МИРЭА – Российский технологический университет
Институт перспективных технологий и индустриального программирования

Frontend & Backend разработка

Лекция 1

Основы ReactJS

Р.В. Шамин
профессор кафедры индустриального программирования

Москва, 2023

Что такое ReactJS

ReactJS - это JavaScript библиотека (фреймворк) для построения веб и иных пользовательских интерфейсов.

ReactJS является кроссплатформенной библиотекой, для работы которой необходим JavaScript и DOM браузера. Можно использовать для мобильных приложений, и десктопных.

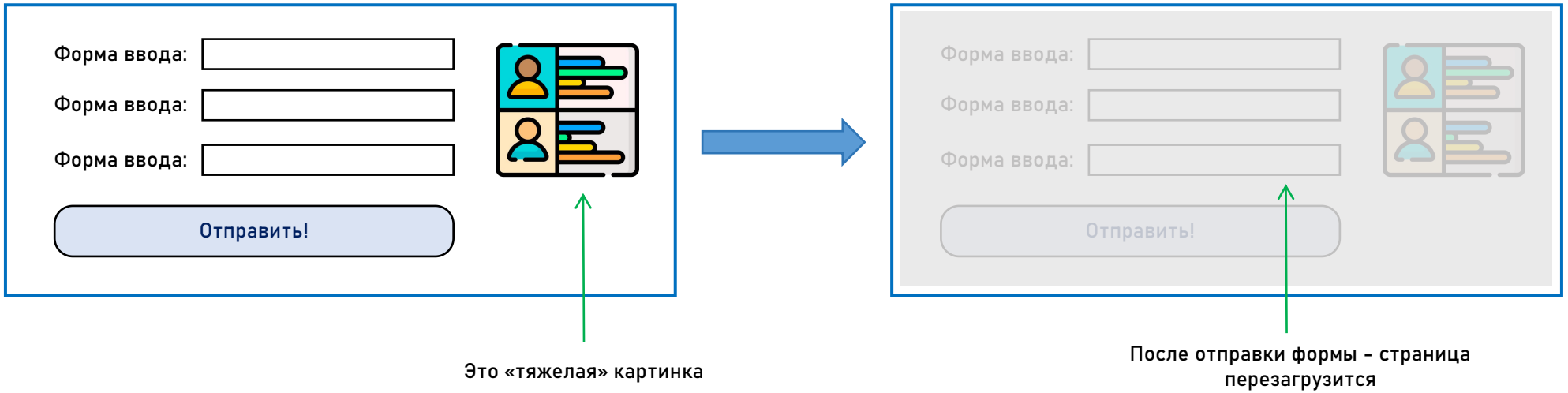
ReactJS основан на иерархии компонентов. Каждый компонент имеет входные параметры (props - “пропсы”) и реализует внутреннюю логику для отбражения (рендеринга), при этом компоненты могут использовать дочерние компоненты, а также иметь сохраняемые данные.

ReactJS использует специальное раширение языка JavaScript, которое называется JSX. JSX - это удобная смесь HTML5 и JavaScript.

Почему ReactJS? Есть много альтернативных фрейморков для фронтенда (Vue, Angular, AngularJS...), ReactJS - самый популярный на 2024 год фреймворк.

А можно не заморачиваться?
Можно создавать веб приложение без фреймворков?

А как делать Ajax?



Установка и запуск ReactJS

Для разработки на ReactJS необходимо установить последнюю версию Node.js

Чтобы создать приложение на React воспользуемся create-react-app. Для этого создадим новую папку и выполним команду (first-app - название приложения):

```
Windows PowerShell
PS C:\React> npm init react-app first-app
```

Через некоторое время будет создана папка first-app:

```
    Каталог: C:\React\first-app


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        05.01.2024     18:14                node_modules
d-----        05.01.2024     18:14                public
d-----        05.01.2024     18:14                src
-a----        05.01.2024     18:14            310 .gitignore
-a----        05.01.2024     18:14         711556 package-lock.json
-a----        05.01.2024     18:14            812 package.json
-a----        05.01.2024     18:14           3359 README.md
```

