Sendout Frontend Coding Challenge 2025

Disclaimer

Limit your invested time to a maximum of 1.5 hours. You don't need to be finished when you reach this time limit since it's rather short.

Please also provide us with a description of how you tackled this task and what challenges you faced.

We want to evaluate how you work and what your process is, so don't worry if you cannot provide a perfect solution.

Part 1: Problem-solving

Problem: Simulate Infinite Scroll Indexing

In a banner carousel, we want to simulate **infinite scrolling** by cloning the **first** and **last** banners and inserting them at the beginning and end of the banner list.

For example, if the original list is:

`["A", "B", "C"]`

And the augmented list is:

`["C", "A", "B", "C", "A"]`

The carousel **starts at index 1**, which is the first real item (`"A"`). When scrolling right (forward), the internal index increases. In the real front-end banner component, the scroll to the cloned banner A is animated, and followed by an instant jump to the real A, creating an illusion of infinite scrolling.

Rules:

- After the scrolling right actions, calculate the **internal index** in the augmented list.
- If the result is on a **cloned item**, a **jump is required**:
  - If the index is `0` → jump to `n` (last real item, won’t happen since we only allow right scroll)
  - If the index is `n + 1` → jump to `1` (first real item)
- Write code for the function and pass all the tests

```
type ScrollResult = {
  internalIndex: number;
  needsJump: boolean;
  jumpToIndex?: number;
};

/**
 * @param banners - an array of strings representing the real banners (length ≥ 2)
 * @param scroll - an integer representing how many times the user has scrolled right (0 <= scroll <= 100)
 * @returns the internal index in the augmented list, and jump behavior if on a cloned item
 */
function simulateScroll(banners: string[], scroll: number): ScrollResult {
  // Your code here
}
```

example runs:

```
simulateScroll(["A", "B", "C"], 0);
// { internalIndex: 1, needsJump: false }

simulateScroll(["A", "B", "C"], 3);
// { internalIndex: 4, needsJump: true, jumpToIndex: 1 }

simulateScroll(["A", "B", "C"], 4);
// { internalIndex: 2, needsJump: false }
```

Part 2: UI Implementation from Figma Design

Please follow the task provided in the README of the following GitHub project:

GitHub - DevOpsESolutionTesto/frontend-coding-challenge

The Figma Mockup is password-protected:

Password is `besure`