ReactJS Essential

Роутинг и ES2015

ReactJS Essential

Поршнева Екатерина
Front-End разработчик

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Доступ можно получить через руководство вашего учебного центра

Проверьте как Вы усвоили данный материал на TestProvider.com

Тема

Роутинг и ES2015

План урока

1. Некоторые возможности ES2015 (ES6)
2. CSS препроцессоры
3. Роутинг
4. Контекст (context)

ES2015

ES2015
let

```
var x = 'foo';

if (true) {
    var x = 'bar';

    console.log(x); // bar
}

console.log(x); // bar
```

```
let x = 'foo';

if (true) {
    let x = 'bar';

    console.log(x); // bar
}

console.log(x); // foo
```

let имеет блочную видимость

ES2015
const

```
const x = 'foo';
x = 'bar'; // error
```

const во всем аналогичен let, но имеет неизменяемое значение

ES2015
деструктуризация массивов

```
let array = [1, 2, 3, 4, 5];

let [a, b] = array;

console.log(a); // 1
console.log(b);  // 2

let [, , c] = array;

console.log(c); // 3
```

ES2015
spread для массивов

```
let array = [1, 2, 3, 4, 5];
let [x, ...rest] = array;

console.log(x); // 1
console.log(rest); // [2, 3, 4, 5]
```

в текущем стандарте доступны только для массивов, для того, чтобы использовать аналогичный синтаксис для объектов, нужно использовать плагин для babel

ES2015
деструктуризация объектов

```
let me = {
    name: 'Kateryna',
    age: 20,
    gender: 'female',
    city: 'Kyiv'
};

let { name, city } = me;

console.log(name); // Kateryna
console.log(city); // Kyiv
```

ES2015
значения по умолчанию

```
let { name = 'Noname', age = 0 } = {};

console.log(name); // Noname
console.log(age); // 0
```

ES2015
стрелочные функции

```
let inc = x => x+1;
// Аналогично с
// let inc = function(x) { return x + 1; };

console.log( inc(1) ); // 2

let sum = (a, b) => a + b;
// Аналогично с
// let inc = function(a, b) { return a + b; };

console.log( sum(5, 2) ); // 7
```

короткая запись для функций

ES2015
стрелочные функции

```
let sayHello = () => 'Hello!';

console.log( sayHello() ); // Hello

let sayHello = () => {
    let name = 'Katya';
    return `Hello, ${name}!`;
}

console.log( sayHello() ); // Hello, Katya!
```

запись при отсутствии аргументов

ES2015
стрелочные функции

```
let a = [
  "Hydrogen",
  "Helium",
  "Lithium",
  "Beryllium"
];

let a2 = a.map( s => s.length );
// Аналогично с
// let a2 = a.map(function(s){ return s.length });
```

очень удобно их использовать в качестве callback'ов

ES2015
стрелочные функции

```
function Person1() {
    // записываем контекст в переменную, чтобы
    // он был доступен внутри вложенных функций
    let self = this;
    self.age = 0;

    let life = setInterval(function growUp() {
        self.age++;
    }, 100);

    self.die = function() {
        clearInterval(life);
    };
}

let person = new Person1();
console.log('Person1', person.age); // 0

setTimeout(() => {
    console.log('Person1', person.age); // 9
    person.die();
}, 1000);
```

```
function Person2() {
    this.age = 0;

    let life = setInterval(() => {
        this.age++;
    }, 100);

    this.die = () => {
        clearInterval(life);
    };
}

let person = new Person2();
console.log('Person2', person.age); // 0

setTimeout(() => {
    console.log('Person2', person.age); // 9
    person.die();
}, 1000);
```

стрелочные функции сохраняют контекст

ES2015
аргументы по умолчанию

```
let sayHello = function(name = 'Everyone') {
    console.log(`Hello ${name}!`);
}

sayHello(); // Hello Everyone!
sayHello('Vasya'); // Hello Vasya!
```

ES2015
spread для аргументов функций

```
let sayHello = function(...people) {
    console.log(`Hello ${people.join(', ')}!`);
}

sayHello('Katya', 'Vasya', 'Andrew'); // Hello Katya, Vasya, Andrew

let numbers = [1, 5, 7, 10, 3, 9, 16];
let max = Math.max(...numbers);

console.log(max); // 16
```

