ReeliePro Code Detailed Documentation

1. JavaScript Detailed Functionality

The JavaScript code provided is responsible for enabling a toggle menu functionality. Here's a detailed explanation:

1. **DOMContentLoaded Event**: Ensures the script runs only after the HTML is fully loaded. This prevents errors related to elements not yet being present in the DOM.

2. **Event Listeners**:
   - **toggleButton**: Toggles the "hidden" class on both the mobile menu (`menu`) and the header logo (`logo`) to show or hide them.
   - **closeButton**: Specifically hides the mobile menu and ensures the logo reappears.

3. **Key Points**:
   - The `classList.toggle()` method dynamically adds or removes the "hidden" class, enabling smooth transitions between visible and hidden states.
   - The code improves user experience by ensuring menu visibility is easily controllable on smaller devices.

```
document.addEventListener('DOMContentLoaded', () => {
    const toggleButton = document.getElementById('menu-toggle');
    const menu = document.getElementById('mobile-menu');
    const logo = document.getElementById('logo');
    const closeButton = document.getElementById('menu-close');

    toggleButton.addEventListener('click', () => {
        menu.classList.toggle('hidden');
        logo.classList.toggle('hidden');
    });

    closeButton.addEventListener('click', () => {
```

```
        menu.classList.add('hidden');
        logo.classList.remove('hidden');
    });
});
```

2. Extensive TailwindCSS Usage

TailwindCSS plays a significant role in creating a responsive and visually appealing layout for the website.

Key TailwindCSS Classes:

1. **bg-[#f9fafb]**: Defines the light gray background color used throughout the header and sections.

2. **flex Utilities**: Ensures proper alignment and spacing:

   - **justify-center**: Centers elements horizontally.

   - **items-center**: Aligns elements vertically.

3. **hidden, lg:flex**: Implements responsive design by showing or hiding elements based on screen size (e.g., hiding the desktop menu on smaller devices).

4. **hover:font-bold**: Adds hover effects for interactive links.

5. **z-50**: Ensures the header stays above other elements on the page.

Hero Section:

The hero section uses:

- **relative** for proper layering of child elements.

- **text-center** for centering content.

- **space-x-4** for evenly spacing child elements.

These utilities simplify the creation of clean and consistent designs without additional CSS.

```
<header class="bg-[#f9fafb] fixed z-50 w-full h-24">
    <div class="mx-auto flex flex-row justify-between items-center px-4 py-4">
        <h1 class="text-2xl font-bold text-orange-500">Reelie<span class="text-[#758173]">Pro</span></h1>
        <nav class="hidden lg:flex space-x-4">
            <a href="#" class="text-[#758173] hover:font-bold">Home</a>
            <a href="#" class="text-[#758173] hover:font-bold">Upcoming Tournaments</a>
        </nav>
    </div>
</header>
```

3. Extended Custom CSS Explanation

Custom CSS enhances TailwindCSS by adding unique styles, such as wave backgrounds and fixed header functionality.

Highlights:

1. **--header-height**:
   - Defined as a CSS variable for consistent spacing between the header and sections.
   - Easily adjustable for future design changes.

2. **.custom-waves**:
   - Creates decorative wave patterns using background images.
   - Leverages pseudo-elements (::after) to add layered effects.

3. **position: fixed**:
   - Keeps the header at the top of the viewport during scrolling.
   - Enhances usability by ensuring navigation is always accessible.

These custom styles complement TailwindCSS utilities and add a unique touch to the layout.

```
.custom-waves {
    position: absolute;
    width: 100%;
    height: 750px;
    background: url('./img/Vector (2).png') no-repeat bottom center / 100% 70%;
    overflow: hidden;
    z-index: 2;
}

.custom-waves::after {
    content: '';
    position: absolute;
    bottom: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: url('./img/Vector (2).png') no-repeat bottom center / 100% 65%;
    z-index: 1;
}
```

4. Additional Notes and Observations

The provided code is well-structured, leveraging modern web technologies effectively. The combination of TailwindCSS and custom CSS ensures a responsive and aesthetically pleasing design. JavaScript enhances interactivity, particularly for mobile users.

Recommendations for Further Improvement:

1. **Accessibility**: Add ARIA labels to improve navigation for screen readers.

2. **Performance**: Minimize and optimize images to reduce load time.

3. **Scalability**: Use TailwindCSS's configuration file to manage custom colors and styles more efficiently.