React

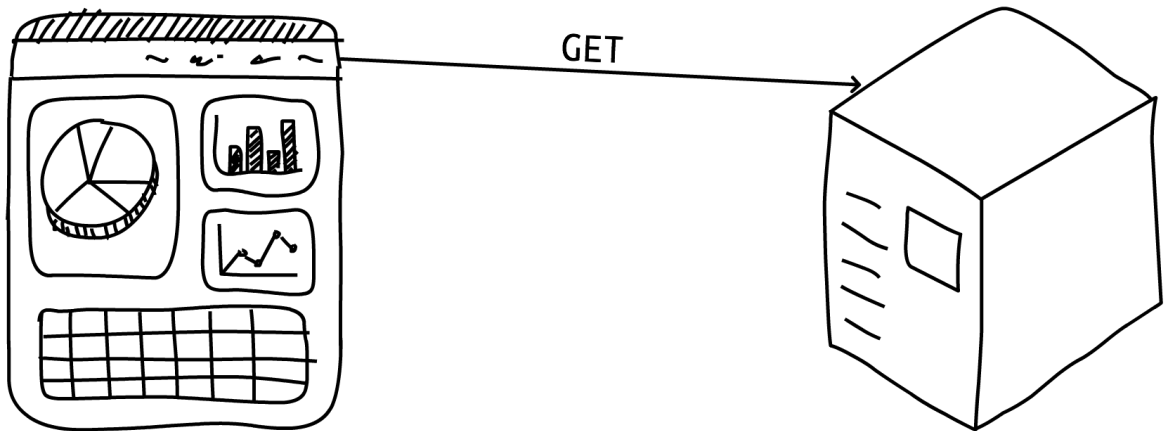
Немного истории

Эволюция задач перед JS

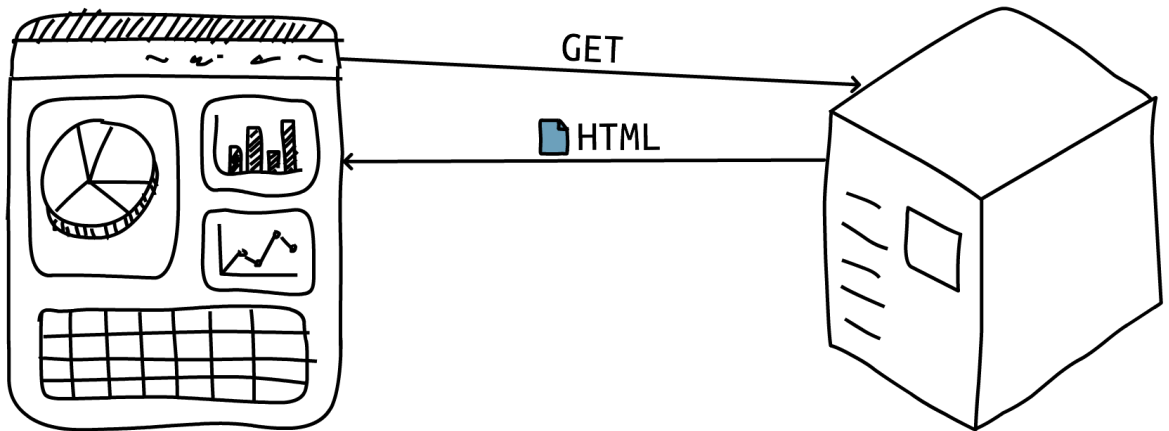
Простые задачи



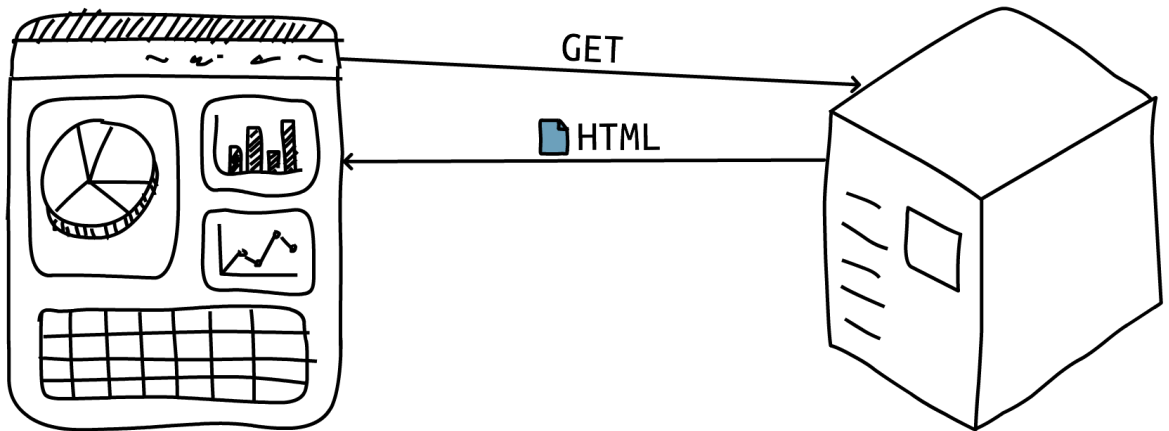
Взаимодействие



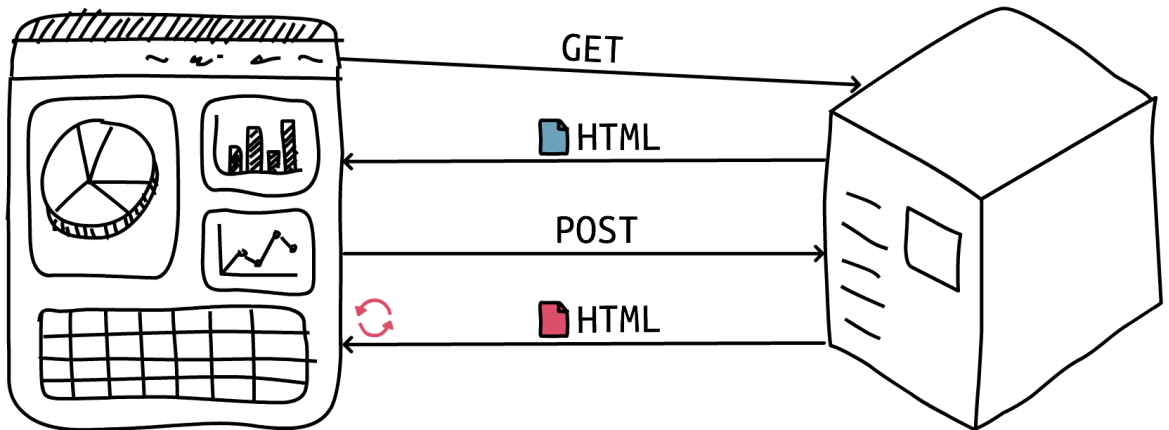
Взаимодействие



Взаимодействие



Взаимодействие



НОВАЯ ЗАМЕТКА

Осталось символов: 100

Отменить

Сохранить

Music

Books

Films

FILMS NOTES

**Voluptate incididunt velit**

Consequat cupidatat