ES2015
строки

```
console.log(`This
    is
    multiline
    string`);

let name = 'Katya';

console.log(`Hello, ${name}!`); // Hello, Katya!

let a = 2;
let b = 5;

console.log(`${a} + ${b} = ${a + b}`); // 2 + 5 = 7
```

ES2015
сокращения при описании объектов

```
let name = 'Katya';

let me = {
    name,
    city: 'Kyiv'
};

console.log(me); // { name: 'Katya', city: 'Kyiv' }

let propName = 'name';

let user = {
    [propName]: 'Katya'
};

console.log(user.name); // Katya
```

ES2015
классы

```
class Point {
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }

    toString() {
        return `(${this.x}, ${this.y})`;
    }
}

let point = new Point(2, 1);
console.log( point.toString() ); // (2, 1)
```

ES2015
классы: наследование

```
class ColorPoint extends Point {
    constructor(x, y, color) {
        super(x, y);
        this.color = color;
    }
    toString() {
        return `${super.toString()} in ${this.color}`;
    }
}

let colorPoint = new ColorPoint(4, 2, 'red');
console.log( colorPoint.toString() ); // (4, 2) in red
```

ES2015
классы: статические методы

```
class Button {
    static getElementType() {
        return 'button';
    }
    constructor(name) {
        this.color = color;
    }
    click() {
        console.log('Button clicked!');
    }
}

console.log( Button.getElementType() ); // button
```

ES2015
классы: геттеры и сеттеры

```
class User {
    constructor(firstName, lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    // геттер
    get fullName() {
        return `${this.firstName} ${this.lastName}`;
    }

    // сеттер
    set fullName(newValue) {
        [this.firstName, this.lastName] = newValue.split(' ');
    }
};

let user = new User('Вася', 'Пупков');
console.log( user.fullName ); // Вася Пупков

user.fullName = 'Иван Петров';
console.log( user.fullName ); // Иван Петров
```

ES2015
модули

```
// exporter.js
export function sayHello() {
    console.log('Hello!')
}

export var object = { a: 1, b: 2 };

//importer.js
import { sayHello, object as namedObject } from './exporter';

sayHello(); // Hello!
console.log( namedObject ); // {a: 1, b: 2}
```

ES2015
модули

```
// exporter.js
export default function foo() {
    console.log('foo');
}

//importer.js
import customName from './export-default';

customName(); // foo
```

ES2015
модули

```
export var variable;
export const CONSTANT = 0;
export let scopedVariable = 20;
export function func(){};
export class Foo {};

export default const object = {};
```

Оператор **export** позволяет экспортировать сущности модуля, чтобы они были доступны из других модулей. По сути, это обычное объявление переменной, функции или класса, с ключевым словом "export" перед ним

ES2015
модули

```
import <любое имя> from '<путь к модулю>';

// Импорт всех экспортируемых свойств
import * as lib from './lib';

// Включение модуля без импорта
import 'jquery';

// Импорт по умолчанию
import $ from 'jquery';

// Именованый импорт
import { $ } from 'jquery';
import { $ as jQuery } from 'jquery';

// Импорт нескольких експртируемых свойств
import { a, b, c as myC } from './lib';
```

Оператор **import** позволяет импортировать в модуль экспортированные значения из другого модуля

CSS препроцессоры

CSS препроцессоры

CSS препроцессоры

1. переменные
2. импорт
3. миксины (примеси)
4. вложенные селекторы
5. функции
6. вложенные медиа-запросы
7. циклы

CSS препроцессоры: переменные

Less

```less
@font-size: 16px;

div {
    font-size: @font-size;
}
```

Css

```css
div {
    font-size: 16px;
}
```

http://lesscss.org/features/

CSS препроцессоры: вложенность

Less

```less
@link-color: #999;
@link-hover: #229ed3;

ul {
    margin: 0;

    li {
        float: left;
    }

    a {
        color: @link-color;

        &:hover {
            color: @link-hover;
        }
    }
}
```

Css

```css
ul { margin: 0; }
ul li { float: left; }
ul a { color: #999; }
ul a:hover { color: #229ed3; }
```

http://lesscss.org/features/

CSS препроцессоры: примеси

Less

```
.bordered {
  border-top: dotted 1px #333;
  border-bottom: solid 2px #333;
}
.article {
  .bordered;
  color: #443d3d;
}
```