Папка с модулями React

Папка со статическим контентом

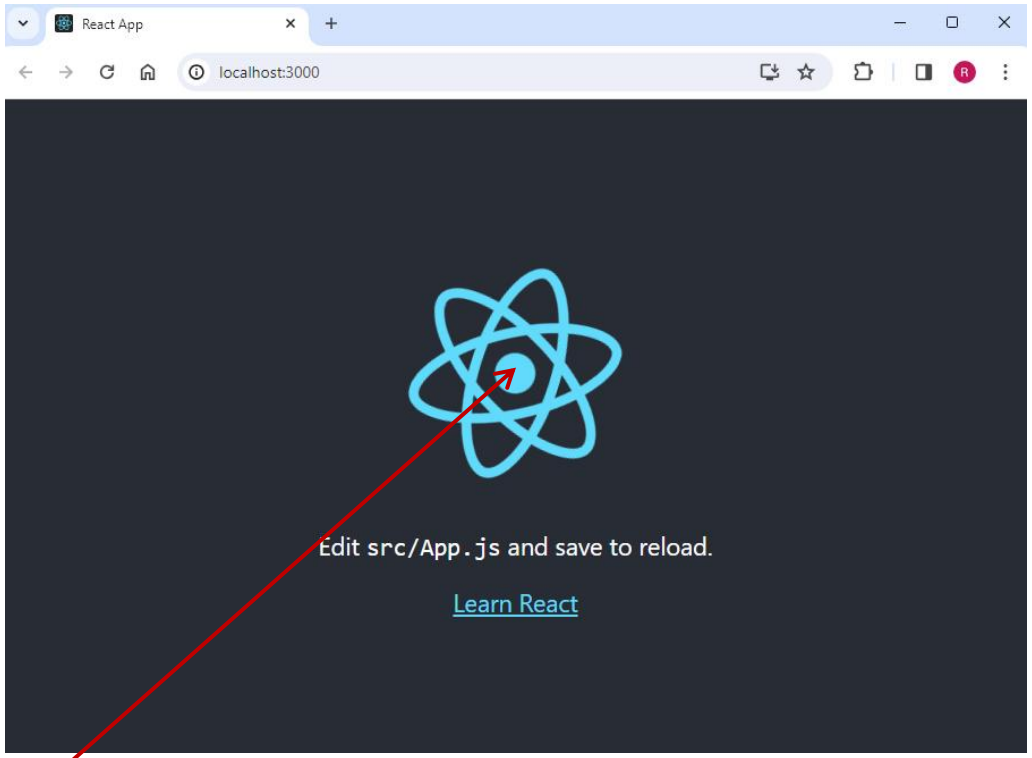
Папка с исходными текстами приложения

Чтобы запустить наше приложение перейдем в папку first-app и выполним команду:

```
Windows PowerShell
PS C:\React\first-app> npm start
```

Запускаем ReactJS

В результате в браузере откроется новое окно (вкладка) с адресом localhost:3000



Это логотип ReactJS, который будет вращаться

Очистим созданный пример и создадим простейшее приложение на ReactJS.

Делаем простейшее приложение ReactJS

В папке public оставляем только файл index.html, из которого оставляем только следующий код:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Это корневой элемент, в котором будут располагаться все компоненты React

В папке src оставляем только три файла App.js, index.js, index.css:

```js
function App() {
  return (
    <div>
      Привет, ReactJS!
    </div>
  );
}

export default App;
```

App.js

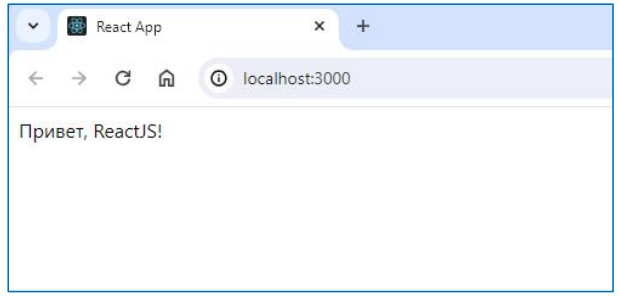
```js
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <App />
);
```

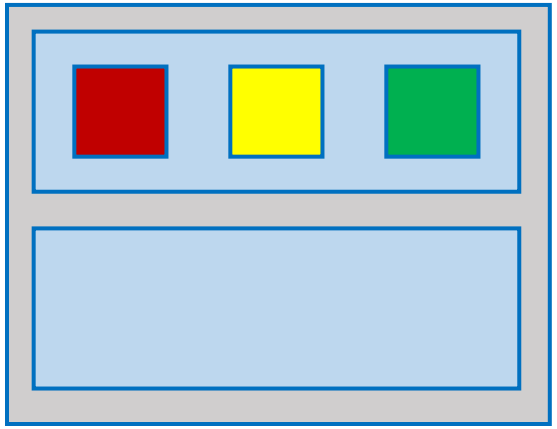
index.js

```css
body {
  font-family: 'Segoe UI', 'Roboto', 'O
}
```