consectetur do irure voluptate.
In nulla consectetur consequat
aliquip enim mollit qui veniam
aliquip.

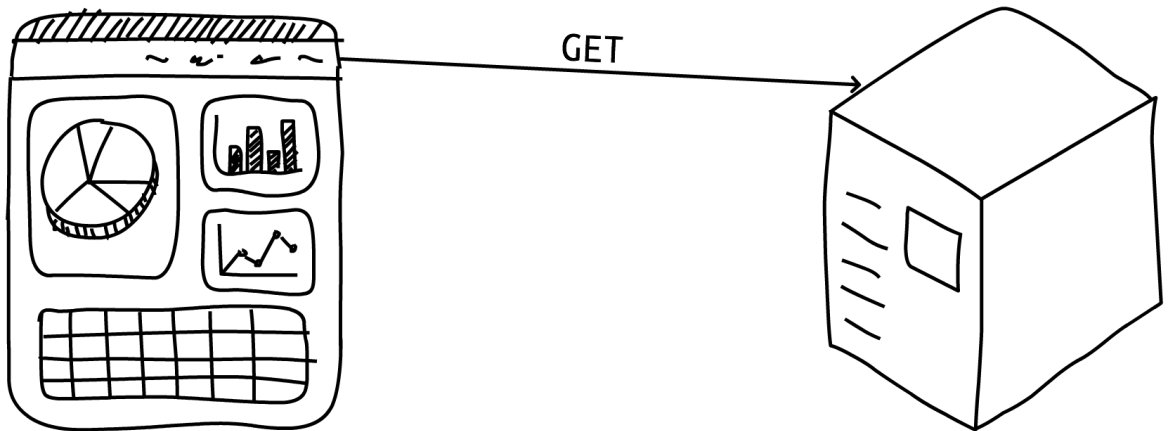
**Dolor ut laborum**

Consectetur nulla dolor est
nulla excepteur ex aliquip sit
pariatur magna. Lorem
laborum excepteur incididunt
elit aliqua occaecat anim ea
incididunt amet do eu
voluptate id.

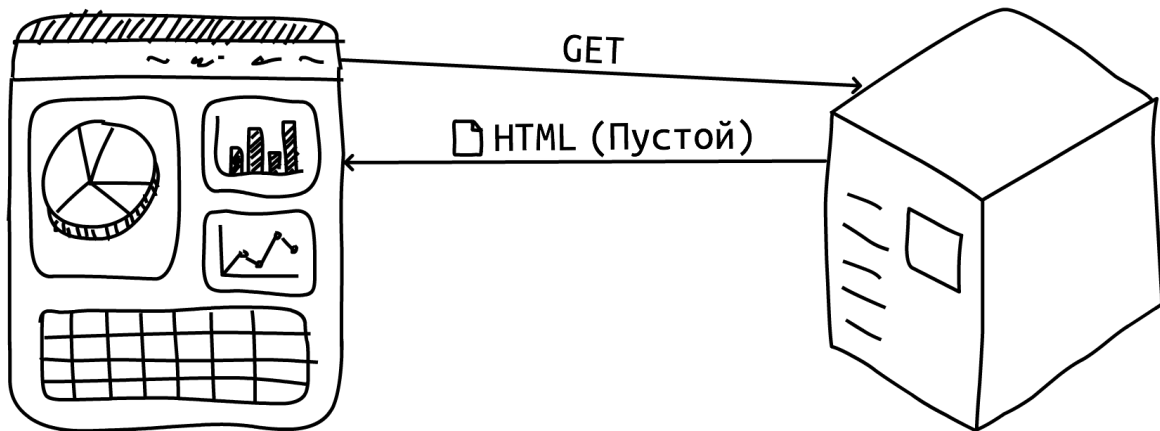
Single Page Application (SPA)