Css

```
.bordered {
  border-top: dotted 1px #333;
  border-bottom: solid 2px #333;
}
.article {
  border-top: dotted 1px #333;
  border-bottom: solid 2px #333;
  color: #443d3d;
}
```

http://lesscss.org/features/

CSS препроцессоры: примеси

Less

```less
.bordered (@width) {
    border: @width solid #ddd;

    &:hover {
        border-color: #999;
    }
}

h1 {
    .bordered(5px);
}
```

Css

```css
h1 { border: 5px solid #ddd; }
h1:hover { border-color: #999; }
```

http://lesscss.org/features/

CSS препроцессоры: функции

Less

```
saturate(@color, @amount)
desaturate(@color, @amount)
lighten(@color, @amount)
darken(@color, @amount)
fadein(@color, @amount)
fadeout(@color, @amount)
fade(@color, @amount)
spin(@color, @amount)
mix(@color1, @color2, @weight)
grayscale(@color)
contrast(@color)
```

http://lesscss.org/features/

CSS препроцессоры: математические операции

Less

```
1cm * 1em => 1cm * 1em
2in * 3in => 6in
(1cm / 1em) * 4em => 4cm
2in + 3cm + 2pc => 3.514in
3in / 2in => 1.5in
```

http://lesscss.org/features/

CSS препроцессоры: импорт

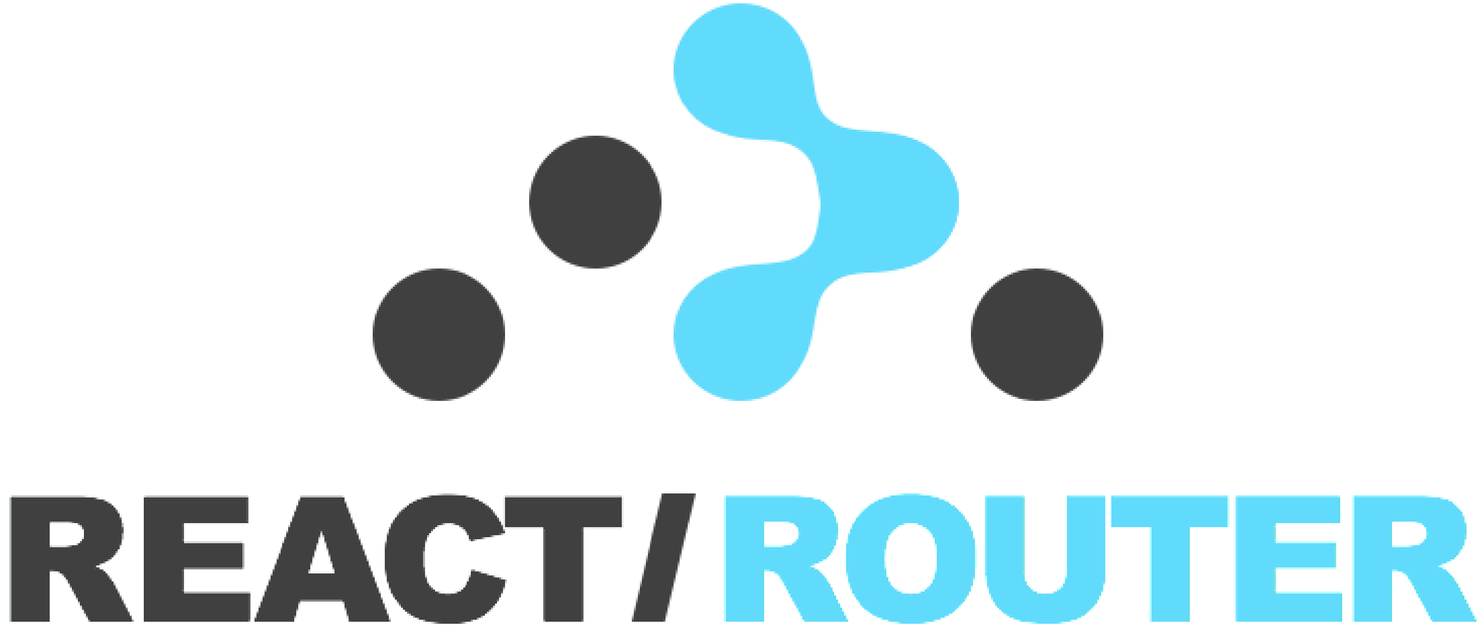
Less

```
@import "library"
@import "mixins/mixin.less"
@import "reset.css"
```

