УДК: 004.432 DOI: 10.34824/VKNIIRAN.2023.14.3.009

СОЗДАНИЕ ОТЗЫВЧИВЫХ ИНТЕРФЕЙСОВ С REACT И CSS FLEXBOX

© Гишлакаев Сайфулла Умарович (a), Гайрабекова Тамара Израиловна (b)

(a) Чеченский государственный университет им. А.А. Кадырова, Российская Федерация, г. Грозный; студент 4 курса Института математики, физики и информационных технологий, sgishlakaev@mail.ru

(b) Чеченский государственный университет им. А.А. Кадырова, Российская Федерация, г. Грозный; кандидат технических наук, доцент кафедры прикладной математики и компьютерных технологий, sti_ing@mail.ru

Аннотация. В современном мире разработки веб-приложений отзывчивость интерфейса стала неотъемлемой частью успешного проекта. Пользователи ожидают, что приложение будет легко использоваться на любом устройстве, будь то компьютер, планшет или мобильный телефон. Для достижения этой цели разработчики активно используют React и CSS Flexbox. В данной статье мы рассмотрим, как создать отзывчивые интерфейсы с помощью этих инструментов.

Ключевые слова: Реакт, вэб-сайт, вэб-приложение, логика, useState, useEffect.

CREATING RESPONSIVE INTERFACES WITH REACT AND CSS FLEXBOX

© Gishlakaev Sayfulla Umarovich (a), Gayrabekova Tamara Israilovna (b)

(a) Chechen State University named after. A.A. Kadyrov, Russian Federation, Grozny; 4th year student at the Institute of Mathematics, Physics and Information Technologies, sgishlakaev@mail.ru

(b) Chechen State University named after. A.A. Kadyrov, Russian Federation, Grozny; Candidate of Technical Sciences, Associate Professor of the Department of Applied Mathematics and Computer Technologies, sti_ing@mail.ru

Abstract. In the modern world of web application development, the responsiveness of the interface has become an integral part of a successful project. Users expect that the application will be easily used on any device, be it a computer, tablet or mobile phone. To achieve this goal, developers actively use React and CSS Flexbox. In this article, we will look at how to create responsive interfaces using these tools.

Key words: React, web site, web application, logic, useState, useEffect.

React является одной из самых популярных JavaScript-библиотек для разработки пользовательских интерфейсов. Она предоставляет удобные инструменты для создания компонентов, которые могут быть переиспользованы и комбинированы для построения сложных интерфейсов. Однако, сам по себе React не обеспечивает отзывчивость интерфейса. Здесь на помощь приходит CSS Flexbox.

CSS Flexbox - это мощный инструмент для создания гибких и адаптивных макетов. Он позволяет легко управлять расположением элементов на странице в зависимости от доступного пространства. Flexbox предоставляет набор свойств и значений, которые позволяют разработчикам создавать адаптивные интерфейсы без необходимости использования сложных сеточных систем или медиа-запросов.

Для начала работы с React и CSS Flexbox необходимо создать новый проект React. Для этого вы можете использовать инструмент Create React App, который позволяет быстро настроить базовую структуру проекта. После создания проекта вам понадобится установить необходимые зависимости, включая React и React-DOM.

Основной компонент вашего интерфейса будет представлять собой корневой элемент приложения. В этом компоненте вы можете определить основной макет и структуру вашего интерфейса с помощью CSS Flexbox. Для этого определите контейнер, который будет содержать все остальные элементы интерфейса. Примените к нему свойство display со значением flex, чтобы указать, что этот контейнер будет использовать Flexbox для управления расположением элементов.

После того, как вы настроили корневой контейнер, вы можете начать добавлять в него дочерние элементы, которые будут составлять ваш интерфейс. Используйте свойства Flexbox, такие как flex-direction, justify-content и align-items, чтобы определить направление, выравнивание и расположение элементов в контейнере. Например, свойство flex-direction со значением row позволит элементам располагаться горизонтально, а свойство justify-content со значением center будет выравнивать элементы по центру контейнера.