Веб-приложение или сайт, который загружает только одну страницу и все последующие запросы обрабатываются без полной перезагрузки страницы

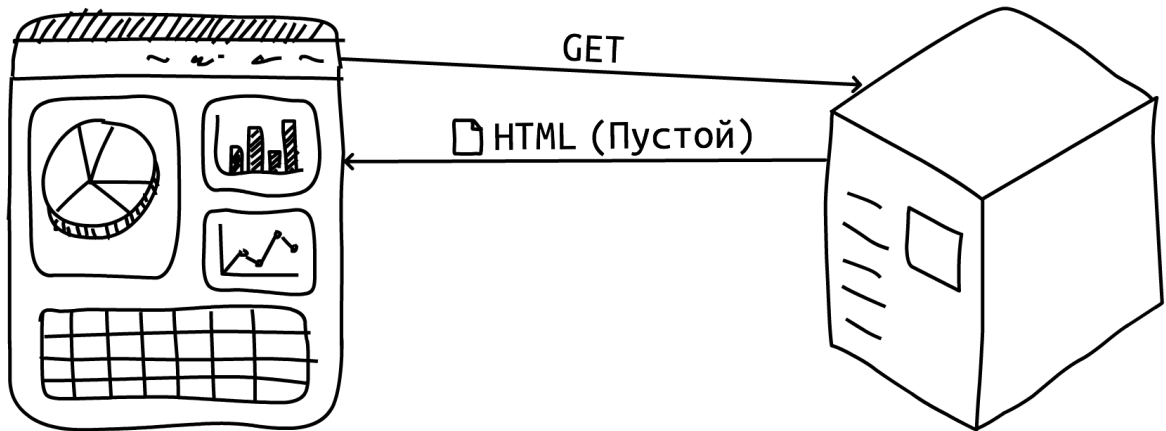
Взаимодействие



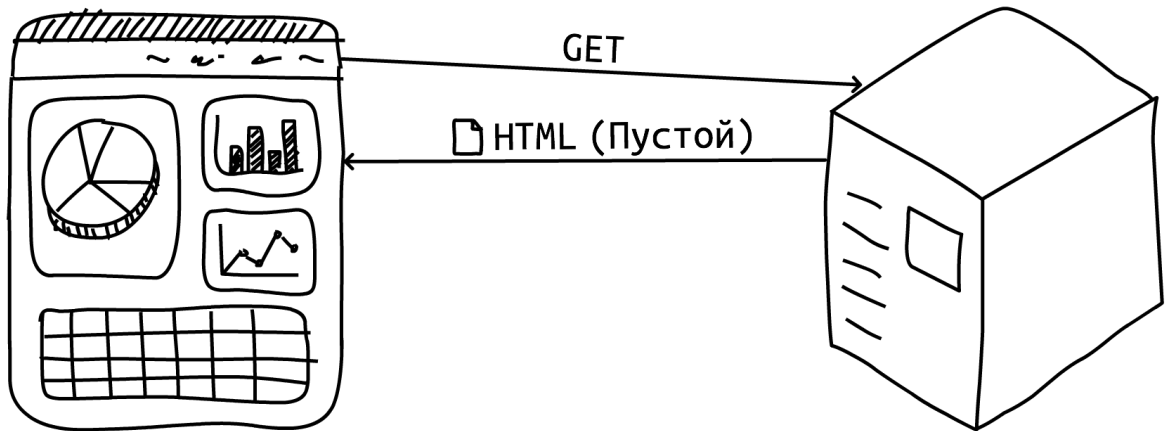
Взаимодействие

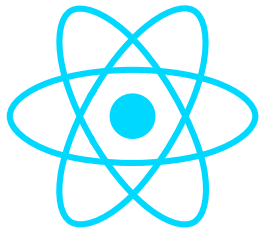


Взаимодействие



Взаимодействие





React

Концепции React

Компонентный подход

Эффективная абстракция над DOM

Реактивный рендеринг

Рендеринг элементов

НОВАЯ ЗАМЕТКА

Осталось символов: 100

МОИ ЗАМЕТКИ



Books
Books to read



Music
Music to listen



Films
Films to watch

Рендеринг элементов

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

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

ReactDOM.render(
  React.createElement(
    'div',
    { className: 'editor' },
    React.createElement('input', { placeholder: 'Ключ заметки' }),
    React.createElement('textarea', { placeholder: 'Текст заметки' }),
    React.createElement('button', null, 'Отменить'),
    React.createElement('button', null, 'Сохранить')
  ),
  root
);
```

Рендеринг элементов

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

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

ReactDOM.render(
  React.createElement(
    'div',
    { className: 'editor' },
    React.createElement('input', { placeholder: 'Ключ заметки' }),
    React.createElement('textarea', { placeholder: 'Текст заметки' }),
    React.createElement('button', null, 'Отменить'),
    React.createElement('button', null, 'Сохранить')
  ),
  root
);
```

