SECURITY RISK ASSESSMENT REPORT

Code Review – News

https://sit-chameleon-website-0bc2323.ts.r.appspot.com/blog

VERSION 0.0.1

28/11/2023

Miriam Azmy

TABLE OF CONTENTS

1. Code to be Reviewed:

```
import React, { Component } from "react";
import SwiperNews from "./Swiper";
import articleList from "./articles";
import "./styles.css";
import SearchButton from "./images/SearchButton.png"
class News extends Component {
  constructor(props) {
    super(props);

    // Initialize the state
  }

  render() {
    return (
      <main className="bg-[#4fa373] py-10 ">
        {/* Main container with a green background and padding */}
        <h1 className="text-[27px] font-bold text-center text-black">
          Latest News
        </h1>
        {/* Header with bold, centered, and black text */}
        <div className="bg-[#deece3] w-full py-6 mt-4 flex flex-col gap-y-2 min-[400px]:flex-row flex-shrink justify-center items-center gap-3 px-[10px]">
          {/* Container with light blue background, flex layout, and padding */}
          <div className="flex relative">
            {/* Flex container with relative positioning */}
            <img
              src={SearchButton}
              className="h-[25px] mt-2 absolute ml-2 object-center"
              alt="searchBar"
            />
            {/* Search icon with absolute positioning */}
            <input
              type="text"
              className="py-2 placeholder:text-black shrink outline-none pl-10  w-40 md:w-52"
              placeholder="Search News"
            />
            {/* Input field with padding, placeholder styling, and width based on screen size */}
            <button className="bg-sky-600 py-2 px-3 text-white border">
              Search
            </button>
            {/* Search button with sky blue background, white text, and border */}
          </div>
          <select className="flex relative justify-center items-center gap-2 bg-white text-[18px]  px-4 py-2">
            {/* Dropdown selection with white background and styling */}
            <option value="" className="">
              Year
              {/* Default option with text "Year" */}
              {/* Arrow icon (commented out as it's not present in the code) */}
            </option>
            <option value="2024">2024</option>
            <option value="2023">2023</option>
            {/* Options for the year dropdown */}
          </select>
        </div>
        {/* End of the search and dropdown container */}
         <div className=" grid-cols-1 md:grid-cols-2 lg:grid-cols-3 items-start justify-center mt-10 gap-3 gap-y-5
overflow-hidden md:grid hidden px-[20px]">
          {/* Container for grid layout with specified column sizes, spacing, and hidden on medium screens */}
          {articleList.map((item) => (
            <div
              className="flex flex-col justify-center items-center "
              key={item.key}
            >
              {/* Individual news item container */}
              <h2 className="text-[24px] font-bold text-center text-black uppercase">
                {item.title}
              </h2>
              {/* News title with specific styling */}
              <img
                src={item.image}
                className="mt-2 w-[360px] h-[180px] border-white border-y-[16px] border-x-[20px]"
              />
              {/* Image with specific dimensions and border styling */}
              <p className="text-black font-medium text-[15px] max-w-[360px] pt-2 text-left hidetext">
                {item.preview}
              </p>
              {/* News preview text with specific styling */}
              <h3 className="text-[14px] text-black font-bold text-left max-w-[360px] w-full">
                {item.date}
              </h3>
              {/* News date with specific styling */}
            </div>
          ))}
        </div>
        {/* End of the news grid layout container */}
        <div className="px-[10px] md:hidden block">
          <SwiperNews />
        </div>
        {/* Container for mobile view with a SwiperNews component */}
      </main>
    );
  }
}

export const newsSearchableContents = articleList.reduce((acc, article) => {
  return [...acc, article.title, article.preview, article.author];
}, []);

export default News;
```

2. Purpose of the Code

The provided code snippet serves the purpose of constructing a responsive navigation bar within a web The provided code serves the purpose of rendering a dynamic and responsive news section within a web application. This section comprises several elements, including a headline, a search bar with a search button, a dropdown for selecting the year, and a grid layout displaying news articles. Additionally, the code incorporates a Swiper component for mobile view, enhancing the user experience on smaller screens.

The <h1> element establishes the headline "Latest News," signaling the user that the content pertains to recent information. The subsequent sections contribute to creating an interactive and visually appealing news browsing experience.

3. Code Structure and Organization

The code maintains a structured and organized format, leveraging React components for modularity. The main container encapsulates different sections, such as the headline, search bar, dropdown, and the grid layout of news articles. The use of Flexbox and CSS Grid ensures a responsive and well-organized layout.

The code follows a mobile-first approach, with the grid layout hidden on medium screens and replaced by a Swiper component for better usability on smaller devices. This approach aligns with modern web development practices, prioritizing a seamless experience on mobile devices.

4. Use of Data Structures

The code employs simple data structures, such as arrays and objects. The articleList array holds objects, each representing a news article with properties like title, image, preview, and date. This structured data is then mapped to create individual news items in the grid layout. The newsSearchableContents array is dynamically generated from the articleList, containing searchable content for future search functionality.

5. Commenting and Documentation

While the code is well-structured in general, adding comments to explain the purpose of certain parts or difficult logic would improve its maintainability. Adding comments to the source to clarify the role of the SwiperNews component, the rationale behind producing newsSearchableContents, or any other sophisticated functionality helps with the ongoing efficiency and growth of the code.