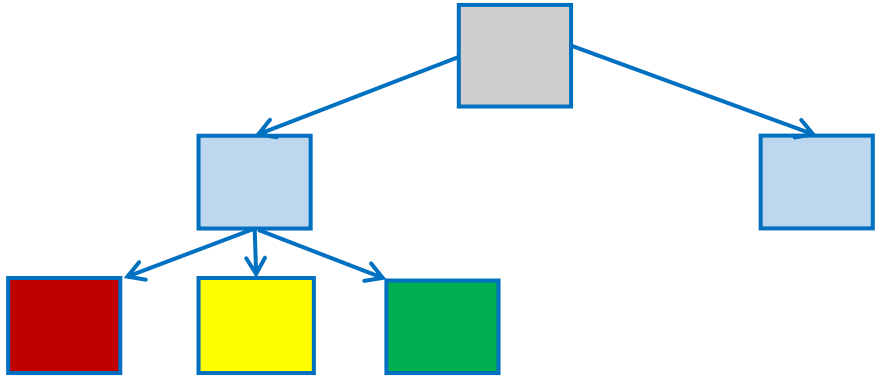
index.css



Компоненты ReactJS



Вложенная структура компонентов



Компоненты ReactJS могут быть реализованы в виде класса или в виде функции. Сейчас более современным и эффективным считается функциональный стиль.

Создадим в папке src папку components, в которой будем размещать исходные тексты наших компонентов. Создадим в этой папке файл Hello.js

```
1  export default function Hello() {
2      return (
3          <div>
4              Привет!
5          </div>
6      )
7  }
```

Компонент - это экспортируемая функция, которая должна возвращать один HTML элемент, в нашем случае
Этот элемент может содержать сколько угодно сложную структуру HTML.
Обратите внимание, что если возвращаем больше одной строки, то все возвращаемые строки оборачиваем в круглые скобки.

```
1  import Hello from "../src/components/Hello";
2
3  export default function App() {
4    return (
5      <div>
6        Привет, ReactJS!
7        <Hello/>
8      </div>
9    );
10 }
```

импортируем компонент

используем компонент

Компонент ReactJS можно понимать как новый пользовательский HTML тег, который однако может иметь свою логику.

Параметры компонентов ReactJS

Компоненты могут принимать параметры. Возможно два способа передачи параметров:
1. Через атрибуты компонента
2. Через тело компонента

```
1  export default function Hello(props) {
2      return (
3          <div>
4              {props.hello}, {props.children}!
5          </div>
6      )
7  }
```

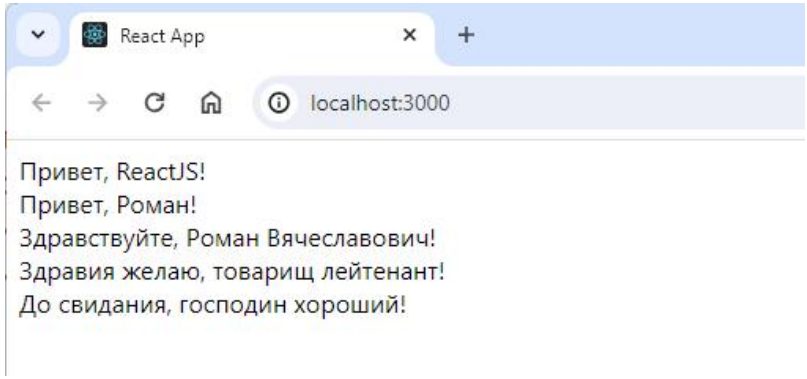
класс, который хранит атрибуты компонента при вызове

поле, в котором значение тела компонента

Заметьте, что при использовании значений мы их оборачиваем в фигурные скобки. Это синтаксис JSX - если внутри HTML мы хотим перейти в JavaScript.

Теперь используем один компонент много раз с различными значениями:

```
1  import Hello from "../src/components/Hello";
2
3  export default function App() {
4    return (
5      <div>
6        Привет, ReactJS!
7        <Hello hello="Привет">Роман</Hello>
8        <Hello hello="Здравствуйте">Роман Вячеславович</Hello>
9        <Hello hello="Здравия желаю">товарищ лейтенант</Hello>
10       <Hello hello="До свидания">господин хороший</Hello>
11     </div>
12   );
13 }
```



Вычисления в компоненте

Компоненты представляют собой функции на JavaScript (JSX) и могут производить различные вычисления любой сложности.

Рассмотрим компонент для вычисления факториала числа, заданного в качестве параметра.

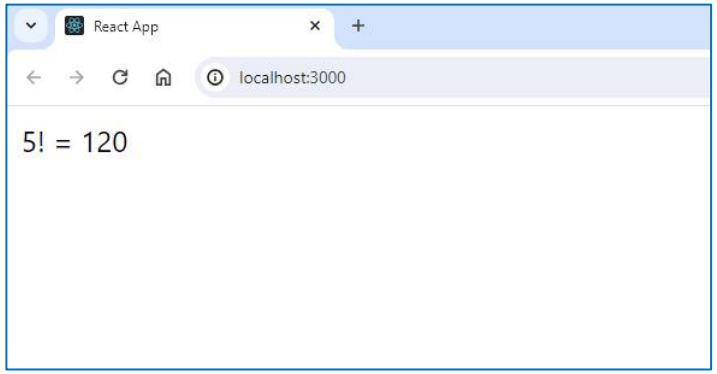
```
// обычная функция для вычисления факториала
function Fact(N) {
    if (N === 1) {
        return 1
    } else {
        return N * Fact(N - 1) // используем рекурсию
    }
}

// компонент
export default function Factorial(props) {
    let N = props.N // получаем число из атрибутов компонента
    let F = Fact(N) // вызываем внешнее вычисление
    return (
        <div>
            {props.N}! = {F}
        </div>
    )
}
```

