**Лабораторная работа 7. React CDN. Работа с компонентами, пропсами и хуками**

[Задание 1 Случайная цитата 2](#__RefHeading___1)

[Задание 2 Цветные блоки 2](#__RefHeading___2)

[Задание 3. Работа с компонентами 2](#__RefHeading___3)

[Задание 4. Цветные блоки с кнопкой 6](#__RefHeading___5)

[Задание 5. Калькулятор и тест на одной странице 6](#__RefHeading___6)

[Задание 6. Свой калькулятор 6](#__RefHeading___7)

**Внимание! Все файлы примеров открываем из под Live Server**

**Компонент React** — это переиспользуемый блок кода, который определяет часть пользовательского интерфейса (UI).

Пример:

function Welcome(props) {

return <h1>Привет, {props.name}!</h1>;

}

**props** — это данные, которые передаются в компонент (как аргументы в функцию). Они неизменяемы (read-only).

Пример передачи props:

// Родительский компонент передает `name` в `Welcome`

function App() {

return <Welcome name="Анна" />;

}

// Дочерний компонент принимает props

function Welcome(props) {

return <h1>Привет, {props.name}!</h1>; // Выведет: "Привет, Анна!"

}

Особенности props:

Можно передавать любые данные: строки, числа, массивы, объекты, функции.

Доступны только для чтения (нельзя изменять внутри компонента).

Можно использовать деструктуризацию:

function Welcome({ name }) {

return <h1>Привет, {name}!</h1>;

}

Пример с несколькими props

function UserCard({ name, age, isAdmin }) {

return (

<div>

<h2>{name}</h2>

<p>Возраст: {age}</p>

{isAdmin && <p>Администратор</p>}

</div>

);

}

// Использование

function App() {

return <UserCard name="Иван" age={30} isAdmin={true} />;

}

Результат:

*Copy*

*Иван*

*Возраст: 30*

*Администратор*

Простая аналогия:  
Представьте компонент как функцию, а props — как её аргументы:

function приветствие(имя) { // `имя` — это props

console.log(`Привет, ${имя}!`);

}

приветствие('Мария'); // Передаем props

# Задание 1 Случайная цитата

1. Разобраться с примером ex1.zip
2. Написать код на react cdn, который выводит 5 компонентов Comment, каждый из которых принимает и обрабатывает свойства quote и author (цитату и автора):

