1. Основы React

Изучите, что такое React и как он отображает пользовательские интерфейсы. Сосредоточьтесь на компонентах, синтаксисе JSX и отображении элементов на экране.

1. Что такое React и зачем он используется?

При создании веб-сайта обычно требуется, чтобы страница реагировала на действия пользователя, такие как нажатие кнопок, отправка форм, обновление данных или отображение нового контента, без перезагрузки страницы. По мере роста проекта управление этими обновлениями с помощью обычного JavaScript становится все сложнее.

Определение

React — это библиотека JavaScript, разработанная для решения этой проблемы. Она помогает создавать пользовательские интерфейсы, описывая, **как должен выглядеть интерфейс на основе данных** , а затем синхронизирует экран при изменении этих данных.

Вместо того чтобы вручную выбирать элементы и обновлять их, React позволяет мыслить в терминах **компонентов** — небольших, многократно используемых частей интерфейса. Каждый компонент описывает часть экрана, и React эффективно обновляет его при изменении чего-либо.

React фокусируется исключительно на слое пользовательского интерфейса. Он не заменяет JavaScript, HTML или CSS. Он работает поверх них. Вы по-прежнему пишете логику на JavaScript и используете знакомые веб-концепции, но в более чистом и структурированном виде для организации кода пользовательского интерфейса.

Такой подход упрощает понимание, поддержку и масштабирование приложений по мере их роста. Именно поэтому React используется в реальных продуктах, от небольших панелей управления до крупных и сложных веб-приложений.

2. Понимание компонентов React

Приложение React строится из компонентов. Компонент — это многократно используемый элемент пользовательского интерфейса, который управляет тем, что отображается на экране. Вместо того чтобы создавать целую страницу как один большой блок, React рекомендует разбивать интерфейс на более мелкие части. Каждая часть отвечает за определенный раздел пользовательского интерфейса, например, заголовок, кнопку или сообщение.

В React компоненты пишутся с использованием JavaScript и определяются как функции. Каждый компонент возвращает описание того, что должно отображаться на экране, а React позаботится об его отображении.

Компоненты упрощают понимание и поддержку интерфейсов. По мере роста пользовательского интерфейса компоненты помогают организовать код, повторно использовать логику и избегать повторений. Ниже приведён простой пример компонента React:

```
function Welcome() {
  return <h1>Welcome to React</h1>;
}
```

Это компонент React, который называется `<input type="div">` Welcome. Он возвращает элемент пользовательского интерфейса, который React может отобразить на экране. Пока вам не нужно понимать синтаксис. Этот пример демонстрирует, как выглядит компонент на высоком уровне.

3. Основы JSX в React

В React мы используем расширение синтаксиса JavaScript, называемое **JSX (JavaScript XML),** для построения пользовательских интерфейсов. JSX позволяет нам описывать, как должен выглядеть пользовательский интерфейс, используя компоненты React. Для этого React использует синтаксис **JSX** .

JSX похож на HTML, но написан на JavaScript. Он позволяет описывать элементы пользовательского интерфейса непосредственно в коде компонента, вместо того чтобы создавать их шаг за шагом с помощью JavaScript.

Хотя JSX выглядит как HTML, это не HTML. По сути, JSX преобразуется в JavaScript, который понимает React. Это делает JSX одновременно выразительным и мощным, оставаясь при этом частью языка JavaScript.

В JSX можно встраивать выражения JavaScript непосредственно в разметку. Это упрощает отображение динамических данных и управление тем, что появляется на экране. Ниже представлен тот же компонент, что и раньше, но теперь с использованием JSX:

```
function Welcome() {
  const name = "React";
  return <h1>Welcome to {name}</h1>;
}
```

Здесь JSX используется для описания пользовательского интерфейса. Этот {name}синтаксис позволяет вставлять значения JavaScript в интерфейс. React автоматически обновляет отображаемый результат при изменении данных.

4. Рендеринг элементов в React.

Компоненты React описывают, как должен выглядеть пользовательский интерфейс, но их все равно необходимо отобразить на экране.

Определение. Рендеринг — это процесс отображения элемента React в браузере. React связывает ваши компоненты с определенным местом на HTML-странице и поддерживает синхронизацию пользовательского интерфейса при изменении данных.

В типичном React-приложении в HTML-файле есть единственный корневой элемент. React использует этот корневой элемент в качестве точки входа, через которую отображается всё приложение.

Ниже приведён упрощённый пример того, как отображается компонент:

```
import ReactDOM from "react-dom/client";

function App() {
  return <h1>Hello, React</h1>;
}

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

В этом примере React берет Арркомпонент и отображает его внутри HTML-элемента с идентификатором root. Вам пока не нужно запоминать этот код. Он демонстрирует, как React связывает компоненты со страницей.

После отрисовки React автоматически обновляет пользовательский интерфейс при изменении выходных данных компонента. Это позволяет вам **сосредоточиться на описании пользовательского интерфейса,** а не на ручном обновлении страницы.

Вопросы для самопроверки:

1. Какова основная цель React?
2. Что такое компонент React?
3. Что лучше всего описывает JSX?
4. Что означает рендеринг в React?