выводим результат работы компонента

```
import Factorial from "./components/Factorial";

export default function App() {
  return (
    <div>
      <Factorial N="5"/>
    </div>
  );
}
```



События

Компоненты ReactJS могут обрабатывать события аналогично как это делается в JavaScript, но более органично.

Рассмотрим пример дублирования вводимого текста.

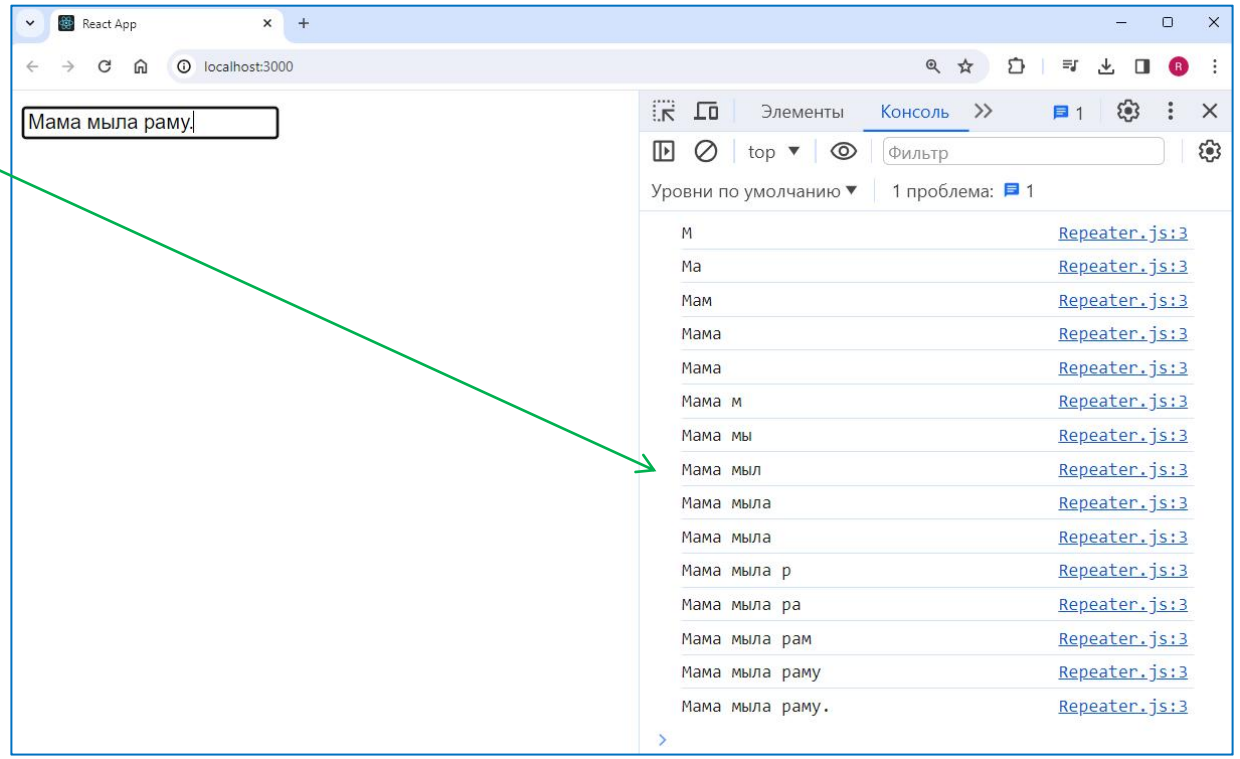
```
1  export default function Repeater() {
2
3      function handlerChange (e) {
4          console.log(e.target.value)
5      }
6
7      return (
8          <div>
9              <input type="text" onChange={handlerChange}/>
10         </div>
11     )
12 }
```

