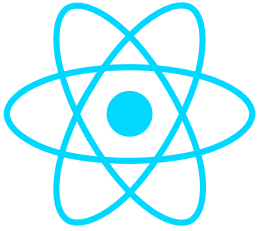

React

Advanced

Single Page Application (SPA)

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

Express.js

```
import express from 'express';

const app = express();

app.get('/notes', (req, res) => res.render('notes'));

app.all('*', (req, res) => res.sendStatus(404));

app.listen(8080);
```

React Router

React Router

```
$ npm install react-router-dom @types/react-router-dom
```

```
import { Fragment } from 'react';
import { BrowserRouter, Route } from 'react-router-dom';

function NotesApp() {
    return (
        <BrowserRouter>
            <Fragment>
                <Route path="/" component={HomePage} />
                <Route path="/notes" component={NotesPage} />
            </Fragment>
        </BrowserRouter>
    );
}
```

React Router

```
$ npm install react-router-dom @types/react-router-dom
```

```
import { Fragment } from 'react';
import { BrowserRouter, Route } from 'react-router-dom';

function NotesApp() {
    return (
        <BrowserRouter>
            <Fragment>
                <Route exact path="/" component={HomePage} />
                <Route path="/notes" component={NotesPage} />
            </Fragment>
        </BrowserRouter>
    );
}
```

React Router

```
import { Fragment } from 'react';
import { BrowserRouter, Route, Link } from 'react-router-dom';


function NotesApp() {
    return (
        <BrowserRouter>
            <Fragment>
                <nav>
                    <Link to="/home">Home</Link>
                    <Link to="/notes">Notes</Link>
                </nav>
                <Route exact path="/home" component={HomePage} />
                <Route exact path="/notes" component={NotesPage} />
            </Fragment>
        </BrowserRouter>
    );
}
```

React Router

```
import { Fragment } from 'react';
import { BrowserRouter, Route, Link } from 'react-router-dom';


function NotesApp() {
    return (
        <BrowserRouter>
            <Fragment>
                <nav>
                    <Link to="/home">Home</Link>
                    <Link to="/notes">Notes</Link>
                </nav>
                <Route exact path="/home" component={HomePage} />
                <Route exact path="/notes" component={NotesPage} />
            </Fragment>
        </BrowserRouter>
    );
}
```

React Router

```
import { Fragment } from 'react';
import { BrowserRouter, Route, NavLink } from 'react-router-dom';

function NotesApp() {
    return (
        <BrowserRouter>
            <Fragment>
                <nav>
                    <NavLink activeClassName="link_active" to="/home">
                        Home
                    </NavLink>
                    <NavLink activeClassName="link_active" to="/notes">
                        Notes
                    </NavLink>
                </nav>
                <Route exact path="/home" component={HomePage} />
                <Route exact path="/notes" component={NotesPage} />
            </Fragment>
        </BrowserRouter>
    );
}
```

React Router

```
import { Fragment } from 'react';
import { BrowserRouter, Route } from 'react-router-dom';

function NotesApp() {
    return (
        <BrowserRouter>
            <Fragment>
                <Route exact path="/notes/:id" component={NotePage} />
            </Fragment>
        </BrowserRouter>
    );
}
```

React Router

```
import { RouteComponentProps } from 'react-router-dom';

type NotePageProps = RouteComponentProps<{ id: string }>;

function NotePage({ match }: NotePageProps) {
    return <h1>Note id: {match.params.id}</h1>;
}
```

React Router

```
import { RouteComponentProps } from 'react-router-dom';

type NotePageProps = RouteComponentProps<{ id: string }>;

function NotePage({ match }: NotePageProps) {
    return <h1>Note id: {match.params.id}</h1>;
}
```

React Router

```
import { RouteComponentProps } from 'react-router-dom';

type NotePageProps = RouteComponentProps<{ id: string }>;

function NotePage({ match }: NotePageProps) {
    return <h1>Note id: {match.params.id}</h1>;
}
```

React Router

```
import { Fragment } from 'react';
import { BrowserRouter, Route } from 'react-router-dom';

function NotesApp() {
    return (
        <BrowserRouter>
            <Fragment>
                <Route exact path="/home" component={HomePage} />
                <Route exact path="/notes" component={NotesPage} />
                <Route exact path="/notes/:id" component={NotePage} />
                <Route component={NotFoundPage} />
            </Fragment>
        </BrowserRouter>
    );
}
```

