1. Extract a set of requirements from the code:

As the application URL is loaded below is expected:

1. The box would be in the top left corner by default until the user takes mouse click action (first mouse click).
2. As soon as the box is clicked for the first time, new random coordinates will be generated for this box and the box will then move to this new location.
3. The constraint for the box coordinates (boundaries) after first click are as below:
    a. Max width + default box size (100 in our case) should be less than window inner width
    b. Max height + default box size (100 in our case) should be less than window inner height
4. User idle state: If the user clicks the box and does not take any action for a specified timeout (10 seconds in our case) then the box will move to default coordinates which are implemented as below:
    a. X coordinate: window.innerWidth - boxSize) / 2
    b. Y coordinate: window.innerHeight - boxSize) / 2

2. Integration tests to confirm those requirements:

1. Test the application under various testing environments such as QA, beta
2. Cross platform and cross browser on different screen sizes and window sizes
3. Application uses a few configuration values such as box size, idle timeout. If it is expected that an api will fetch these details from DB, the integration with this API without mocking lies in scope of testing.
4. If the app will be hosted somewhere and the user will access it through a redirection from other page then:
    a. Redirection from page or other app
    b. One flow where user can access the app (successful fetch of code from source to browser)
    c. Sanity for different network speeds
5. Flow sanity with various types of mouse such as wireless, wired etc. because mouse click is the most important part of this app.
6. Verify the timeout value on different OS to make sure the app reads system time as expected.

3. **What would be the best test for this — integration or unit tests?**

The answer depends on some conditions. It is subjective to the future setup of this app.

Answer:
Unit tests for now: because the code I see has no such integration with any api or DB yet. As long as it has no such dependencies, the app can be tested standalone with automated unit tests. The requirements I extracted can be tested with unit tests added in the pipelines.

The scenario where this app will have backend dependency, then **integration** and E2E cases need to be tested separately. Also if the app is going to be hosted on a remote instance, then for once after first deployment we should test redirections, flow testing to check latency and user experience.

Note: Compatibility with browsers, screen size, devices is separate and needs to be done dedicatedly. Unit tests may not cover this aspect. I excluded this side from my answer.