CAA 2

Web Programming

Alejandro Pérez Bueno

Dec 21, 2023

Table of Contents

Question 1

Given this code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document Title</title>
    <script src="mymodule.js" type="module"></script>
  </head>
  <body>
  </body>
</html>
```

The correct answer is `b` (The `script` tag should have a `defer` attribute to improve the loading process). While an `async` attribute can load the script asynchronously, it does not guarantee that it will be loaded after the HTML is loaded. Furthermore putting the `script` at the end of the body will only delay the loading of the `script`, as it would be done sequentially.

So in conclusion, we can use a `defer` attribute so that the script can be loaded asynchronously while still waiting for the *HTML* to be parsed.

Question 2

Given this code:

```
<a href="#" class="js-btn">Hi!</a>
<script>
class MyApp {
  #btn;
  #label;

  constructor (btnTag) {
    this.#label = btnTag;
    this.#btn = document.querySelector('.js-btn');
    this.#btn.innerText = btnTag;

    this.#init(btnTag);
  };
  #init(btnTag) {
    this.#btn.addEventListener('click',this.greet);
  };

  greet(ev) {
    ev.preventDefault();
    console.log(this.#label);
  };
}

const app = new MyApp('Hello!');
</script>
```

The error in the code would be on the fact that the `greet` method accesses the private `#label` attribute. A private attribute can only be accessed from inside the class itself, and in this case a *DOM* object is trying to access this very attribute, which is not defined and will throw an error. To fix this, we must change the `addEventListener` call to the following using `bind`:

```
this.#btn.addEventListener('click',this.greet.bind(this));
```

Using `bind` with `this`, we ensure that the `this` inside the `greet` refers to the `MyApp` class, and thus the access is correct.

Question 3

Given this code:

```
function onClick(ev) {
    console.log('click!');
    document.addEventListener('click', onClickAlt);
}

function onClickAlt(ev) {
    console.log('alt!');
}

document.addEventListener('click', onClick);
```

The correct answer is `c`: The first click in the document will show `click!`, the second click will show `click! alt!` and the third click will show `click! alt!`.

The code adds a listener that reads for a click. When a click is detected, `click!` is printed and a second listener is set up to read a click as well, but prints `alt!` instead. Thus, if we press once we will see `click!` on the console and every subsequent click will print first `click!` and then `alt!`.

Question 4

Note

See the code in the `question4/` folder

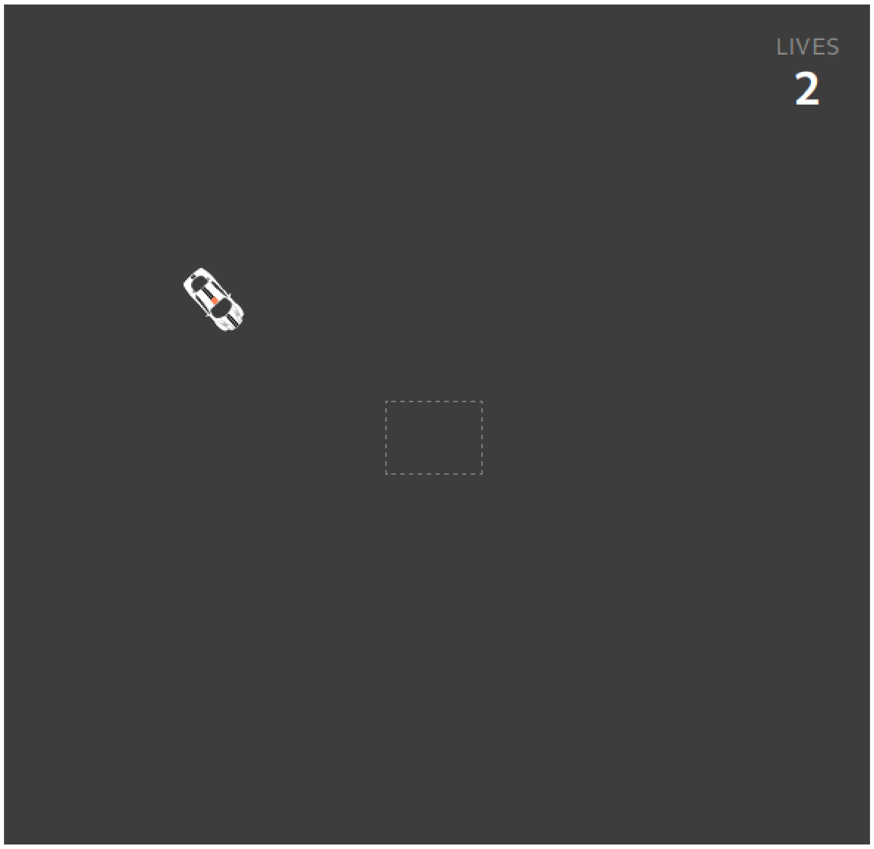

Figure 1: Gameplay Screenshot

See demo

Warning

To run the site, a local server must be running. The easiest way to do this is to run the project from VSCode (or similar) and use an extension like `Live Server`.