6. Coding Standards Adherence

The code shows a noteworthy dedication to keeping a consistent and coherent style throughout its structure while adhering to known styling rules. Consistent styling fosters a shared understanding of the visual language of the codebase, making it easier for developers to read and contribute to various portions of the programme. It reduces the likelihood of misunderstandings or errors caused by coding style differences by ensuring that all team members adhere to the same set of norms. Furthermore, it simplifies future maintenance and changes because developers can easily understand the purpose and operation of individual pieces or sections without having to deal with diverse styles. As a result, maintaining adherence to styling guidelines across the codebase is critical for maintaining an efficient, collaborative, and scalable development environment.

7. Code Complexity

The code exhibits a moderate level of complexity due to the inclusion of dynamic elements such as the grid layout and the Swiper component. The conditional rendering based on screen size contributes to a cleaner user interface, but the code could benefit from additional comments to clarify any intricate logic.

8. Potential Problem Areas

The code appears to be well-constructed without any immediate problem areas. However, as the application evolves, ensure that the integration of new features or modifications doesn't introduce issues, and regularly conduct testing, especially when updating external dependencies.

9. Security and Performance Checks

From a security perspective, the provided code for the News component demonstrates a focus on frontend presentation and user interface, with an emphasis on styling and layout. While it adheres to general frontend security practices, there are areas to consider for further improvement.

Image Source Security: Ensure that the image sources used in the `img` elements, such as `{item.image}`, are validated and sanitized to prevent potential security risks like injection attacks. If these sources are dynamically generated or fetched from user inputs or external APIs, robust validation is crucial to prevent issues like Cross-Site Scripting (XSS) attacks.

Input Validation: The code includes an input field for searching news. It's essential to implement proper validation and sanitization for user inputs to prevent security vulnerabilities. Consider incorporating input validation functions or libraries to sanitize user-provided data and avoid potential injection attacks.

Performance Considerations: While the code focuses on frontend presentation, it's important to ensure optimal performance. Consider lazy loading images to improve initial page load times, especially if the news articles contain large images. Additionally, implement responsive image loading techniques to deliver appropriately sized images based on the user's device, contributing to a smoother user experience.

Server Interaction: If the News component interacts with a backend server, ensure that data exchanged between the frontend and backend is handled securely. Implement secure communication protocols (HTTPS) and validate inputs on the server-side to prevent security vulnerabilities.

Accessibility: Consider enhancing the accessibility of the code by ensuring that all interactive elements, such as buttons and input fields, are appropriately labeled. This contributes to a more inclusive user experience and aligns with best practices for web development.

10. Proposed changes to Code

```
// Import necessary modules and components
import React, { Component } from "react";
import SwiperNews from "./Swiper";
import articleList from "./articles";
import "./styles.css";
import SearchButton from "./images/SearchButton.png";

// Define the News component
class News extends Component {
 constructor(props) {
  super(props);
  // Initialize the state if needed
 }

 render() {
  return (
   <main className="bg-green-500 py-10">
    {/* Improved headline with consistent styling */}
    <h1 className="text-3xl font-bold text-center text-black">
     Latest News
    </h1>

    {/* Enhanced search and dropdown container */}
     <div className="bg-gray-200 w-full py-6 mt-4 flex flex-col md:flex-row flex-shrink justify-center items-center gap-3 px-4">
     {/* Search container with improved styling */}
     <div className="flex relative">
      <img
       src={SearchButton}
       className="h-6 mt-2 absolute ml-2 object-center"
       alt="searchBar"
      />
      <input
       type="text"
       className="py-2 placeholder:text-black outline-none pl-10 w-40 md:w-52"
       placeholder="Search News"
      />
      <button className="bg-blue-500 py-2 px-3 text-white border">
       Search
      </button>
     </div>

     {/* Improved dropdown selection */}
     <select className="flex justify-center items-center gap-2 bg-white text-lg px-4 py-2">
      <option value="" className="">
       Year
      </option>
      <option value="2024">2024</option>
      <option value="2023">2023</option>
     </select>
    </div>
    {/* End of the search and dropdown container */}

    {/* Grid layout with consistent styling and readability */}
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 items-start justify-center mt-10 gap-3 gap-y-
5 overflow-hidden md:grid hidden px-4">
     {articleList.map((item) => (
      <div className="flex flex-col justify-center items-center" key={item.key}>
       <h2 className="text-xl font-bold text-center text-black uppercase">
        {item.title}
       </h2>
       <img
        src={item.image}
        className="mt-2 w-full h-[180px] border-white border-y-[16px] border-x-[20px]"
        alt={item.title}
       />
       <p className="text-black font-medium text-lg max-w-full pt-2 text-left hidetext">
        {item.preview}
       </p>
       <h3 className="text-base text-black font-bold text-left max-w-full w-full">
        {item.date}
       </h3>
      </div>
     ))}
    </div>
    {/* End of the news grid layout container */}

    {/* Mobile view with SwiperNews component */}
    <div className="px-4 md:hidden block">
     <SwiperNews />
    </div>
   </main>
  );
 }
}

// Extract searchable content for future use
export const newsSearchableContents = articleList.reduce((acc, article) => {
 return [...acc, article.title, article.preview, article.author];
}, []);

export default News;
```

11. Findings

The code creates a visually appealing and responsive news section that displays items in a grid structure. It has a clear structure that maintains readability and organisation while paying close attention to detail in styling and presentation. The code, on the other hand, lacks specific comments, which are critical for encouraging cooperation and assisting developers unfamiliar with the codebase. Furthermore, security considerations centre on image source validation and input sanitization, both of which are critical for mitigating potential injection attacks. To optimise page load times, suggestions for performance enhancements include lazy loading images and responsive image loading. Overall, the component adheres to frontend best practices; nevertheless, including extensive comments and addressing security and performance concerns would improve its quality, maintainability, and user experience.