Рендеринг элементов

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

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

ReactDOM.render(
  React.createElement(
    'div',
    { className: 'editor' },
    React.createElement('input', { placeholder: 'Ключ заметки' }),
    React.createElement('textarea', { placeholder: 'Текст заметки' }),
    React.createElement('button', null, 'Отменить'),
    React.createElement('button', null, 'Сохранить')
  ),
  root
);
```

Рендеринг элементов

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

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

ReactDOM.render(
  React.createElement(
    'div',
    { className: 'editor' },
    React.createElement('input', { placeholder: 'Ключ заметки' }),
    React.createElement('textarea', { placeholder: 'Текст заметки' }),
    React.createElement('button', null, 'Отменить'),
    React.createElement('button', null, 'Сохранить')
  ),
  root
);
```

Рендеринг элементов

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

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

ReactDOM.render(
  React.createElement(
    'div',
    { className: 'editor' },
    React.createElement('input', { placeholder: 'Ключ заметки' }),
    React.createElement('textarea', { placeholder: 'Текст заметки' }),
    React.createElement('button', null, 'Отменить'),
    React.createElement('button', null, 'Сохранить')
  ),
  root
);
```

to listen

символов: 85

нить

Сохранить

Music

Books

Films

MUSIC NOTES

**Culpa sint quis**

Id velit elit laborum mollit ex
adipiscing ipsum. Qui labore
eu irure voluptate cillum
aliquip proident.

**Ea laboris commodo
occaecat**

Sint ut commodo laborum
aliqua nulla nulla enim sunt.
Culpa reprehenderit aute aute
pariat est proident sit
commodo aliquip aute culpa
elit.

**Music**

Music to listen

Elements Console Sources Network Performance Memory

top Filter Default levels

Console Rendering

- ☒ **Paint flashing**
Highlights areas of the page (green) that need to be repainted
- ☐ **Layer borders**
Shows layer borders (orange/olive) and tiles (cyan)
- ☐ **FPS meter**
Plots frames per second, frame rate distribution, and GPU memory
- ☐ **Scrolling performance issues**
Highlights elements (teal) that can slow down scrolling, including touch & wheel event handler thread scrolling situations.
- ☐ **Hit-test borders**
Shows borders around hit-test regions

Emulate CSS media
Forces media type for testing print and screen styles

No emulation

```
const element = <h1>Что такое JSX?</h1>;
```

JSX

```
<select multiple>
  <option value="Пункт 1">Пункт 1</option>
  <option selected value="Пункт 2">Пункт 2</option>
</select>
```

JS

```
React.createElement(
  'select',
  { multiple: true },
  React.createElement(
    'option',
    { value: 'Пункт 1' },
    'Пункт 1'
  ),
  React.createElement(
    'option',
    { selected: true, value: 'Пункт 2' },
    'Пункт 2'
  )
);
```



```
input.jsx


    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Adipisci aut
    sunt cumque deleniti dolor dolores expedita id illo laboriosam, magnam
    numquam obcaecati officia pariatur perferendis quaerat reprehenderit
    andae, sint tempore totam ullam ut veniam. Assumenda blanditiis dicta,
    rationem mollitia necessitatibus nihil perferendis totam ut! Fugit itaque
    non qui?
  


<a href="https://yandex.ru"></a>


```

ch ♥ by @okbel

```
output.js

"use strict";

React.createElement(
  "div",
  null,
  React.createElement("img", { src: "./images/logo.svg", className: "logo" }),
  React.createElement(
    "p",
    null,
    "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Adipisci
    consequuntur cumque deleniti dolor dolores expedita id illo laboriosam, ma
    maxime numquam obcaecati officia pariatur perferendis quaerat reprehenderit
    repudiandae, sint tempore totam ullam ut veniam. Assumenda blanditiis dicta
    exercitationem mollitia necessitatibus nihil perferendis totam ut! Fugit it
    iusto non qui?",
    React.createElement("a", { href: "https://yandex.ru" })
  )
);
```

Все **html** атрибуты именуются в **camelCase** стиле

HTML

```
<input  
  class="name"  
  tabindex="2"  
  onchange="console.log('changed!');" />
```

JSX

```
<input  
  className="name"  
  tabIndex="2"  
  onChange={event => console.log('changed!')} />
```

Все элементы должны быть закрыты

HTML

```
<div>  
  <span>Введите ваше имя:</span><br>  
  <input type="text">  
</div>
```

JSX

```
<div>  
  <span>Введите ваше имя:</span><br />  
  <input type="text" />  
</div>
```

Имена пользовательских компонентов
должны начинаться с **заглавной** буквы

```
<Article />    React.createElement(Article, null);
```

```
<article />    React.createElement("article", null);
```

Может быть только **один** корневой элемент

```
<p>Первый абзац</p>
```

```
<p>Второй абзац</p>
```

SyntaxError: Adjacent JSX elements must be wrapped in an enclosing tag

Может быть только **один** корневой элемент

