cursor: pointer;

is used to change the mouse cursor appearance when it hovers over an element to indicate that it is clickable or interactive.

border-radius: 5px;

means that the corners of the element will have a circular arc with a radius of **5px**

margin: 10px auto;

is a shorthand property for setting the top and bottom margins to **10px** and the left and right margins to **auto**

**rgb(192, 217, 255)** represents a color in the RGB (Red, Green, Blue) color model.

"Sans-serif" is a generic font family in CSS.

position: relative;

This property specifies the positioning scheme of the element. Setting it to relative means the element is positioned relative to its normal position in the document flow. Other positioned elements will still respect the space occupied by this element.

display: flex;

Flexbox is a one-dimensional layout model that provides a more efficient way to align and distribute space among items in a container.

justify-content: space-between;

This property aligns the flex items along the main axis (in this case, vertically) and distributes the remaining space evenly between them, with equal space placed before the first item and after the last item.

The CSS property **box-shadow** is used to add shadow effects around an element's frame

The CSS property **opacity** is used to specify the transparency level of an element.

Sure, let's break down the JavaScript code line by line:

1. `document.addEventListener('DOMContentLoaded', function() {`

- This line adds an event listener to the `DOMContentLoaded` event of the `document`. The `DOMContentLoaded` event is fired when the initial HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading.

- When this event occurs, the provided function is executed.

2. `const lights = document.querySelectorAll('.light');`

- This line selects all elements with the class name `light` using the `querySelectorAll()` method and stores them in a constant called `lights`.

- The `querySelectorAll()` method returns a static (non-live) `NodeList` of all matching elements in the document.

3. `let intervalId;`

- This line declares a variable called `intervalId` without assigning it a value. This variable will be used to store the ID returned by `setInterval()` so that it can later be cleared by `clearInterval()`.

4. `function turnOnLights() {`

- This line declares a function named `turnOnLights()`.

5. `let currentLightIndex = 0;`

- This line declares a variable called `currentLightIndex` and initializes it with a value of `0`.

- This variable will be used to keep track of the index of the currently lit traffic light.

6. `intervalId = setInterval(() => {`

- This line sets up a recurring interval using the `setInterval()` function. It takes two parameters: a function and a time interval in milliseconds.

- Here, an arrow function is used as the first parameter. This function will be executed repeatedly at the specified interval.

7. `lights.forEach(light => light.style.opacity = '0');`

- This line uses the `forEach()` method to iterate over each element in the `lights` NodeList.

- For each element, it sets the `opacity` CSS property to `'0'`, effectively turning off all the lights.

8. `lights[currentLightIndex].style.opacity = '1';`

- This line sets the `opacity` CSS property of the currently lit traffic light (determined by `currentLightIndex`) to `'1'`, effectively turning on the next light in the sequence.

9. `currentLightIndex = (currentLightIndex + 1) % lights.length;`

- This line increments `currentLightIndex` by `1` and ensures that it wraps around to `0` when it reaches the end of the `lights` NodeList.

- This allows the traffic light sequence to loop continuously.

10. `}, 3000);`

- This line specifies the interval duration in milliseconds. In this case, the function provided to `setInterval()` will be executed every `3000` milliseconds (or every 3 seconds).

11. `function turnOffLights() {`

- This line declares a function named `turnOffLights()`.

12. `clearInterval(intervalId);`

- This line clears the interval previously set by `setInterval()` using the `clearInterval()` function. It takes the `intervalId` as a parameter.

13. `lights.forEach(light => light.style.opacity = '0');`

- This line turns off all the lights by setting their `opacity` CSS property to `'0'`.

14. `document.getElementById('onButton').addEventListener('click', turnOnLights);`

- This line adds an event listener to the button with the id `onButton`. When this button is clicked, the `turnOnLights()` function will be called.

15. `document.getElementById('offButton').addEventListener('click', turnOffLights);`

- This line adds an event listener to the button with the id `offButton`. When this button is clicked, the `turnOffLights()` function will be called.

Overall, this JavaScript code sets up functionality for a basic traffic light simulation, where clicking the "Turn On" button starts a sequence of lights turning on and off in a loop, and clicking the "Turn Off" button stops the sequence and turns off all the lights.