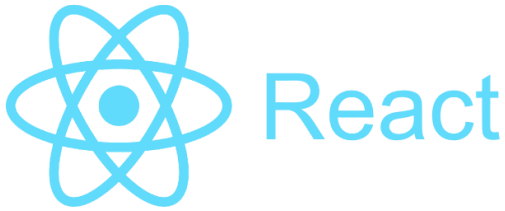


React Essential

Создание списка заметок

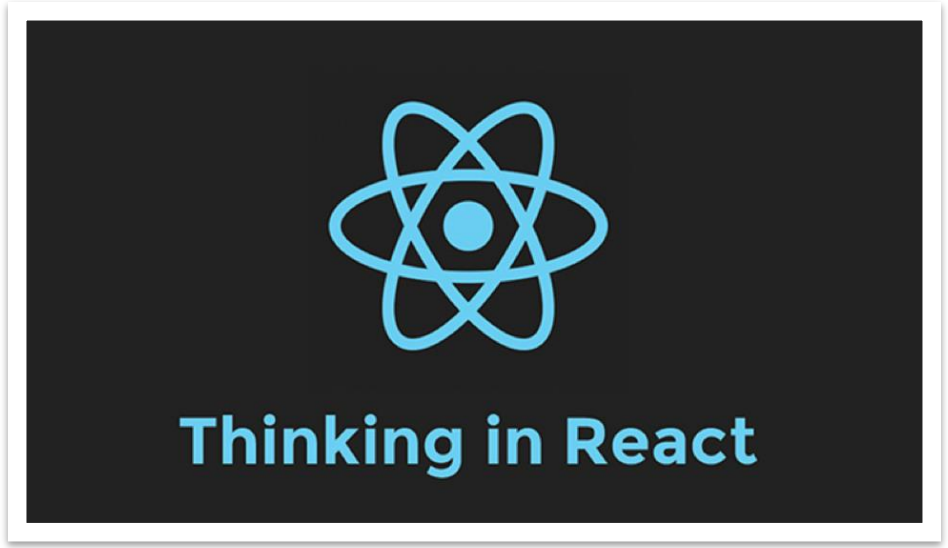
Introduction

Муляк Дмитрий

Front-end developer at GlobalTechMakers

dmitriymuliak

dmitriymuliak



Тема

Создание списка заметок

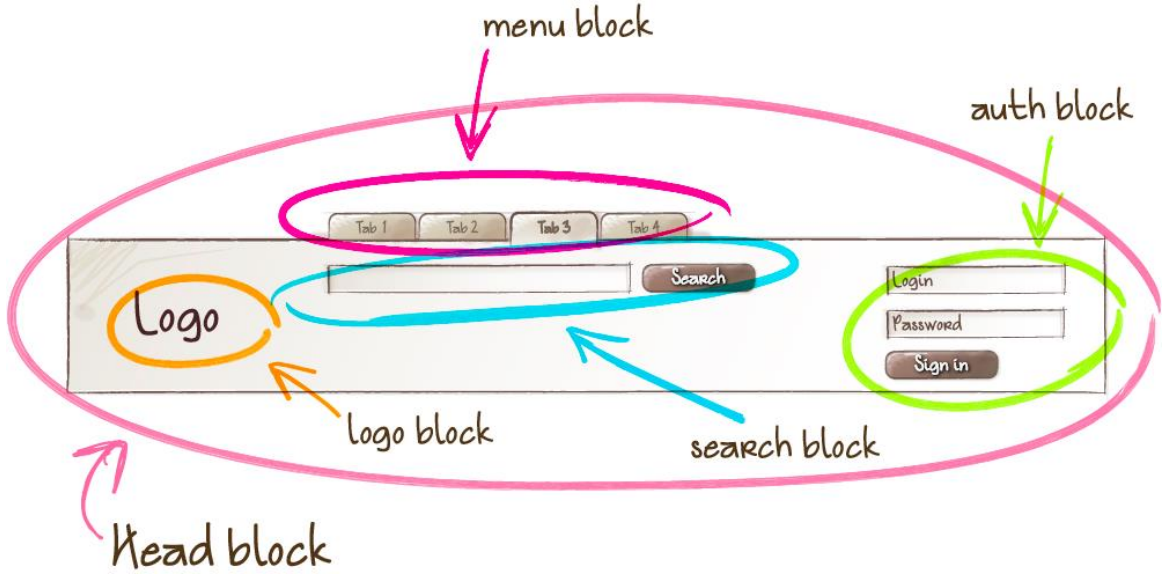
Создание списка заметок

1. Разбиение на компоненты
2. Верстка списка
3. Компонент Кнопка
4. Добавление и удаление элементов

Разбиение на компоненты

Прежде чем начинать создавать комплексный компонент, нужно разбить его на более мелкие части.