```
<div>  
  <p>Первый абзац</p>  
  <p>Второй абзац</p>  
</div>
```

```
<React.Fragment>  
  <p>Первый абзац</p>  
  <p>Второй абзац</p>  
</React.Fragment>
```

JavaScript выражения должны быть
заключены в {}

```
const user = { name: 'Максим' };
```

```
const element = <div>Привет, {user.name}</div>;
```

Всё содержимое экранируется

```
const html = '<strong>Мир</strong>';
```

```
const element = <div>Привет, {html}</div>;
```


Всё содержимое экранируется

```
const html = 'Привет, <strong>Мир</strong>!';

const element = (
  <div dangerouslySetInnerHTML={{ __html: html }} />
);
```

Компоненты

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

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

ReactDOM.render(
  <div className="editor">
    <input placeholder="Ключ заметки" />
    <textarea placeholder="Текст заметки" />
    <button>Отменить</button>
    <button>Сохранить</button>
  </div>,
  root
);
```

Function component

```
import React from 'react';

function Editor() {
  return (
    <div className="editor">
      <input placeholder="Ключ заметки" />
      <textarea placeholder="Текст заметки" />
      <button>Отменить</button>
      <button>Сохранить</button>
    </div>
  );
}
```

Class component

```
import React, { Component } from 'react';

class Editor extends Component {
  render() {
    return (
      <div className="editor">
        <input placeholder="Ключ заметки" />
        <textarea placeholder="Текст заметки" />
        <button>Отменить</button>
        <button>Сохранить</button>
      </div>
    );
  }
}
```

Объединение компонентов

НОВАЯ ЗАМЕТКА

Название заметки


Текст заметки

Осталось символов: 100


Отменить

Сохранить


МОИ ЗАМЕТКИ



Books
Books to read



Music
Music to listen



Films
Films to watch

Объединение компонентов

```
import Editor from './Editor';  
import Notes from './Notes';  
  
function NotesApp() {  
  return (  
    <div className="notes-app">  
      <Editor />  
      <Notes />  
    </div>  
  );  
}
```

Атрибуты

Props

Атрибуты (Props)

```
function Notes() {  
  return (  
    <div className="notes">  
      <Note name="Books" text="Books to read" />  
      <Note name="Music" text="Music to listen" />  
      <Note name="Films" text="Films to watch" />  
    </div>  
  );  
}
```


Атрибуты (Props)

```
interface NoteProps {  
  name: string;  
  
  text: string;  
}  
  
function Note({ name, text }: NoteProps) {  
  return (  
    <div className="note">  
      <h1>{name}</h1>  
      <p>{text}</p>  
    </div>  
  );  
}
```

Атрибуты (Props)

```
import React, { Component } from 'react';

interface NoteProps {
  name: string;
  text: string;
}

class Note extends Component<NoteProps> {
  render() {
    return (
      <div className="note">
        <h1>{this.props.name}</h1>
        <p>{this.props.text}</p>
      </div>
    );
  }
}
```

ПОТОМКИ

Children

ПОТОМКИ (Children)

```
function Notes() {  
  return (  
    <div className="notes">  
      <Note name="Books">  
        Books to read  
      </Note>  
  
      <Note name="Films">  
        <p>Films to read</p>  
        <button>Like</button>  
      </Note>  
  
      ...  
    </div>  
  );  
}
```

ПОТОМКИ (Children)

```
import React, { ReactNode } from 'react';

interface NoteProps {
  children: ReactNode;
  name: string;
}

function Note({ children, name }: NoteProps) {
  return (
    <div className="note">
      <h1 className="note__title">
        {name}
      </h1>
      <div className="note__content">
        {children}
      </div>
    </div>
  );
}
```

ПОТОМКИ (Children)

```
import React, { ReactNode } from 'react';

interface NoteProps {
  children: ReactNode;
  name: string;
}

function Note({ children, name }: NoteProps) {
  return (
    <div className="note">
      <h1 className="note__title">
        {name}
      </h1>
      <div className="note__content">
        {children}
      </div>
    </div>
  );
}
```

Условный рендеринг



Warning

Non labore cillum nulla consequat ea
proident sit dolore sint sunt ad
nostrud cupidatat laboris.



Books

Cillum est veniam culpa culpa sunt
sunt. Enim pariatur et irure sint nulla
quis non ea dolor Lorem in duis.
Commodo non laboris exercitation
consectetur adipisicing enim ad
occaecat dolor elit quis qui irure nisi.

Условный рендеринг

```
function Note({ children, name, type }: NoteProps) {  
  return (  
    <div className="note">  
      <h1 className="note__title">  
        {type === 'warning' ? 'Warning' : name}  
      </h1>  
      ...  
    </div>  
  );  
}
```


Условный рендеринг

```
function Note({ children, name, type }: NoteProps) {  
  return (  
    <div className="note">  
      <h1 className="note__title">  
        {type === 'warning' ? 'Warning' : name}  
      </h1>  
      ...  
    </div>  
  );  
}
```

Работа со списками



Warning

Non labore cillum nulla consequat ea
proident sit dolore sint sunt ad
nostrud cupidatat laboris.



Books

Cillum est veniam culpa culpa sunt
sunt. Enim pariatur et irure sint nulla
quis non ea dolor Lorem in duis.
Commodo non laboris exercitation
consectetur adipisicing enim ad
occaecat dolor elit quis qui irure nisi.