React Router

```
import { Fragment } from 'react';
import { BrowserRouter, Route, Switch } from 'react-router-dom';

function NotesApp() {
    return (
        <BrowserRouter>
            <Switch>
                <Route exact path="/home" component={HomePage} />
                <Route exact path="/notes" component={NotesPage} />
                <Route exact path="/notes/:id" component={NotePage} />
                <Route component={NotFoundPage} />
            </Switch>
        </BrowserRouter>
    );
}
```

Работа с API

Работа с API

```
class Notes extends Component {
    ...

    render() {
        if (this.state.loading) {
            return 'Загрузка...';
        }

        if (this.state.error) {
            return 'Что-то пошло не так';
        }

        return (
            <div className="notes">
                {this.state.notes.map(note => (
                    <Note key={note.id} name={note.name} text={note.text} />
                ))}
            </div>
        );
    }
}
```

Работа с API

```
class Notes extends Component<NotesProps, NotesState> {
    state: NotesState = {
        loading: true,
        error: false,
        notes: null
    }

    componentDidMount() {
        fetch('/api/notes')
            .then(response => response.json())
            .then(notes => {
                this.setState({ notes, loading: false });
            })
            .catch(() => {
                this.setState({ error: true, loading: false });
            });
    }

    ...
}
```

Работа с API

```
class Notes extends Component<NotesProps, NotesState> {
    state: NotesState = {
        loading: true,
        error: false,
        notes: null
    }

    componentDidMount() {
        fetch('/api/notes')
            .then(response => response.json())
            .then(notes => {
                this.setState({ notes, loading: false });
            })
            .catch(() => {
                this.setState({ error: true, loading: false });
            });
    }

    ...
}
```

Работа с API

```
class Notes extends Component<NotesProps, NotesState> {
    state: NotesState = {
        loading: true,
        error: false,
        notes: null
    }

    componentDidMount() {
        fetch('/api/notes')
            .then(response => response.json())
            .then(notes => {
                this.setState({ loading: false, notes });
            })
            .catch(() => {
                this.setState({ error: true, loading: false });
            });
    }

    ...
}
```

Компоненты высшего порядка

Higher-Order Components (HOC)

Функции высшего порядка

Higher-Order Functions

Функция, которая принимает функцию
в качестве аргумента или
возвращает функцию в качестве результата,
называется функцией высшего порядка

Функции высшего порядка

```
function greaterThen(x) {
    return function (y) {
        return x > y;
    }
}

const greaterThen10 = greaterThen(10);

greaterThen10(42); // true
```

Функции высшего порядка

```
function logArguments(f) {
    return function (...args) {
        const result = f(...args);

        console.log('called with', args, 'result', result);

        return result;
    }
}

logArguments(Math.max)(1, 2, 3);
// called with [1, 2, 3] result 3
```

Функции высшего порядка

```
[...].map(n => ...);

[...].filter(n => ...);

[...].reduce((acc, n) => ...);
```

Компоненты высшего порядка

Принимают компонент в качестве аргумента

Возвращают новый компонент

Могут отрендерить переданный компонент

withRouter

```
import React, { Component } from 'react';
import { RouteComponentProps, withRouter } from 'react-router-dom';


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

type NoteProps = NoteOwnProps & RouteComponentProps<{ id: string }>;

function Note({ name, text, match }: NoteProps) {
    return <div>ID: {match.params.id}</div>;
}

const NoteWithRouter = withRouter(Note);
```

withRouter

```
import React, { Component } from 'react';
import { RouteComponentProps, withRouter } from 'react-router-dom';


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

type NoteProps = NoteOwnProps & RouteComponentProps<{ id: string }>;

function Note({ name, text, match }: NoteProps) {
    return <div>ID: {match.params.id}</div>;
}

const NoteWithRouter = withRouter(Note);
```

withRouter

```
import React, { Component } from 'react';
import { RouteComponentProps, withRouter } from 'react-router-dom';

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

type NoteProps = NoteOwnProps & RouteComponentProps<{ id: string }>;

function Note({ name, text, match }: NoteProps) {
    return <div>ID: {match.params.id}</div>;
}

const NoteWithRouter = withRouter(Note);
```

withRouter

```
import React, { Component } from 'react';
import { RouteComponentProps, withRouter } from 'react-router-dom';

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

type NoteProps = NoteOwnProps & RouteComponentProps<{ id: string }>;

function Note({ name, text, match }: NoteProps) {
    return <div>ID: {match.params.id}</div>;
}

const NoteWithRouter = withRouter(Note);
```