http://lesscss.org/features/

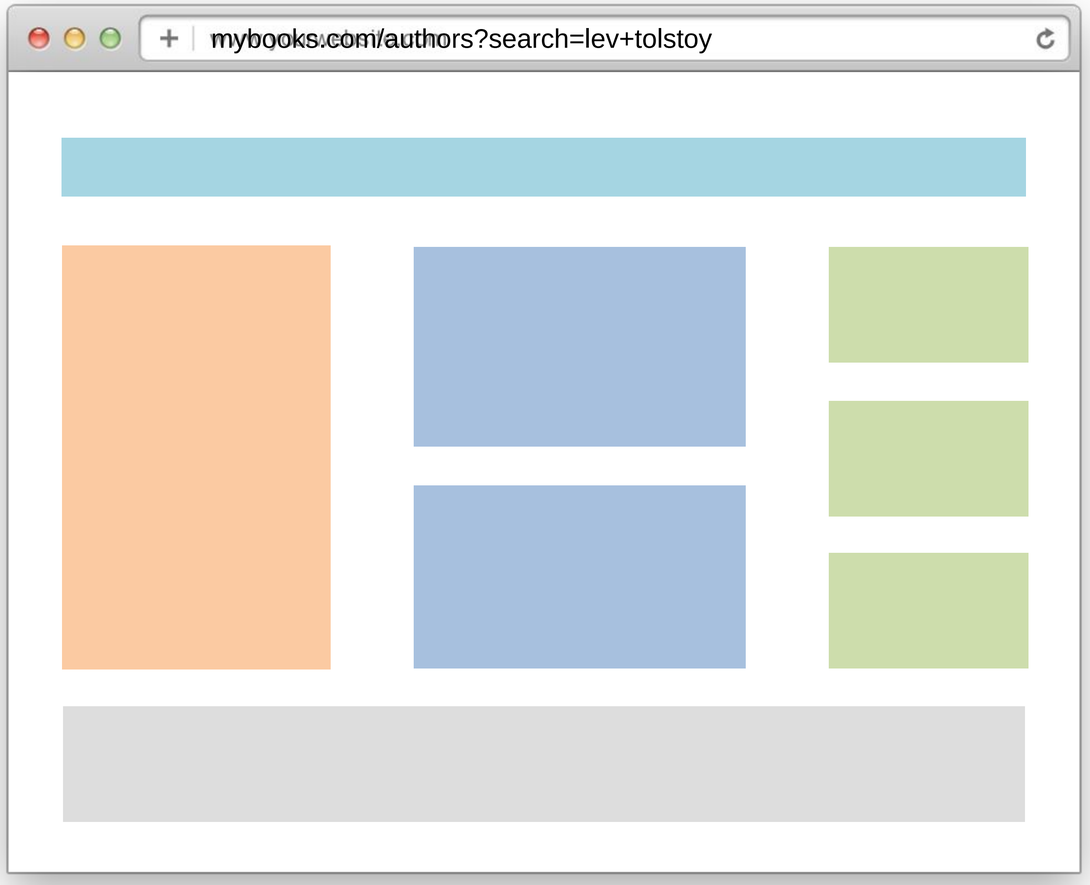
Роутинг (маршрутизация)

Роутинг (маршрутизация)