*Мы живем не для того, чтобы есть, а едим для того, чтобы жить.*

*Сократ*

*Плохо живут те, которые всю жизнь лишь собираются жить.*

*Публилий Сир*

*Все люди хотят жить долго, но никто не хочет быть старым.*

*Джонатан Свифт*

*Мы не имеем права потреблять счастье, не производя его.*

*Бернард Шоу*

*Счастлив тот, кто считает себя счастливым.*

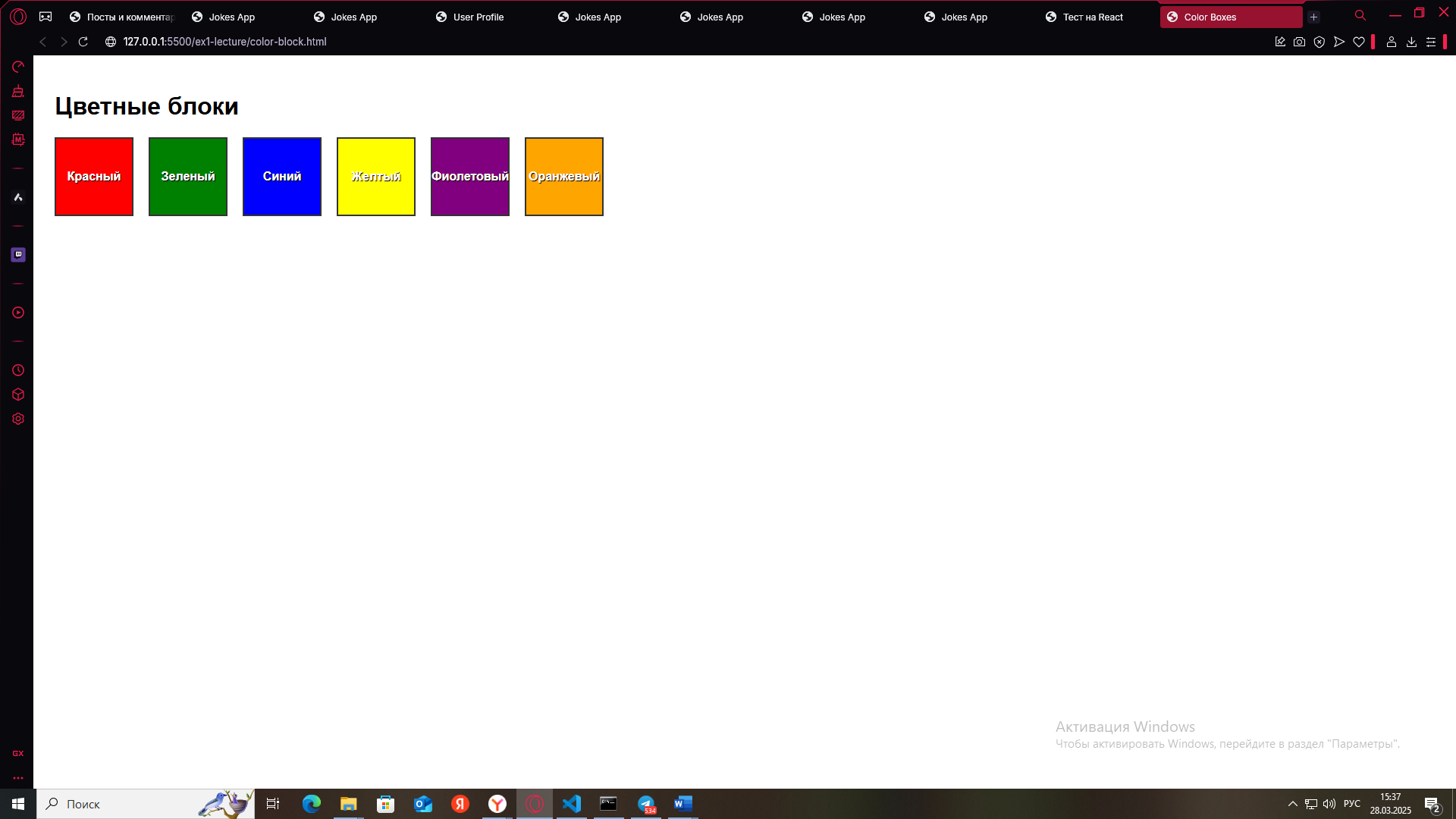
*Генри Филдинг*

1. Изменить код так, чтобы выводилась одна цитата случайным образом. Получить случайный индекс массива можно так:

randomIndex = Math.floor(Math.random() \* ваш массив.length);

# Задание 2 Цветные блоки

Написать приложение на react cdn, которое создает компонент - закрашенный блок и выводит его 6 раз с разной заливкой и названием цвета.



# Задание 3. Работа с компонентами

1. Изучить пример react3.html
2. Удалить MyComponent
3. Создать новый компонент, в который включить topbar из сайта лабораторной работы 3

<div>

   <!-- top-bar -->

   <div class="container-fluid"> <!--класс container-fluid позволяет заполнить все окно браузера -->

    <div class="row top-bar">

      <div class="col-12">

        <a href="#" class="text-white"><span class="mr-2 text-white icon-envelope-open-o"></span> <span class="d-none d-md-inline-block">info@yourdomain.com</span></a>

        <span class="mx-md-2 d-inline-block"></span>

        <a href="#" class="text-white"><span class="mr-2 text-white icon-phone"></span> <span class="d-none d-md-inline-block">+7 (960) 1234 5678</span></a>

                <div class="float-right">

                  <a class="text-white" href="#myModal">Регистрация</a>

        <a href="#" class="text-white"><span class="mr-2 text-white icon-vk"></span> </a>

        <span class="mx-md-2 d-inline-block"></span>

        <a href="#" class="text-white"><span class="mr-2 text-white icon-home"></span> </a>

      </div>

    </div>

</div>

1. Стилизовать с помощью Bootstrap основной компонент

**Хуки**

Хуки позволяют определять и использовать состояние и другие возможности React без создания классов. По сути хуки представляют функции, которые позволяют подключиться к состоянию и другим возможностям, которые есть в React.

React по умолчанию предоставляет ряд встроенных хуков:

* useState: предназначен для управления состоянием компонентов
* useEffect: предназначен для перехвата различного рода изменений в компонентах, которые нельзя обработать внутри компонентов
* useContext: позволяет подписываться на контекст React
* useCallback: позволяет управлять функциями обратного вызова
* useRef: возвращать некоторое изменяемое значение, например, ссылку на html-элементы DOM, которыми затем можно управлять в коде JavaScript и т.д.