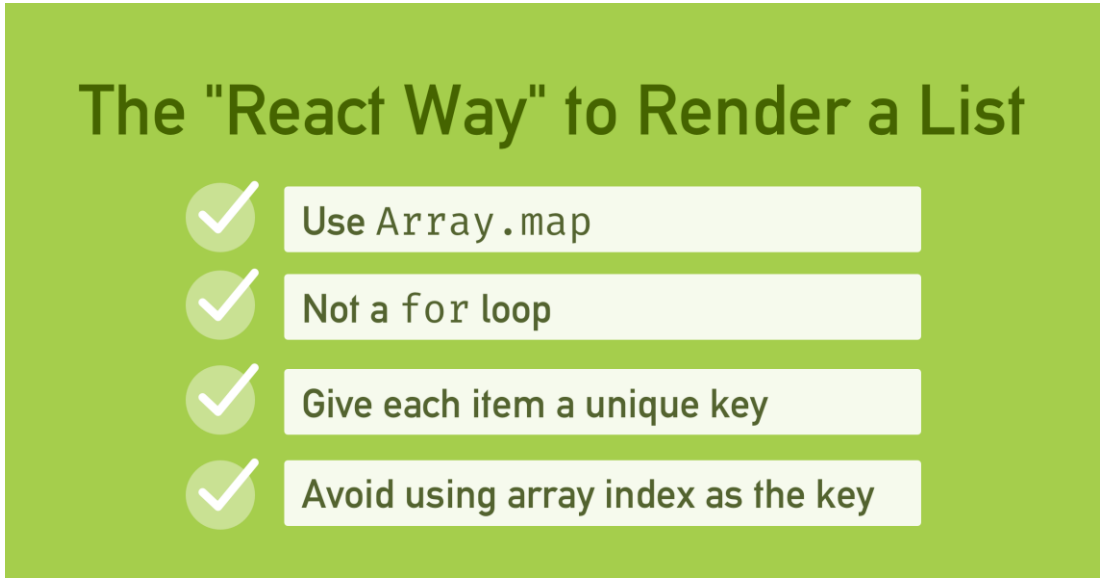
- Понять какие компоненты нуждаются в state, а каким будет достаточно props.
- Вынести состояние на уровень выше если в нем нуждается несколько компонентов.



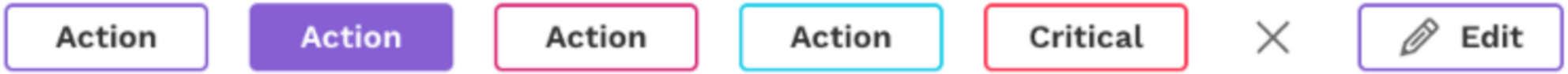
Верстка списка

При создании списков нужно не забывать несколько простых правил и рекомендаций:

- Использовать Array.map вместо for loop.
- Присвоить каждому элементу списка атрибут «key» === uid
- Избегать использование индекса как ключа.
- Не создавать лишние элементы обертки если рендерите массив.



Компонент Кнопка



Компонент кнопка может быть достаточно функциональным и опциональным.

Наш компонент должен иметь несколько возможностей:

- принимать тип нужной иконки;
- принимать любой контент, который мы захотим;
- принимать свойство «color» и «size» для стандартной стилизации;
- иметь возможность изменить нужные нам css классы;
- принимать все возможные атрибуты html <button></button>.

Указание типа React-элемента

Использование записи через точку.

Вы можете ссылаться на React-компонент, используя запись через точку. Это удобно, если у вас есть модуль, который экспортирует много React-компонентов.

Названия типов пользовательских компонентов должны начинаться с большой буквы.