Компоненты высшего порядка

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

function hoc(WrappedComponent) {
    return class extends Component {
        render() {
            return <WrappedComponent {...this.props} />;
        }
    }
}
```

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

function withData(url, WrappedComponent) {
    export class extends Component {
        state = { ... }


        componentDidMount() { // Запрашиваем данные, используя полученный url
            ...               // См. «Работа с API»
        }

        render() {
            if (this.state.loading) {
                return 'Загрузка...';
            }

            if (this.state.error) {
                return 'Что-то пошло не так';
            }

            return <WrappedComponent data={this.state.data} {...this.props} />;
        }
    }
}
```

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

function withData(url, WrappedComponent) {
    export class extends Component {
        state = { ... }


        componentDidMount() { // Получаем данные, используя полученный url
            ...               // См. «Работа с API»
        }

        render() {
            if (this.state.loading) {
                return 'Загрузка...';
            }

            if (this.state.error) {
                return 'Что-то пошло не так';
            }

            return <WrappedComponent data={this.state.data} {...this.props} />;
        }
    }
}
```

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

function withData(url, WrappedComponent) {
    export class extends Component {
        state = { ... }


        componentDidMount() { // Получаем данные, используя полученный url
            ...               // См. «Работа с API»
        }

        render() {
            if (this.state.loading) {
                return 'Загрузка...';
            }

            if (this.state.error) {
                return 'Что-то пошло не так';
            }

            return <WrappedComponent data={this.state.data} {...this.props} />;
        }
    }
}
```

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

function withData(url, WrappedComponent) {
    export class extends Component {
        state = { ... }


        componentDidMount() { // Получаем данные, используя полученный url
            ...               // См. «Работа с API»
        }

        render() {
            if (this.state.loading) {
                return 'Загрузка...';
            }

            if (this.state.error) {
                return 'Что-то пошло не так';
            }

            return <WrappedComponent data={this.state.data} {...this.props} />;
        }
    }
}
```

```
import withData from './hoc/with-data';

// Данные просто приходят в пропсах

function Notes({ data }: NotesProps)  {
    return (
        <div className="notes">
            {data.map(note => (
                <Note key={note.id} name={note.name} text={note.text} />
            ))}
        <div>
    );
}

// Оборачиваем Notes, чтобы получить данные
export default withData('/api/notes', Notes);
```

```
import Notes from './components/notes';


function NotesApp() {
    // Используем как самый обычный компонент
    return (
        <div className="notes-app">
            <Notes />
            ...
        </div>
    )
}
```

```
withData('/api/notes', Notes);

withData('/api/notes/???', Note);
```

```
withData('/api/notes', Notes);

withData(props => `/api/notes/${props.id}`, Note);
```

```
withData({
    url: props => `/api/notes/${props.id}`,
    propName: 'note',
    headers: {
        Authorization: 'OAuth ...'
    }
}, Note);
```

Композиция

```
withRouter(withData('/api/notes', Notes));

withSomethingElse(
    withRouter(
        withData('/api/notes', Notes)
    )
);
```

Композиция

```
// lodash, underscore, ramda ...
import { compose } from 'redux';

compose(f, g, h)(x) === f(g(h(x)));
```

Композиция

```
withData('/api/notes')(Notes);


function withData(url) {
    return function (WrappedComponent) {
        return class extends Component {
            ...
        }
    }
}

compose(
    withSomethingElse,
    withRouter,
    withData('/api/notes')
)(Notes);
```

Компоненты высшего порядка

Решают проблему повторного использования кода

Возможны конфликты имен

Не понятно откуда пришли данные

Render Props

Render Props

```
function Notes() {
    return (
        <DataProvider
            url="/api/notes"
            render={notes => notes.map(note => (
                <Note
                    key={note.id}
                    name={note.name}
                    text={note.text}
                />
            ))}
        />
    );
}
```

Render Props

```
function renderNotes(notes) {
    return notes.map(note => (
        <Note key={note.id} name={note.name} text={note.text} />
    ));
}

function Notes() {
    return <DataProvider url="/api/notes" render={renderNotes} />;
}
```

Render Props