функция для обработки события

выводим вводимый текст на консоль

обратите внимание на название события

здесь указываем функцию, но не вызываем ее

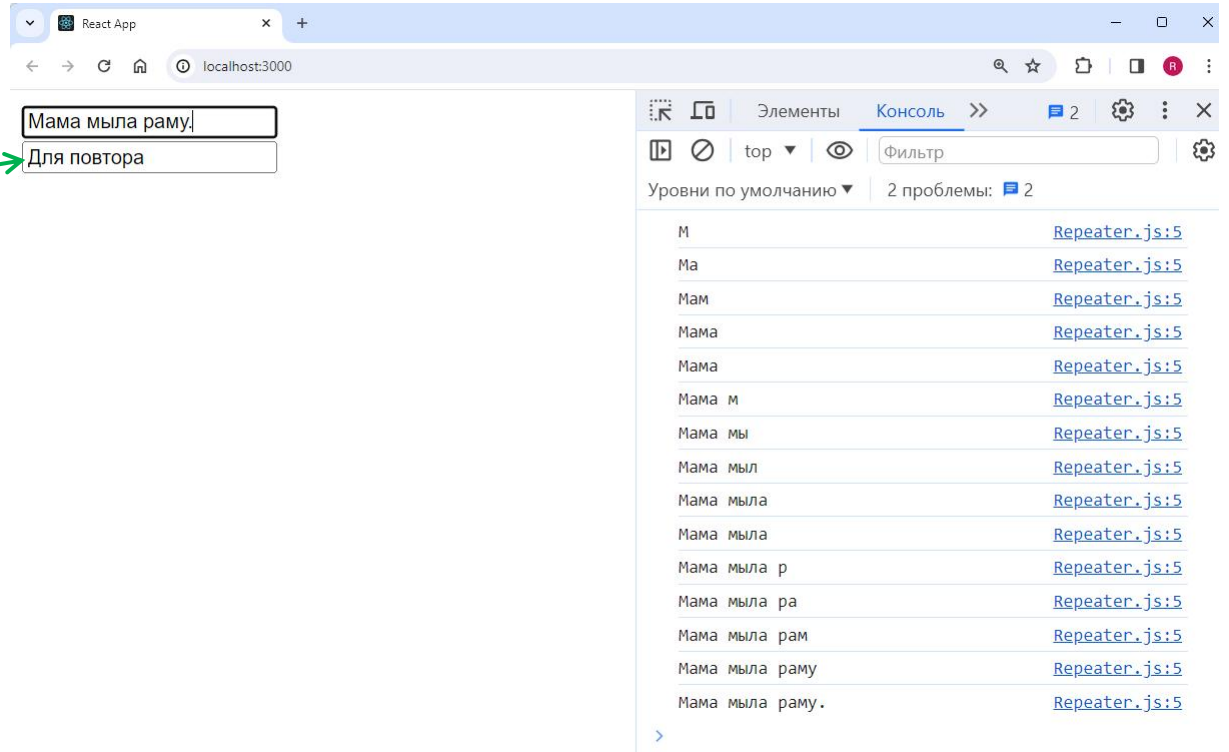


События

Попробуем изменить пример, чтобы текст дублировался на в консоль, а в другой HTML-элемент.

```
1  export default function Repeater() {
2      let text = "Для повтора"
3
4      function handlerChange (e) {
5          console.log(e.target.value)
6          text = e.target.value
7      }
8
9      return (
10         <div>
11             <input type="text" onChange={handlerChange}/>
12             <br/>
13             <input type="text" readOnly value={text}/>
14         </div>
15     )
16 }
```

изменения не происходит!