**useState - хук состояния**

Это встроенная функция React, которая позволяет добавлять состояние в функциональные компоненты.

const [amount, setAmount] = useState(0);

useState(0):

Инициализирует состояние со значением 0 (стартовое значение)

Возвращает массив из двух элементов: текущее значение и функцию для его обновления

[amount, setAmount]:

Деструктуризация массива

amount - переменная с текущим значением состояния

setAmount - функция для обновления этого значения

***Пример:***

function Counter() {

const [amount, setAmount] = useState(0);

return (

<div>

<p>Текущее значение: {amount}</p>

<button onClick={() => setAmount(amount + 1)}>

Увеличить на 1

</button>

<button onClick={() => setAmount(0)}>

Сбросить

</button>

</div>

);

}

Каждый вызов setAmount вызывает повторный рендер компонента

***Пример с разными типами данных****:*

const [count, setCount] = useState(0); // Число

const [text, setText] = useState(''); // Строка

const [list, setList] = useState([]); // Массив

const [user, setUser] = useState(null); // Объект

***Рабочий пример: Счетчик***

<!DOCTYPE html>

<html lang="ru">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Счетчик на React (CDN)</title>

  <!-- Подключаем  React - основная библиотека React, DOM - для работы с DOM, Babel - позволяет писать современный JSX-синтаксис, который браузеры понимают после преобразования -->

  <script src="https://unpkg.com/react@17/umd/react.development.js"></script>

  <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>

  <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>

  <!-- Стили-->

  <style>

    body {

      font-family: Arial, sans-serif;

      display: flex;

      justify-content: center;

      align-items: center;

      height: 100vh;

      margin: 0;

      background-color: #f5f5f5;

    }

    .counter-container {

      text-align: center;

      background: white;

      padding: 2rem;

      border-radius: 10px;

      box-shadow: 0 4px 8px rgba(0,0,0,0.1);

    }

    .counter-value {

      font-size: 3rem;

      margin: 1rem;

      color: #333;

    }

    button {

      background: #4CAF50;

      color: white;

      border: none;

      padding: 0.5rem 1rem;

      margin: 0.5rem;

      border-radius: 5px;

      cursor: pointer;

      font-size: 1rem;

      transition: background 0.3s;

    }

    button:hover {

      background: #45a049;

    }

    .reset-btn {

      background: #f44336;

    }

    .reset-btn:hover {

      background: #d32f2f;

    }

  </style>

</head>

<body>

  <div id="root"></div>

  <script type="text/babel">

    //компонент Counter

    const { useState } = React;

//useState - это хук React для управления состоянием

//count - текущее значение счетчика (начинается с 0)

//setCount - функция для обновления значения

    function Counter() {

      const [count, setCount] = useState(0);

      const increment = () => setCount(count + 1);

      const decrement = () => setCount(count > 0 ? count - 1 : 0);

      //популярный паттерн в React для защиты от нежелательных значений состояния

      //count > 0 - проверяем, больше ли текущее значение счетчика нуля

      //? - означает "тогда"

      //count - 1 - если условие истинно (count > 0), уменьшаем значение на 1

      //: - означает "иначе"

      //0 - если условие ложно (count ≤ 0), возвращаем 0

      const reset = () => setCount(0);

      //Рендеринг компонента. Отображаем текущее значение {count},

      // отрисовываем три кнопки с обработчиками событий onClick

      return (

        <div className="counter-container">

          <h1>Счетчик</h1>

          <div className="counter-value">{count}</div>

          <div>

            <button onClick={decrement}>-</button>

            <button onClick={increment}>+</button>

            <button className="reset-btn" onClick={reset}>Сбросить</button>

          </div>

        </div>

      );

    }

    ReactDOM.render(<Counter />, document.getElementById('root'));

  </script>

</body>

</html>

# Задание 4. Цветные блоки с кнопкой

Изменить код задания 2 таким образом, чтобы блоки закрашивались случайным образом и по нажатии на кнопку перекрашивались.

Генерация случайного цвета в HEX-формате (например, #ff34a5)

randomColor = '#' + Math.floor(Math.random()\*16777215).toString(16);

# Задание 5. Калькулятор и тест на одной странице

Разобраться с примерами calc-credit.html и test.html. Создать два компонента calc.js и test.js. Создать на react cdn приложение, которое выводит на страницу эти два компонента.

# Задание 6. Свой калькулятор

Изменить код созданного в 5 задании компонента калькулятор так, чтобы выводился ваш калькулятор, созданный в лабораторной работе 5.