К примеру, если MyComponents.DatePicker является компонентом, то вы можете обратиться к нему напрямую, используя запись через точку:

```
import React from 'react';

const MyComponents = {
  DatePicker: function DatePicker(props) {
    return <div>Представьте, что здесь цвет {props.color} виджета выбора даты.</div>;
  }
}

function BlueDatePicker() {
  return <MyComponents.DatePicker color="blue" />;
}
```

Выбор типа во время исполнения

В качестве типа React-элемента нельзя использовать выражение. Если вы хотите использовать выражение, чтобы указать тип элемента, присвойте его в переменную, начинающуюся с заглавной буквы.

```
import { PhotoStory, VideoStory } from './stories';

const components = {
  photo: PhotoStory,
  video: VideoStory
};

function Story(props) {
  // Неправильно! JSX-тип не может являться выражением
  return <components[props.storyType] story={props.story} />;
}
```

Это подходит для рендера компонентов в зависимости от ваших пропсов:

Чтобы исправить это, мы присвоим тип в переменную, начинающуюся с заглавной буквы:

```
import { PhotoStory, VideoStory } from './stories';

const components = {
  photo: PhotoStory,
  video: VideoStory
};

function Story(props) {
  // Правильно! JSX-тип может являться переменной, названной с заглавной буквы
  const SpecificStory = components[props.storyType];
  return <SpecificStory story={props.story} />;
}
```

Атрибуты расширения

Если у вас уже есть пропсы внутри объекта props и вы хотите передать их в JSX, вы можете использовать оператор расширения .... Эти два компонента эквивалентны:

```
function App1() {
  return <Greeting firstName="Иван" lastName="Иванов" />;
}

function App2() {
  const props = {firstName: 'Иван', lastName: 'Иванов'};
  return <Greeting {...props} />;
}
```

Вы так же можете выбрать конкретные пропсы, которые ваш компонент будет использовать, передавая все остальные пропсы с помощью оператора расширения:

```
const Button = props => {
  const { kind, ...other } = props;
  const className = kind === "primary" ? "PrimaryButton" : "SecondaryButton";
  return <button className={className} {...other} />;
};

const App = () => {
  return (
    <div>
      <Button kind="primary" onClick={() => console.log("Кнопка нажата!")}>
        Привет, мир!
      </Button>
    </div>
  );
};
```

Функции как дочерние компоненты

Обычно JavaScript-выражения, вставленные в JSX, будут приведены к строке, React-элементу или списку из всего этого. Тем не менее, props.children работает так же, как и любой другой проп, поэтому в него можно передавать любые типы данных, а не только те, которые React знает как рендерить. К примеру, если у вас есть пользовательский компонент, можно было бы передать колбэк в props.children:

Дочерние компоненты, передаваемые пользовательскому компоненту, могут быть чем угодно с тем условием, что компонент преобразует их во что-то, что React сможет понять и отрендерить. Следующий пример редко встречается, но им можно воспользоваться, если необходимо расширить возможности JSX.

```
// Вызывает колбэк numTimes раз для создания повторяющего компонента
function Repeat(props) {
  let items = [];
  for (let i = 0; i < props.numTimes; i++) {
    items.push(props.children(i));
  }
  return <div>{items}</div>;
}

function ListOfTenThings() {
  return (
    <Repeat numTimes={10}>
      {(index) => <div key={index}>Это элемент списка с ключом {index}</div>}
    </Repeat>
  );
}
```