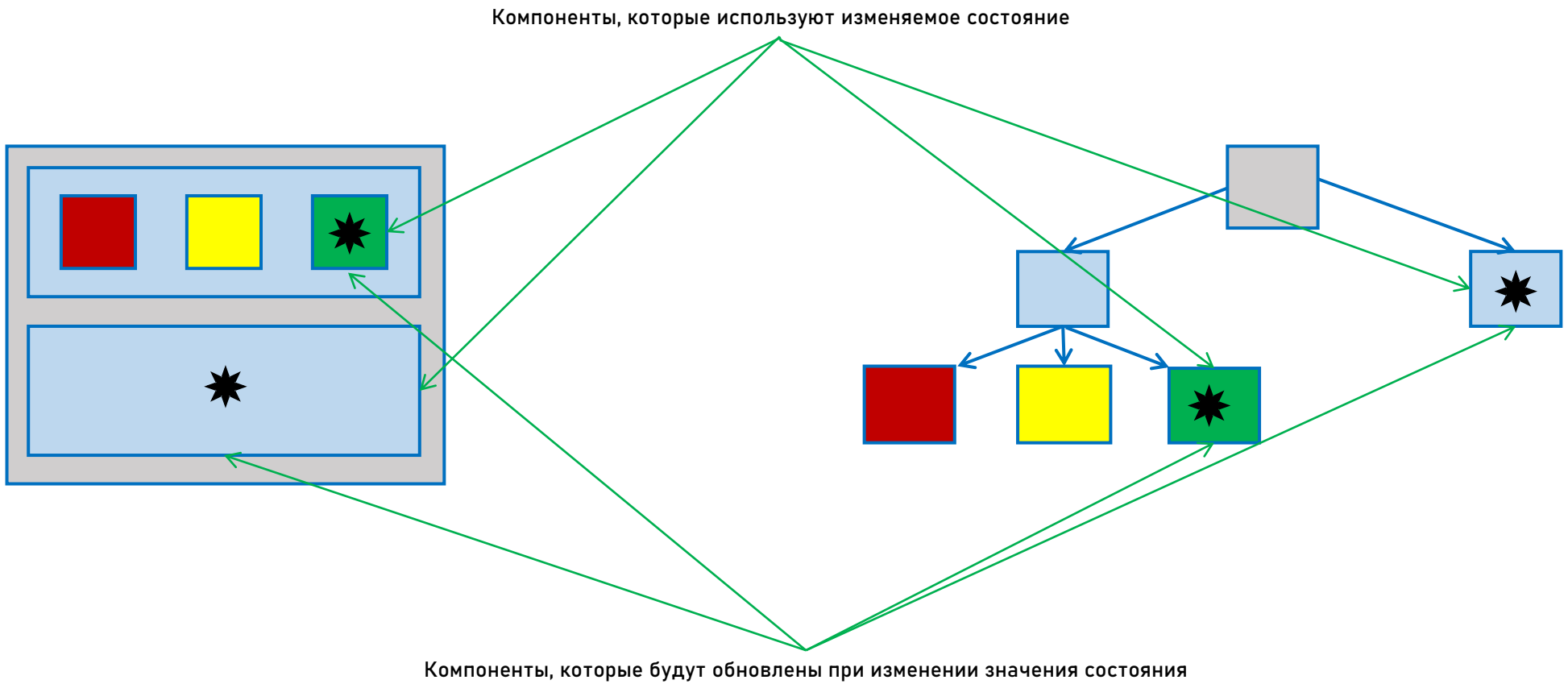


Элемент <input/> не меняет своего значения при изменении переменной text, потому что ReactJS не произволит редендрига всех элементов. Для того, чтобы можно было менять значения в компонентах «на лету» необходимо использовать хуки или состояния ReactJS.

Хуки (состояния)

Понятие состояний (хуков) в ReactJS - это ключевое понятие. Мы можем в компоненте явно указать, какие переменные являются состояними (хуками) и при изменении которых необходимо перерисовать (с измененным значением) элементы, которые использует этот хук.

Рассмотрим как ReactJS перерисовывает компонеты.



Хуки (состояния)

Как реализуются хуки в ReactJS? Состояние представляет собой пару: переменную и функцию для изменения этой переменной. При этом использовать переменную для чтения можно обычным образом, а изменять только с помощью созданной функции.

Для создания состояния необходимо использовать функцию React useState(). Покажем на примере как это работает.

```
1  import { useState } from "react"
2
3  export default function Repeater() {
4      const [text, setText] = useState("Для повтора")
5
6      function handlerChange (e) {
7          console.log(e.target.value)
8          setText(e.target.value)
9      }
10
11     return (
12         <div>
13             <input type="text" onChange={handlerChange}/>
14             <br/>
15             <input type="text" readOnly value={text}/>
16         </div>
17     )
18 }
```