```
class DataProvider extends Component<Props, State> {
    state = { ... }

    componentDidMount() { // Запрашиваем данные, используя полученный url
        ...               // См. «Работа с API»
    }

    render() {
        if (this.state.loading) {
            return 'Загрузка...';
        }

        if (this.state.error) {
            return 'Что-то пошло не так';
        }

        return this.props.render(this.state.data);
    }
}
```

Render Props

```
function Notes() {
    return (
        <DataProvider url="/api/notes">
            {notes => notes.map(note => (
                <Note
                    key={note.id}
                    name={note.name}
                    text={note.text}
                />
            ))}
        </DataProvider>
    );
}
```

Render Props

```
function Notes() {
    return (
        <DataProvider url="/api/notes">
            {({ data, error, loading }) => {

                if (loading) {
                    return <LoadingScreen />;
                }

                ...

                return notes.map(note => (
                    <Note
                        key={note.id}
                        name={note.name}
                        text={note.text}
                    />
                ));
            }}
        </DataProvider>
```

Render Props

```
class DataProvider extends Component<Props, State> {
    state = { ... }

    componentDidMount() { // Запрашиваем данные, используя полученный url
        ...               // См. «Работа с API»
    }

    render() {
        return this.props.children(this.state);
    }
}
```

Render Props

Решают проблему повторного использования кода

Не имеют проблем HOC

Нет сложностей с типизацией

Увеличивает вложенность

Render Props Hell

```
<DataFetcher>
    {data => (
        <Actions>
            {actions => (
                <Translations>
                    {translations => (
                        <Styles>
                            {styles => ...}
                        </Styles>
                    )}
                </Translations>
            )}
        </Actions>
    )}
</DataFetcher>
```

```
const tabs = [

    {
        label: 'Books',
        content: 'Dolore exercitation consequat sunt est anim occaecat.'
    },
    {
        label: 'Films',
        content: 'Cillum proident enim id pariatur minim reprehenderit.'
    },
    ...
];
```

```
<Tabs tabs={tabs} />
```

```
<Tabs
      tabs={tabs}
      tabsPosition="bottom"
/>
```

```
<Tabs
    tabs={tabs}
    tabsPosition="bottom"
    disabled={[ 1 ]}
/>
```

```
<Tabs
    tabs={tabs}

    tabsPosition="bottom"
    disabled={[ 1 ]}
    onTabChange={...}
    activeTabClassName="..."
    icons={[...]}
    ...
/>
```

Составные компоненты

Compound components

Составные компоненты

```
<Tabs initialTabId="books">
    <TabList>

        <Tab id="books">Books</Tab>
        <Tab id="films">Films</Tab>
    </TabList>
    <TabPanels>
        <TabPanel id="books">
            Dolore exercitation consequat sunt est anim occaecat.
        </TabPanel>
        <TabPanel id="films">
            Cillum proident enim id pariatur minim reprehenderit.
        </TabPanel>
    </TabPanels>
</Tabs>
```

Составные компоненты

```
<Tabs initialTabId="books">
    <TabPanels>

        <TabPanel id="books">
            Dolore exercitation consequat sunt est anim occaecat.
        </TabPanel>
        <TabPanel id="films">
            Cillum proident enim id pariatur minim reprehenderit.
        </TabPanel>
    </TabPanels>
    <TabList>
        <Tab id="books">Books</Tab>
        <Tab id="films">Films</Tab>
    </TabList>
</Tabs>
```

Составные компоненты

```
<Tabs initialTabId="books">
    <TabPanels>

        <TabPanel id="books">
            Dolore exercitation consequat sunt est anim occaecat.
        </TabPanel>
        <TabPanel id="films">
            Cillum proident enim id pariatur minim reprehenderit.
        </TabPanel>
    </TabPanels>
    <TabList>
        <Tab id="books">Books</Tab>
        <Tab id="films" disabled>Films</Tab>
    </TabList>
</Tabs>
```

Составные компоненты

```
<select>
    <optgroup label="Группа">
        <option>Пункт 1</option>
        <option>Пункт 2</option>
    </optgroup>
    <option>Пункт 3</option>
    <option>Пункт 4</option>
</select>
```

Контекст

Context

Контекст

```
// theme-context.ts
import { createContext } from 'react';

const ThemeContext = createContext('light');

export default ThemeContext;
```

Контекст

```
import React from 'react';

import ThemeContext from './theme-context';

function Note() {
    return (
        <ThemeContext.Consumer>
            {theme => ...}
        </ThemeContext.Consumer>
    );
}
```

Контекст

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

import ThemeContext from './theme-context';

class NotesApp extends Component {
    ...

    render() {
        return (
            <ThemeContext.Provider value="dark">
                ...
            </ThemeContext.Provider>
        );
    }
}
```

