Sure, I'll explain the code step-by-step:

1. The code is wrapped inside an Immediately Invoked Function Expression (IIFE), denoted by the `(function () { ... })();` syntax. This is used to create a private scope for the variables and functions inside it. It ensures that the variables defined inside the IIFE do not interfere with the global scope.

2. The code assumes that there are HTML elements with the class 'counterBtn' for buttons and an element with the ID 'counter' for displaying the count.

3. `const buttons = document.querySelectorAll('.counterBtn');`: This line selects all elements with the class 'counterBtn' and stores them in the `buttons` variable. The `document.querySelectorAll` method returns a NodeList containing all the elements that match the given CSS selector.

4. `let count = 0;`: This line initializes a variable `count` with a value of 0. This variable will keep track of the current count.

5. `buttons.forEach(function (button) { ... });`: This line iterates through each button in the `buttons` NodeList using the `forEach` method. For each button, it attaches a click event listener.

6. `button.addEventListener('click', function () { ... });`: This line adds a click event listener to each button. When a button is clicked, the provided function is executed.

7. Inside the click event listener function, it checks which button was clicked using `button.classList.contains('prevBtn')` and `button.classList.contains('nextBtn')`. If the button contains the class 'prevBtn', the count is decremented by one. If the button contains the class 'nextBtn', the count is incremented by one.

8. `const counter = document.querySelector('#counter');`: This line selects the element with the ID 'counter' and stores it in the `counter` variable.

9. `counter.textContent = count;`: This line updates the text content of the element with the current value of the `count` variable. So, when a button is clicked, the count value is displayed in the 'counter' element.

10. The code then checks the value of the `count` variable to determine the color of the text inside the 'counter' element. If the count is less than 0, the text color is set to red. If the count is greater than 0, the text color is set to green. If the count is 0, the text color is set to a default color (#333333, a dark gray).

In summary, this code sets up a simple counter functionality using JavaScript. When you click on buttons with the class 'prevBtn', the counter decrements, and when you click on buttons with the class 'nextBtn', the counter increments. The current count value is displayed in an element with the ID 'counter', and the text color of the element changes based on whether the count is positive, negative, or zero.