Работа со списками

```
function NotesList() {  
  return (  
    <div className="notes-list">  
      <Note name="Books" text="Books to read" />  
      <Note name="Films" text="Films to watch" />  
      <Note name="Music" text="Music to listen" />  
    </div>  
  );  
}
```

Работа со списками

```
function NotesList({ notes }: NotesListProps) {  
  return (  
    <div className="notes-list">  
      {notes.map(note => (  
        <Note name={note.name} text={note.text} />  
      ))}  
    </div>  
  );  
}
```

Each child in an array or iterator should have a unique "key" prop.
Check the render method of **NotesList**. See <https://fb.me/react-warning-keys> for more information.

Работа со списками

```
function NotesList({ notes }: NotesListProps) {  
  return (  
  
    <div className="notes-list">  
      {notes.map(note => (  
        <Note  
          name={note.name}  
          text={note.text}  
          key={note.id}  
        />  
      ))}  
    </div>  
  );  
}
```

Keys

Key должен однозначно определять элемент списка и быть стабильным

Повторяющиеся key в рамках одного списка недопустимы

Использовать индекс элемента в массиве лучше только тогда, когда другого выхода нет

Состояние компонента (State)



Books

Non labore cillum nulla consequat ea
proident sit dolore sint sunt ad
nostrud cupidatat laboris.

Read more



Состояние компонента (State)

Изменяется через специальный интерфейс

Компонент автоматически реагирует на изменения состояния

Состояние компонента (State)

```
interface NoteState {  
    isReadMoreClicked: boolean;  
}  
  
class Note extends Component<NoteProps, NoteState> {  
    state: NoteState = { isReadMoreClicked: false }  
  
    handleReadMoreClick = () => {  
        this.setState({ isReadMoreClicked: true });  
    }  
  
    ...  
}
```

Состояние компонента (State)

```
interface NoteState {  
    isReadMoreClicked: boolean;  
}  
  
class Note extends Component<NoteProps, NoteState> {  
    state: NoteState = { isReadMoreClicked: false }  
  
    handleReadMoreClick = () => {  
        this.setState({ isReadMoreClicked: true });  
    }  
  
    ...  
}
```

Состояние компонента (State)

```
interface NoteState {  
    isReadMoreClicked: boolean;  
}  
  
class Note extends Component<NoteProps, NoteState> {  
    state: NoteState = { isReadMoreClicked: false }  
  
    handleReadMoreClick = () => {  
        this.setState({ isReadMoreClicked: true });  
    }  
  
    ...  
}
```

Состояние компонента (State)

```
interface NoteState {  
    isReadMoreClicked: boolean;  
}  
  
class Note extends Component<NoteProps, NoteState> {  
    state: NoteState = { isReadMoreClicked: false }  
  
    handleReadMoreClick = () => {  
        this.setState({ isReadMoreClicked: true });  
    }  
  
    ...  
}
```

Состояние компонента (State)

```
class Note extends Component<NoteProps, NoteState> {  
  ...  
  
  render() {  
    return (  
      <div className="note">  
        <div className="note__name">Books</div>  
        <div className="note__text">Books to read</div>  
        {this.state.isReadMoreClicked  
          ? <button onClick={handleReadMoreClick}>Read more</button>  
          : <div className="note__additional-text">Additional text</div>  
        }  
      </div>  
    );  
  }  
}
```

Состояние компонента (State)

```
class Note extends Component<NoteProps, NoteState> {  
  ...  
  
  render() {  
    return (  
      <div className="note">  
        <div className="note__name">Books</div>  
        <div className="note__text">Books to read</div>  
        {this.state.isReadMoreClicked  
          ? <button onClick={handleReadMoreClick}>Read more</button>  
          : <div className="note__additional-text">Additional text</div>  
        }  
      </div>  
    );  
  }  
}
```

Не изменяйте состояние напрямую

// Неправильно

```
this.state.isReadMoreClicked = true;
```

// Правильно

```
this.setState({ isReadMoreClicked: true });
```

Изменение состояния может быть асинхронным

// Неправильно

```
this.setState({ counter: this.state.counter + 1 });  
this.setState({ counter: this.state.counter + 1 });
```

// Правильно

```
this.setState(state => ({ counter: state.counter + 1 }));  
this.setState(state => ({ counter: state.counter + 1 }));
```


Можно обновлять не все состояние целиком

```
interface State {  
    notes: Note[];  
    isReadMoreClicked: boolean;  
}  
  
this.setState({ isReadMoreClicked: true });
```

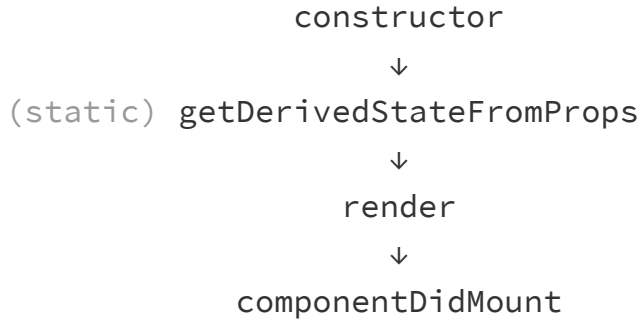
Жизненный цикл компонента

Component life-cycle

Этапы жизненного цикла

1. Монтирование компонента (Mounting)
2. Изменение атрибутов или состояния (Updating)
3. Удаление компонента (Unmounting)