импортируем функцию

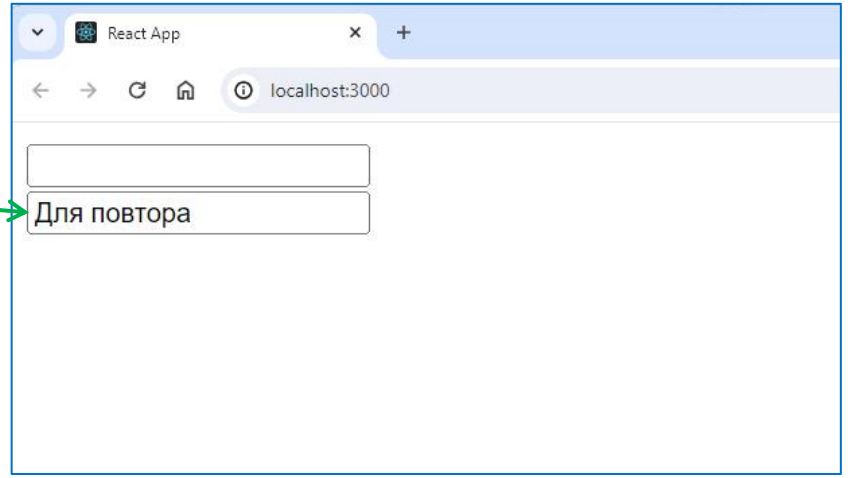
начальное значение

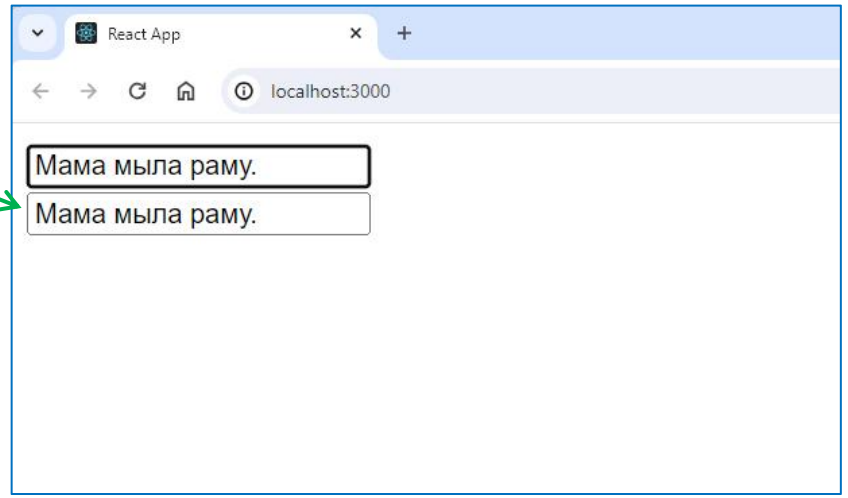
создаем состояние и устанавливаем начальное значение

обновленное значение

изменяем значение состояния

используем значение состояния





Пример

```
1  import { useState } from "react"
2
3  export default function Ligth() {
4      const [ligth, setLigth] = useState('white')
5
6      function Lamp() {
7          const styles = {
8              width: "100px",
9              height: "100px",
10         }
11         styles.background = ligth
12         return (
13             <div style={styles}/>
14         )
15     }
16
17     function handlerClick(e) {
18         switch(e.target.id) {
19             case "1":
20                 setLigth('red')
21                 break
22             case "2":
23                 setLigth('yellow')
24                 break
25             case "3":
26                 setLigth('green')
27                 break
28         }
29     }
30
31     return (
32         <div>
33             <button id="1" onClick={handlerClick}>Красный</button><br/>
34             <button id="2" onClick={handlerClick}>Желтый</button><br/>
35             <button id="3" onClick={handlerClick}>Зеленый</button><br/>
36             <br/>
37             <Lamp/>
38         </div>
39     )
40 }
```

создаем состояние

встроенный компонент

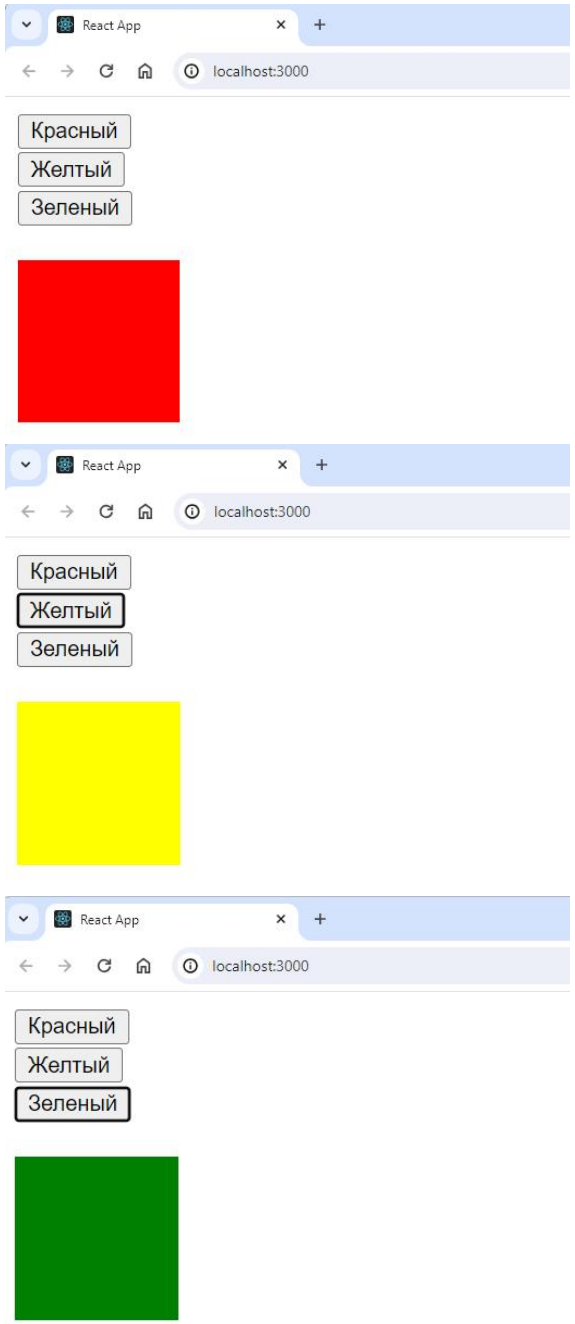
определяем стили

устанавливаем цвет

обработчик события

выводим кнопки с указанием id

выводим встроенный компонент



Обработка списков на ReactJS

Рассмотрим пример, в котором будем обрабатывать списки с помощью компонентов.

```
1  import Lister from "./components/Lister.js";
2
3  export default function App() {
4    const list = [
5      {id: 1, name: "Математический анализ"},
6      {id: 2, name: "Линейная алгебра"},
7      {id: 3, name: "Теория вероятностей"}
8    ]
9    return (
10     <div>
11       <Lister curs={list}/>
12     </div>
13   );
14 }
15
```