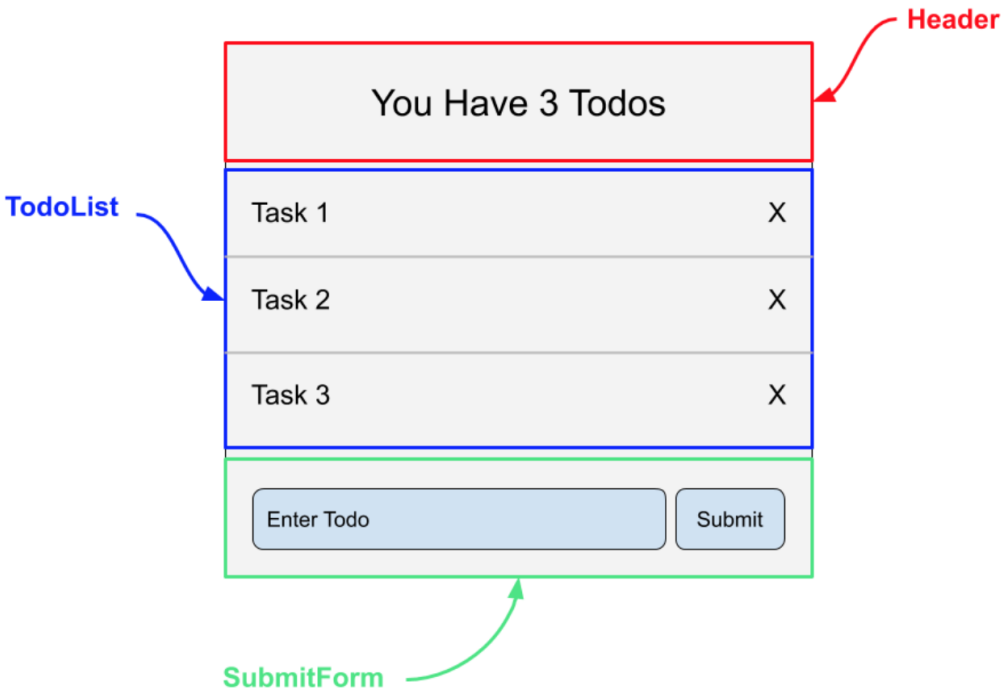
Добавление и удаление элементов

UUID (universally unique identifier) — это стандарт идентификации, используемый в создании программного обеспечения, стандартизированный Open Software Foundation. UUID представляет собой 16-байтный (128-битный) номер.

В шестнадцатеричной системе счисления UUID выглядит как: «d29e9b42-9b97-ab6f-bc0c-2c3f938d10e4».

Если у вас нет возможности взять уникальный id со стороны бэкенда, вы можете использовать несколько подходов:

- использовать библиотеку для генерации uid (к примеру «shortid»)
- использовать singleton для генерации id (зависит от ситуации)
- использовать Date.now() (зависит от ситуации)
- использовать Math.random() + String.replace()(зависит от ситуации)
- использовать Crypto API + String.replace() + Арифметический сдвиг



index - это анти-паттерн

генерация символьных UUID

Проверка знаний

TestProvider.com

Проверьте как Вы усвоили данный материал на TestProvider.com

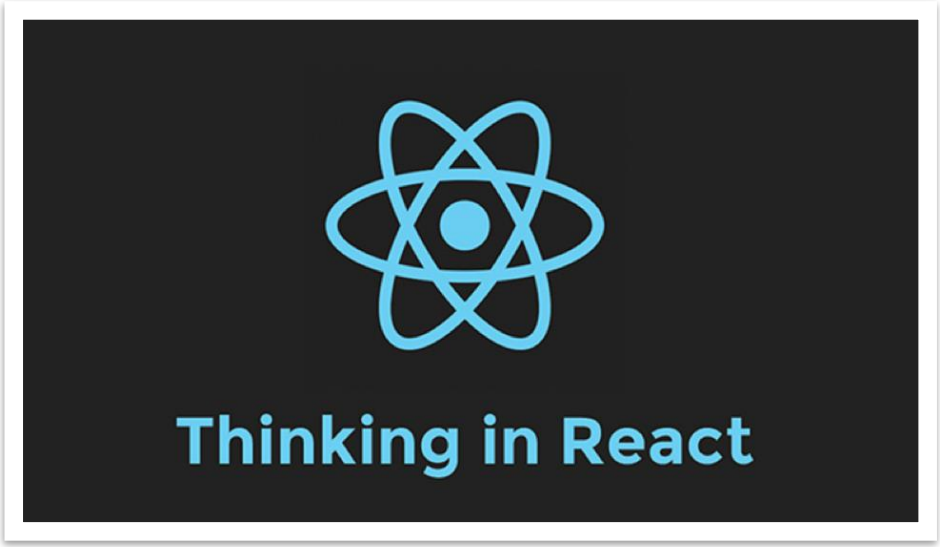
TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Спасибо за внимание! До новых встреч!

Муляк Дмитрий

Front-end Developer at GTM



Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association