Монтирование (Mounting)



Изменение атрибутов (Updating)

`(static) getDerivedStateFromProps`



`shouldComponentUpdate`



`render`



`componentDidUpdate`

Изменение состояния (Updating)

`shouldComponentUpdate`



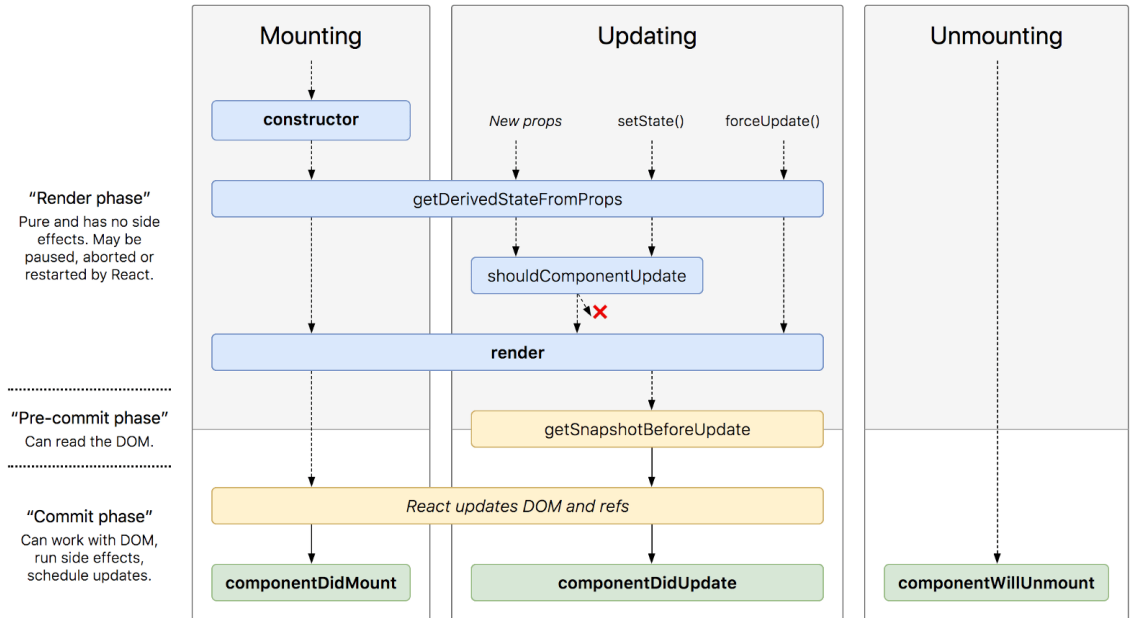
`render`



`componentDidUpdate`

Удаление компонента (Unmounting)

`componentWillUnmount`





Consectetur qui non irure

Nostrud est officia occaecat
minim dolore do voluptate
labore id Lorem adipisicing.
Non qui sunt fugiat labore
pariatur labore.

♡ 8



Dolor mollit quis duis sit incididunt

Nostrud est officia occaecat
minim dolore do voluptate
labore id Lorem adipisicing.
Non qui sunt fugiat labore
pariatur labore.

♡ 8



Sunt culpa voluptate

Nostrud est officia occaecat
minim dolore do voluptate
labore id Lorem adipisicing.
Non qui sunt fugiat labore
pariatur labore.

♡ 47



Culpa veniam ut pariatur ad

```
class Notes extends Component<NotesProps, NotesState> {  
  state: NotesState = { backToTop: false }  
  
  componentDidMount() {  
    window.addEventListener('scroll', this.handleScroll);  
  }  
  
  componentWillUnmount() {  
    window.removeEventListener('scroll', this.handleScroll);  
  }  
  
  handleScroll = () => {  
    if (!this.state.backToTop && document.documentElement.scrollTop > 300) {  
      this.setState({ backToTop: true });  
    }  
  
    ...  
  }  
  
  ...  
}
```

```
class Notes extends Component<NotesProps, NotesState> {  
  state: NotesState = { backToTop: false }  
  
  componentDidMount() {  
    window.addEventListener('scroll', this.handleScroll);  
  }  
  
  componentWillUnmount() {  
    window.removeEventListener('scroll', this.handleScroll);  
  }  
  
  handleScroll = () => {  
    if (!this.state.backToTop && document.documentElement.scrollTop > 300) {  
      this.setState({ backToTop: true });  
    }  
  
    ...  
  }  
  
  ...  
}
```

```
class Notes extends Component<NotesProps, NotesState> {  
  state: NotesState = { backToTop: false }  
  
  componentDidMount() {  
    window.addEventListener('scroll', this.handleScroll);  
  }  
  
  componentWillUnmount() {  
    window.removeEventListener('scroll', this.handleScroll);  
  }  
  
  handleScroll = () => {  
    if (!this.state.backToTop && document.documentElement.scrollTop > 300) {  
      this.setState({ backToTop: true });  
    }  
  
    ...  
  }  
  
  ...  
}
```

Работа с формами

Название заметки

Films to wa|

Осталось символов: 89

Отменить

Сохранить

Работа с формами

Как получить данные из формы?