Кроме того, CSS Flexbox предоставляет возможность создавать адаптивные макеты, которые автоматически меняются в зависимости от размера экрана. Для этого можно использовать свойство flex-wrap, которое определяет, переносить ли элементы на новую строку при нехватке места. С помощью медиа-запросов вы можете настроить различные значения этого свойства для разных размеров экрана, чтобы ваш интерфейс адаптировался к любым условиям.

Одним из важных аспектов отзывчивого интерфейса является адаптивное изменение размеров элементов. С CSS Flexbox вы можете легко управлять размерами элементов, используя свойство flex-grow. Установка этого свойства для элемента позволяет указать, какую долю доступного пространства он должен занимать. Например, если установить flex-grow равным 1 для всех элементов в контейнере, они будут равномерно распределять доступное пространство между собой.

Пример использования Flexbox в React-компоненте:

```
import React from 'react';
```

```
import './App.css';

const App = () => {
return (
<div className="container">
<div className="item">Item 1</div>
<div className="item">Item 2</div>
<div className="item">Item 3</div>
</div>
);
}

export default App;
```

В этом примере мы создаем простой React-компонент App, в котором используется Flexbox. Классы "container" и "item" определены в CSS-файле и применяются к соответствующим элементам.

Пример использования Flexbox для адаптивного расположения элементов:

```
import React from 'react';
import './App.css';

const App = () => {
return (
<div className="container">
<div className="item">Item 1</div>
<div className="item">Item 2</div>
<div className="item">Item 3</div>
</div>
);
}

export default App;
```

В этом примере мы добавляем медиа-запросы для адаптивного изменения макета на разных размерах экрана. С помощью свойства flex-direction и flex-wrap мы изменяем направление и перенос элементов.

Важно отметить, что помимо CSS Flexbox существует и другой подход к созданию отзывчивых интерфейсов с использованием CSS Grid. CSS Grid предоставляет более мощные возможности для создания сеток и сложных макетов. Однако, CSS Flexbox обладает более простым синтаксисом и широкой поддержкой среди браузеров, что делает его предпочтительным выбором для большинства случаев.

React и CSS Flexbox взаимодействуют отлично вместе, позволяя разработчикам создавать отзывчивые интерфейсы с минимальными усилиями. React обеспечивает деклара-

тивную модель разработки, позволяющую легко обновлять интерфейс при изменении состояния приложения, в то время как CSS Flexbox обеспечивает гибкость и адаптивность макета.

В заключение, создание отзывчивых интерфейсов с React и CSS Flexbox является важным аспектом современной веб-разработки. Использование этих инструментов позволяет создавать интерфейсы, которые будут прекрасно работать на различных устройствах и размерах экранов. React предоставляет мощные возможности для создания компонентов и управления состоянием приложения, в то время как CSS Flexbox позволяет гибко управлять расположением и размерами элементов.

При разработке отзывчивых интерфейсов с React и CSS Flexbox важно учитывать потребности и ожидания пользователей. Используйте медиа-запросы для настройки макета и размеров элементов под разные размеры экранов. Также уделите внимание доступности вашего интерфейса, обеспечивая удобное взаимодействие с помощью клавиатуры и экранного устройства для людей с ограниченными возможностями.

В конечном итоге, создание отзывчивых интерфейсов требует практики и экспериментов.

ЛИТЕРАТУРА

1. Николас З. ECMAScript 6 для разработчиков. 2017 г.
2. Прасти Н. Введение в ECMAScript 6. 2016г.
3. Марейн Х. Выразительный JavaScript. Современное веб-программирование. 3-е издание. 2020 г.
4. Рыбаков Е. JavaScript и Node.js для Web-разработчиков. 2022 г.
5. Черный Б. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений. 2021 г.

REFERENCES

1. Nicholas Z. ECMAScript 6 for developers. 2017
2. Prasti N. Introduction to ECMAScript 6. 2016.
3. Marein H. Expressive JavaScript. Modern web programming. 3rd edition. 2020
4. Rybakov E. JavaScript and Node.js for Web developers. 2022
5. Cherny B. Professional TypeScript. Development of scalable JavaScript applications. 2021