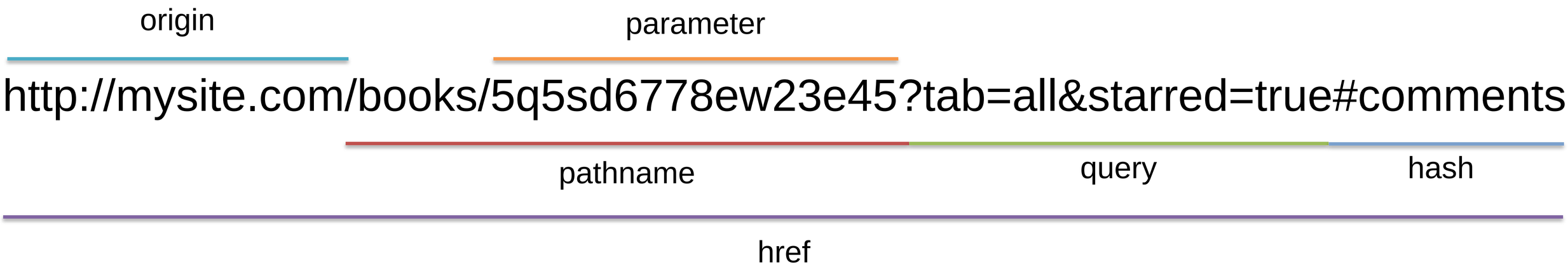


https://github.com/rackt/react-router

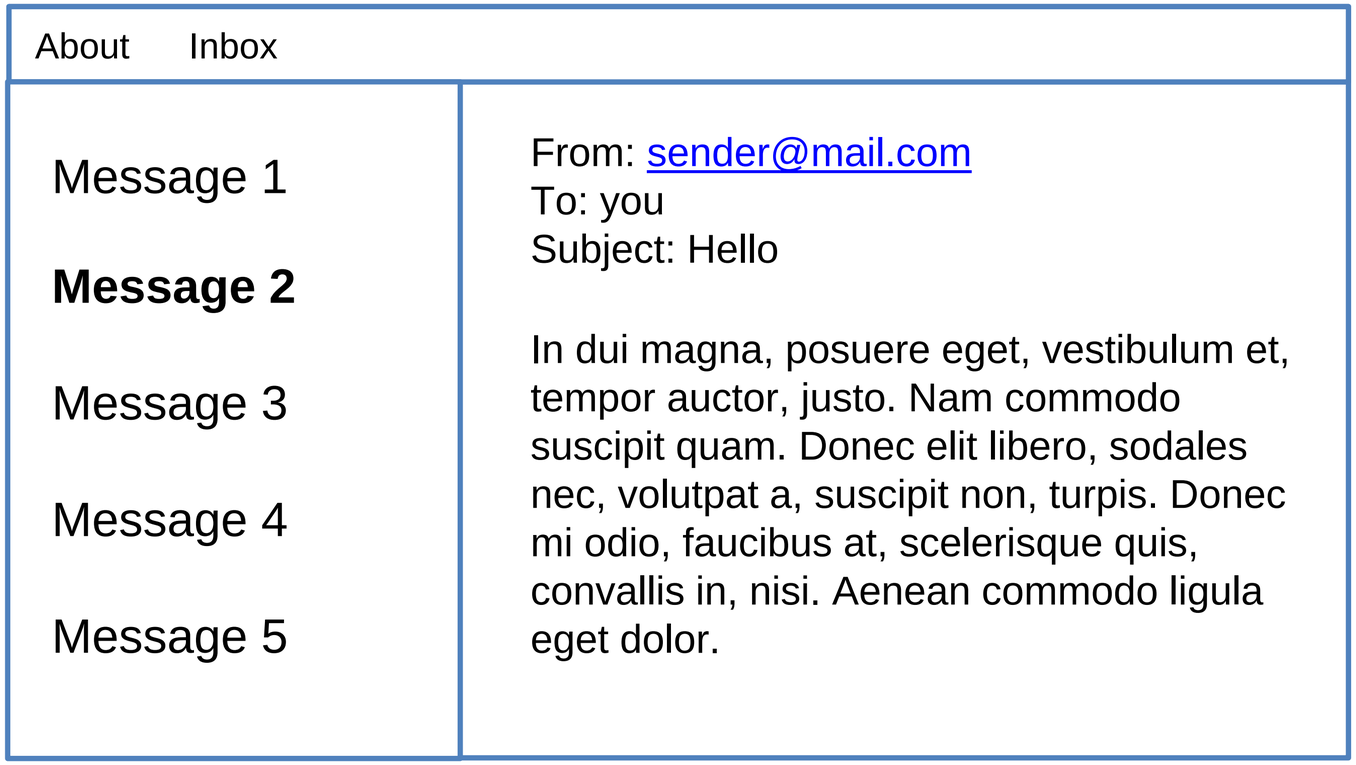
Роутинг (маршрутизация)