Как обрабатывать события?

Виды компонентов

1. Неуправляемые (Uncontrolled)
2. Управляемые (Controlled)

Неуправляемые компоненты

(Uncontrolled components)

DOM управляет текущим состоянием компонента

Ссылка на DOM элемент (Ref)

```
import React, { Component, createRef } from 'react';

class UncontrolledForm extends Component {
  inputRef = createRef<HTMLInputElement>()

  handleSubmit = () => {
    if (this.inputRef.current) {
      this.makeSomeApiRequest(this.inputRef.current.value);
    }
  }

  render() {
    return (
      <div>
        <input ref={this.inputRef} />
        <button onClick={this.handleSubmit}>Отправить</button>
      </div>
    );
  }
}
```

Ссылка на DOM элемент (Ref)

```
import React, { Component, createRef } from 'react';

class UncontrolledForm extends Component {
  inputRef = createRef<HTMLInputElement>()

  handleSubmit = () => {
    if (this.inputRef.current) {
      this.makeSomeApiRequest(this.inputRef.current.value);
    }
  }

  render() {
    return (
      <div>
        <input ref={this.inputRef} />
        <button onClick={this.handleSubmit}>Отправить</button>
      </div>
    );
  }
}
```

Ссылка на DOM элемент (Ref)

```
import React, { Component, createRef } from 'react';

class UncontrolledForm extends Component {
  inputRef = createRef<HTMLInputElement>()

  handleSubmit = () => {
    if (this.inputRef.current) {
      this.makeSomeApiRequest(this.inputRef.current.value);
    }
  }

  render() {
    return (
      <div>
        <input ref={this.inputRef} />
        <button onClick={this.handleSubmit}>Отправить</button>
      </div>
    );
  }
}
```

Ссылка на DOM элемент (Ref)

```
import React, { Component, createRef } from 'react';

class UncontrolledForm extends Component {
  inputRef = createRef<HTMLInputElement>()

  handleSubmit = () => {
    if (this.inputRef.current) {
      this.makeSomeApiRequest(this.inputRef.current.value);
    }
  }

  render() {
    return (
      <div>
        <input ref={this.inputRef} />
        <button onClick={this.handleSubmit}>Отправить</button>
      </div>
    );
  }
}
```

Неуправляемые компоненты

(Uncontrolled components)

Простота

Один обработчик на всю форму

Контроль из кода

Управляемые компоненты

(Controlled components)

React управляет текущим состоянием компонента

Управляемые компоненты

```
import React, { Component, FormEvent } from 'react';

class ControlledForm extends Component<ControlledFormProps, ControlledFormState> {
  state: ControlledFormState = { value: '' }

  handleChange = (event: FormEvent<HTMLInputElement>) => {
    this.setState({ value: event.target.value });
  }

  handleSubmit = () => this.makeSomeApiRequest(this.state.value)

  render() {
    return (
      <div>
        <input value={this.state.value} onChange={this.handleChange} />
        <button onClick={this.handleSubmit}>Отправить</button>
      </div>
    );
  }
}
```

Управляемые компоненты

```
import React, { Component, FormEvent } from 'react';

class ControlledForm extends Component<ControlledFormProps, ControlledFormState> {
  state: ControlledFormState = { value: '' }

  handleChange = (event: FormEvent<HTMLInputElement>) => {
    this.setState({ value: event.target.value });
  }

  handleSubmit = () => this.makeSomeApiRequest(this.state.value)

  render() {
    return (
      <div>
        <input value={this.state.value} onChange={this.handleChange} />
        <button onClick={this.handleSubmit}>Отправить</button>
      </div>
    );
  }
}
```


Управляемые компоненты

```
import React, { Component, FormEvent } from 'react';

class ControlledForm extends Component<ControlledFormProps, ControlledFormState> {
  state: ControlledFormState = { value: '' }

  handleChange = (event: FormEvent<HTMLInputElement>) => {
    this.setState({ value: event.target.value });
  }

  handleSubmit = () => this.makeSomeApiRequest(this.state.value)

  render() {
    return (
      <div>
        <input value={this.state.value} onChange={this.handleChange} />
        <button onClick={this.handleSubmit}>Отправить</button>
      </div>
    );
  }
}
```

Управляемые компоненты

(Controlled components)

Состояние управляется через интерфейс

Полный контроль над изменениями

Более сложное взаимодействие

Быстрый старт

create-react-app

```
$ npm install -g create-react-app  
$ create-react-app notes-app --typescript  
$ cd notes-app  
$ npm run start
```

Compiled successfully!

You can now view **notes-app** in the browser.

Local: <http://localhost:3000/>

Note that the development build is not optimized.

To create a production build, use `npm run build`.

create-react-app

Очень быстрый старт

Активное развитие и поддержка от разработчиков React

TypeScript «из коробки»

Почитать

Официальная документация React

What is JSX?

Index as a key is an anti-pattern

Посмотреть

The Beginner's Guide to React

Start Learning React

Advanced React