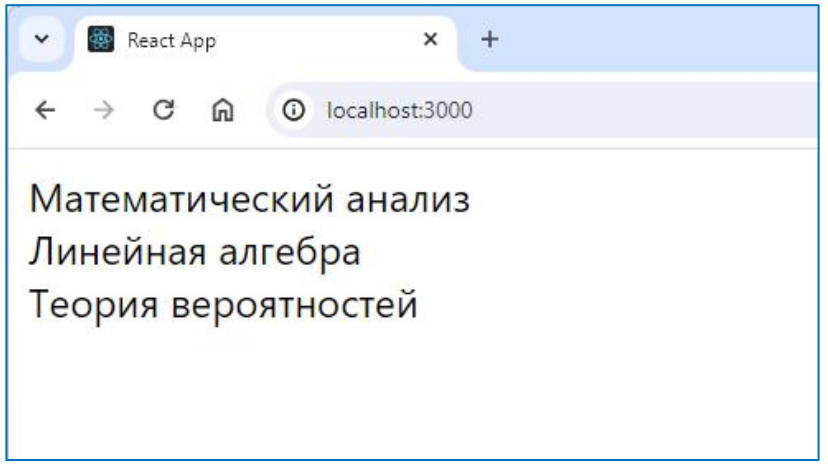
формируем список с уникальными id

вызываем основной компонент

```
1  export default function Lister(props) {
2
3      function Print(props) {
4          return (
5              <div>
6                  {props.item}
7              </div>
8          )
9      }
10
11     let cursitem = props.curs.map((curs) => {
12         return (<Print key={curs.id} item={curs.name} />)
13     })
14
15     return (
16         <div>
17             {cursitem}
18         </div>
19     )
20 }
```

компонент для отрисоки одного курса

обязательно нужно указывать уникальный ключ

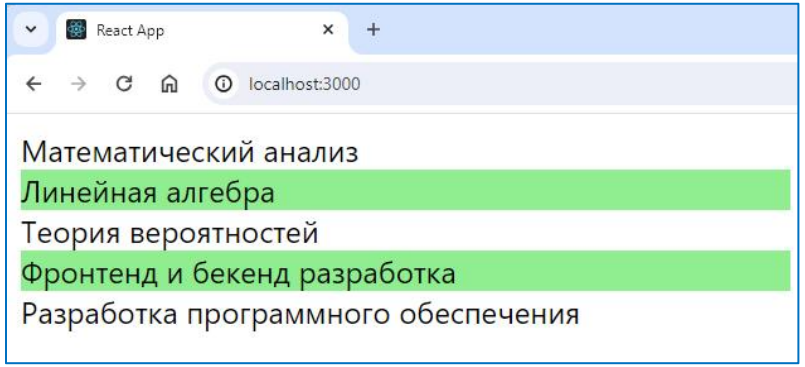


Обработка списков на ReactJS

```
1  export default function Lister(props) {
2
3      function Print(props) {
4          const style = {}
5          if (!props.even) {
6              style.background = "LightGreen"
7          }
8          return (
9              <div style={style}>
10                 {props.item}
11             </div>
12         )
13     }
14
15     let cursitem = props.curs.map((curs, index) => {
16         let even = index % 2 === 0
17         return (<Print key={curs.id} item={curs.name} even={even} />)
18     })
19
20     return (
21         <div>
22             {cursitem}
23         </div>
24     )
25 }
```

для нечетных строк меняем фон

индекс (с 0) списка curs



Обработка списков на ReactJS

```
1  import { useState } from "react"
2
3  export default function Lister(props) {
4      const [checked, setChecked] = useState(0)
5
6      function handlerClick(e) {
7          setChecked(e.target.id)
8      }
9
10     function Print(props) {
11         const style = {}
12         if (checked == props.id) {
13             style.background = "Red"
14         } else if (!props.even) {
15             style.background = "LightGreen"
16         }
17         return (<div id={props.id} style={style} onClick={handlerClick}>{props.item}</div>)
18     }
19
20     let cursitem = props.curs.map((curs, index) => {
21         let even = index % 2 === 0
22         return (<Print key={curs.id} id={curs.id} item={curs.name} even={even} />)
23     })
24
25     return (<div>{cursitem}</div>)
26 }
```

создаем состояние, где будем запоминать отмеченный элемент

передаем id