Question 4

Say a page has 3 buttons with class "item". What is wrong with the following code of binding click event to the buttons? How would you fix it before and after ES6?

```
var items = document.querySelectorAll(".item");
for (var i = 0; i < items.length; i += 1) {
    items[i].addEventListener("click", function () {
        console.log("You clicked on button # " + i);
    });
}
```

The problem with the provided code lies in the use of `var` to declare the `for loop` variable of `i`. Since `var` is function-scoped, the variable `i` is shared across all iterations of the loop. By the time the click event handler has executed (which happens asynchronously after the loop completes), the value of `i` will always be equal to `items.length`, and not the index of the button that was clicked.

Before ES6, I would fix it using an IIFE to create a new scope for each iteration of the loop, or in other words, isolate the `var` variable, ensuring that i has the correct value when the event handler is executed, as in the following code:

```
var items = document.querySelectorAll(".item");
for (var i = 0; i < items.length; i += 1) {
    (function (index) {  // index is a parameter of the IIFE
        items[index].addEventListener("click", function () {
            console.log("You clicked on button # " + index);  // the
current value of i is captured at each iteration
        });
    })(i);
}
```

After ES6, we can simply replace `var` with `let` to declare the `for loop` variable `i`. `let` is block-scoped, so a new `i` is created for each iteration of the loop, as in the following code:

```
const items = document.querySelectorAll(".item");  // might as well
replace this var as well
for (let i = 0; i < items.length; i += 1) {
     items[i].addEventListener("click", function () {
        console.log("You clicked on button # " + i);
    });
}
```