Структура URL

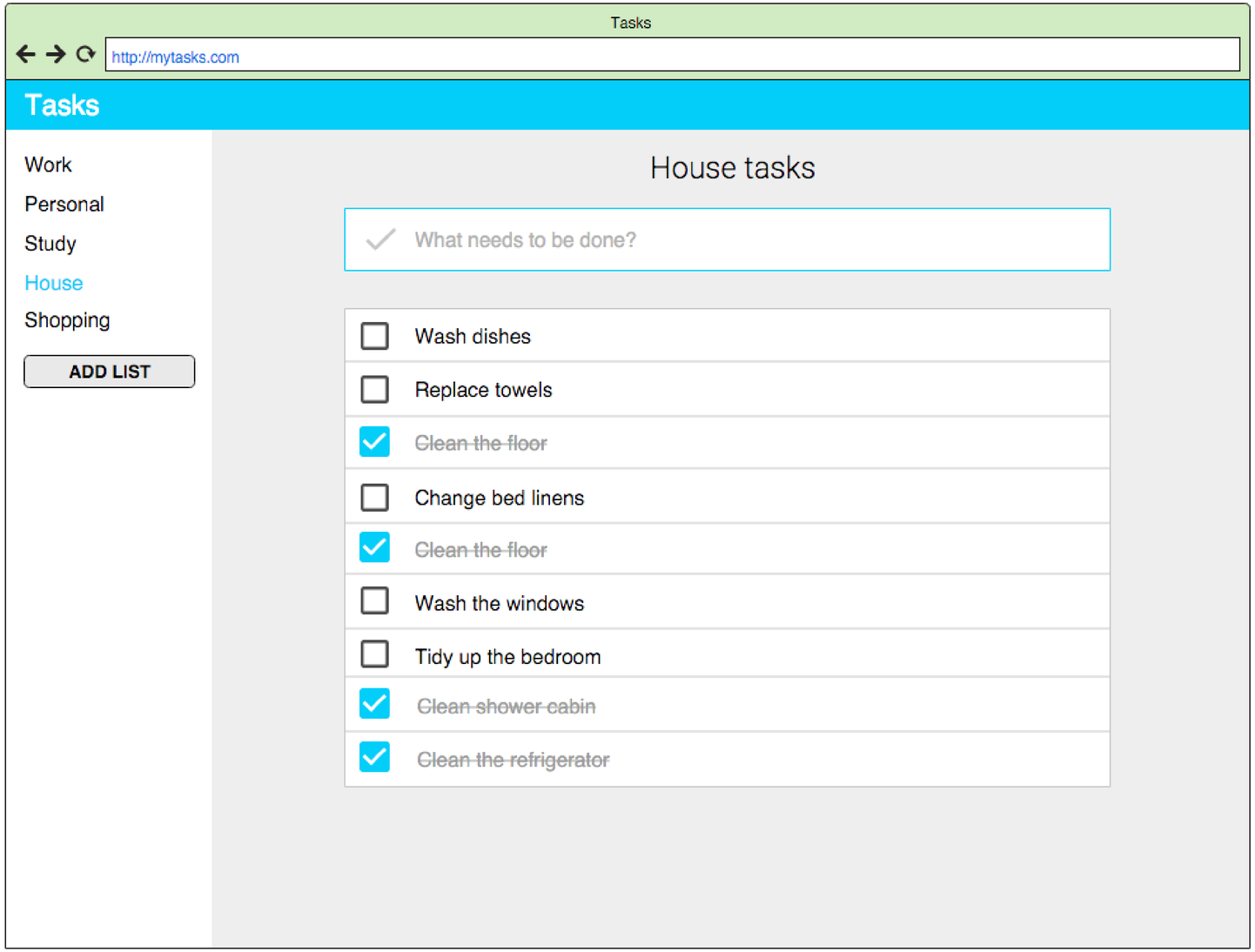


Роутинг (маршрутизация)



Роутинг (маршрутизация)

URL	Компоненты
/about	App -> AboutPage
/inbox	App -> InboxPage
/inbox/messages/:id	App -> InboxPage -> Message


Tasks
http://mytasks.com
Tasks
Work
Personal
Study
House
Shopping
ADD LIST
House tasks
What needs to be done?
Wash dishes
Replace towels
Clean the floor
Change bed linens
Clean the floor
Wash the windows
Tidy up the bedroom
Clean shower cabin
Clean the refrigerator