Контекст

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

import ThemeContext from './theme-context';

class NotesApp extends Component {
    ...

    render() {
        return (
            <ThemeContext.Provider value={this.state.theme}>
                ...
            </ThemeContext.Provider>
        );
    }
}
```

Составные компоненты

```
<Tabs initialTabId="books">
    <TabList>

        <Tab id="books">Books</Tab>
        <Tab id="films">Films</Tab>
    </TabList>
    <TabPanels>
        <TabPanel id="books">
            Dolore exercitation consequat sunt est anim occaecat.
        </TabPanel>
        <TabPanel id="films">
            Cillum proident enim id pariatur minim reprehenderit.
        </TabPanel>
    </TabPanels>
</Tabs>
```

Составные компоненты

```
import { createContext } from 'react';

const TabsContext = createContext({
    activeTabId: '',
    changeTab: () => {}
});
```

Составные компоненты

```
class Tabs extends Component<TabsProps, TabsState> {
    state: TabsState = {
        activeTabId: this.props.initialTabId

    }

    changeTab = (id: string) => this.setState({ activeTabId: id })

    render() {
        const contextValue = { activeTabId, changeTab: this.changeTab };

        return (
            <TabsContext.Provider value={contextValue}>
                {children}
            </TabsContext.Provider>
        );
    }
}
```

Составные компоненты

```
class Tabs extends Component<TabsProps, TabsState> {
    state: TabsState = {
        activeTabId: this.props.initialTabId

    }

    changeTab = (id: string) => this.setState({ activeTabId: id })

    render() {
        const contextValue = { activeTabId, changeTab: this.changeTab };

        return (
            <TabsContext.Provider value={contextValue}>
                {children}
            </TabsContext.Provider>
        );
    }
}
```

Составные компоненты

```
class Tabs extends Component<TabsProps, TabsState> {
    state: TabsState = {
        activeTabId: this.props.initialTabId

    }

    changeTab = (id: string) => this.setState({ activeTabId: id })

    render() {
        const contextValue = { activeTabId, changeTab: this.changeTab };

        return (
            <TabsContext.Provider value={contextValue}>
                {children}
            </TabsContext.Provider>
        );
    }
}
```

Составные компоненты

```
class Tabs extends Component<TabsProps, TabsState> {
    state: TabsState = {
        activeTabId: this.props.initialTabId

    }

    changeTab = (id: string) => this.setState({ activeTabId: id })

    render() {
        const contextValue = { activeTabId, changeTab: this.changeTab };

        return (
            <TabsContext.Provider value={contextValue}>
                {children}
            </TabsContext.Provider>
        );
    }
}
```

Составные компоненты

```
function TabList({ children }: TabListProps) {
    return <div className="tab-list">{children}</div>;
}

function TabPanels({ children }: TabPanelsProps) {
    return <div className="tab-panels">{children}</div>;
}
```

Составные компоненты

```
function Tab({ children, id }: TabProps) {
    return (
        <TabsContext.Consumer>

            {context => {
                function changeTab() {
                    context.changeTab(id);
                }

                return (
                    <div className="tab" onClick={changeTab}>
                        {children}
                    </div>
                );
            }}
        </TabsContext.Consumer>
    );
}
```

Составные компоненты

```
function TabPanel({ children, id }: TabPanelProps) {
    return (
        <TabsContext.Consumer>
            {context => context.activeTabId === id
                ? <div className="tab-panel">{children}</div>
                : null
            }
        </TabsContext.Consumer>
    );
}
```

Составные компоненты

```
function SimpleTabs({ tabs }: SimpleTabsProps) {
    return (
        <Tabs>
            <TabList>
                {tabs.map(tab => (
                    <Tab id={tab.label}>{tab.label}</Tab>
                ))}
            </TabList>
            <TabPanels>
                {tabs.map(tab => (
                    <TabPanel id={tab.label}>{tab.description}</TabPanel>
                ))}
            </TabPanels>
        </Tabs>
    );
}
```

Обработка ошибок

Error Boundary

Компонент, который отлавливает ошибки в любом месте поддерева компонентов и реагирует на них

Error Boundary

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

class ErrorBoundary extends Component {

    state = {
        hasError: false
    }

    static getDerivedStateFromError(error) { // Здесь можно обновить state,
        return { hasError: true };           // чтобы отобразить запасной UI
    }

    componentDidCatch(error, info) {
        logError(error, info); // Здесь можно залогировать ошибку
    }

    ...
}
```

