React Carousel Logic Explanation

This code creates a responsive carousel that displays projects differently based on screen size. Here's a detailed breakdown:

State Management

`const [currentCount, setCurrentCount] = useState(0)`

This tracks which project is currently being viewed. It starts at index 0 (the first project). When you navigate through projects, this value changes to show different ones.

`const [cardShow, setCardShow] = useState(1)`

This controls how many project cards are displayed at once. It starts at 1 but will be updated based on screen width. On small screens (phones/tablets), you see 1 card. On large screens (desktop, 1024px+), you see all projects at once.

The useEffect Hook

This effect runs once when the component mounts (note the empty dependency array `[]`). It handles responsive behavior:

```javascript
useEffect(()=> {
  const updatecardShow = () => {
    if (window.innerWidth >= 1024){
      setCardShow(projectsData.length)  // Show all projects on desktop
    } else {
      setCardShow(1)  // Show 1 project on mobile/tablet
    };
  };
  updatecardShow();  // Run it immediately when component loads

  window.addEventListener('resize', updatecardShow);  // Listen for window resizing
  return ()=> window.removeEventListener('resize', updatecardShow);  // Cleanup
},[])
```

What it does:

- Calls `updatecardShow()` immediately to set the correct display count on page load
- Adds a resize listener so that when the user resizes their browser window, the number of visible cards updates

- Returns a cleanup function that removes the event listener when the component unmounts (prevents memory leaks)

Navigation Functions

`const nextProjects = () => { ... }`

Moves forward through the projects list:

```javascript
setCurrentCount((prevIndex) => (prevIndex + 1) % projectsData.length)
```

How it works:

- `prevIndex` is the current index
- `(prevIndex + 1)` moves to the next project
- `% projectsData.length` wraps around—when you reach the last project and click next, it loops back to the first project (0)

Example: If there are 5 projects, indices go 0→1→2→3→4→0→1...

`const prevProjects = () => { ... }`

Moves backward through the projects list:

```javascript
setCurrentCount((prevIndex) => prevIndex === 0 ? projectsData.length - 1 : prevIndex - 1)
```

How it works:

- `prevIndex === 0` checks if you're at the first project
- If yes, it jumps to the last project (`projectsData.length - 1`)
- If no, it decrements by 1 to show the previous project
- This creates a loop in the reverse direction

Example: If there are 5 projects, indices go 4→3→2→1→0→4→3...

Complete Behavior

On Desktop (1024px+): All projects display at once, so `currentCount` isn't really used for hiding/showing—it might control highlighting or something similar.

On Mobile/Tablet (<1024px): One project displays at a time. Users click next/prev buttons to cycle through them. The carousel loops infinitely in both directions.

This pattern is commonly used for image galleries, product carousels, and testimonial sliders.