Валидация параметров

Валидация параметров

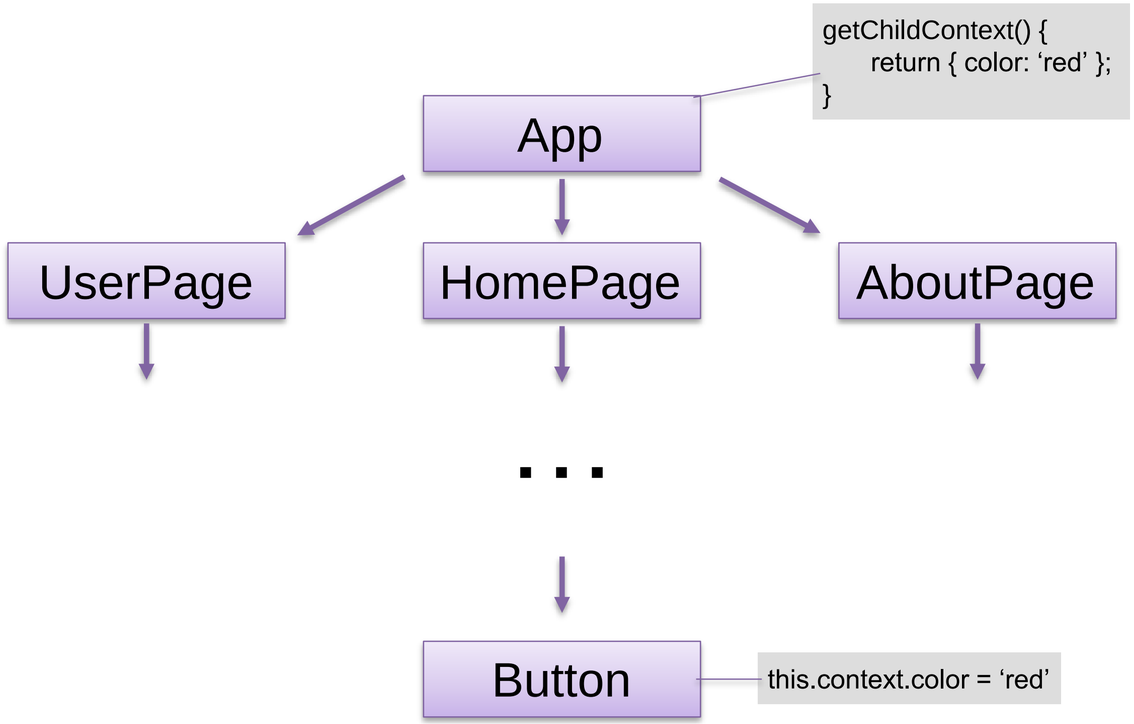
```
const Article = React.createClass({
    propTypes: {
        title: React.PropTypes.string.isRequired,
        text: React.PropTypes.string.isRequired,
        pictureURL: React.PropTypes.string,
        author: React.PropTypes.shape({
            name: React.PropTypes.string,
            avatarURL: React.PropTypes.string,
            numberOfArticles: React.PropTypes.number
        }),
        type: React.PropTypes.oneOf(['education', 'entertainment']),
        comments: React.PropTypes.arrayOf(React.PropTypes.object),
        numberOfLikes: React.PropTypes.number
    },

    // ...
});
```

https://facebook.github.io/react/docs/reusable-components.html

Контекст

Контекст



https://facebook.github.io/react/docs/context.html

Контекст

```
var Message = React.createClass({
  render: function() {
    return (
      <div>
        {this.props.text} <Button>Delete</Button>
      </div>
    );
  }
});
```

```
var Button = React.createClass({
  contextTypes: {
    color: React.PropTypes.string
  },
  render: function() {
    return (
      <button style={{background: this.context.color}}>
        {this.props.children}
      </button>
    );
  }
});
```

Материалы урока

https://github.com/krambertech/react-essential-course

Смотрите наши уроки в видео формате

ITVDN.com

Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

Все курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics

ITVDN
IT VIDEO DEVELOPERS NETWORK

Проверка знаний

TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

TestProvider

Q&A

Информационный видеосервис для разработчиков программного обеспечения