Error Boundary

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

class ErrorBoundary extends Component {

    state = {
        hasError: false
    }

    static getDerivedStateFromError(error) { // Здесь можно обновить state,
        return { hasError: true };           // чтобы отобразить запасной UI
    }

    componentDidCatch(error, info) {
        logError(error, info); // Здесь можно залогировать ошибку
    }

    ...
}
```

Error Boundary

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

class ErrorBoundary extends Component {

    state = {
        hasError: false
    }

    static getDerivedStateFromError(error) { // Здесь можно обновить state,
        return { hasError: true };           // чтобы отобразить запасной UI
    }

    componentDidCatch(error, info) {
        logError(error, info); // Здесь можно залогировать ошибку
    }

    ...
}
```

Error Boundary

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

class ErrorBoundary extends Component {
    ...

    render() {
        // В случае ошибки можно отобразить запасной UI
        if (this.state.hasError) {
            return <h1>Что-то пошло не так</h1>;
        }

        return this.props.children;
    }
}
```

Error Boundary

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


import ErrorBoundary from './error-boundary';

class App extends Component {
    render() {
        return (
            <ErrorBoundary>
                ...
            </ErrorBoundary>
        );
    }
}
```

Не будут отловлены

Ошибки в обработчиках событий

Ошибки в коде самого Error Boundary компонента

Ошибки в асинхронном коде

React Developer Tools

Yandex Browser, Chrome extension

Firefox extension


NEXT

Next.js

```
$ mkdir notes-app && cd notes-app
$ npm install --save next react react-dom
```

```
// package.json
{
    "scripts": {
        "dev": "next",
        "build": "next build",
        "start": "next start"
    }
}
```

```
// ./pages/index.js
export default function IndexPage() {
    return <h1>Welcome to next.js!</h1>;
}
```

```
$ npm run dev
DONE  Compiled successfully span 1773ms
 > Ready on http://localhost:3000
```

Next.js

Свой сервер

```
import { parse } from 'url';


import next from 'next';
import express from 'express';

const server = express();
const app = next({ dev: process.env.NODE_ENV !== 'production' });

const requestHandler = app.getRequestHandler();

app.prepare().then(() => {
    server
        .get('/notes', (req, res) => app.render(req, res, '/index'))
        .get('/notes/:note', (req, res) => app.render(req, res, '/note'))
        .get('*', (req, res) => requestHandler(req, res, parse(req.url, true)))
        .listen(3000, () => console.log('Listening on http://localhost:3000'));
});
```

Next.js

Свой сервер

```
import { parse } from 'url';


import next from 'next';
import express from 'express';

const server = express();
const app = next({ dev: process.env.NODE_ENV !== 'production' });

const requestHandler = app.getRequestHandler();

app.prepare().then(() => {
    server
        .get('/notes', (req, res) => app.render(req, res, '/index'))
        .get('/notes/:note', (req, res) => app.render(req, res, '/note'))
        .get('*', (req, res) => requestHandler(req, res, parse(req.url, true)))
        .listen(3000, () => console.log('Listening on http://localhost:3000'));
});
```

Next.js

Свой сервер

```
import { parse } from 'url';


import next from 'next';
import express from 'express';

const server = express();
const app = next({ dev: process.env.NODE_ENV !== 'production' });

const requestHandler = app.getRequestHandler();

app.prepare().then(() => {
    server
        .get('/notes', (req, res) => app.render(req, res, '/index'))
        .get('/notes/:note', (req, res) => app.render(req, res, '/note'))
        .get('*', (req, res) => requestHandler(req, res, parse(req.url, true)))
        .listen(3000, () => console.log('Listening on http://localhost:3000'));
});
```

Next.js

Свой сервер

```
import { parse } from 'url';


import next from 'next';
import express from 'express';

const server = express();
const app = next({ dev: process.env.NODE_ENV !== 'production' });

const requestHandler = app.getRequestHandler();

app.prepare().then(() => {
    server
        .get('/notes', (req, res) => app.render(req, res, '/index'))
        .get('/notes/:note', (req, res) => app.render(req, res, '/note'))
        .get('*', (req, res) => requestHandler(req, res, parse(req.url, true)))
        .listen(3000, () => console.log('Listening on http://localhost:3000'));
});
```

Next.js

Очень быстрый старт

Множество точек для расширения и кастомизации

Активное развитие и поддержка

Документация React

Context

Error boundaries

Компоненты высшего порядка

Render Props

Next.js

React Router

React Component Patterns