Closure in a loop

for(var i = 0; i <= 3; i++)

{

  setTimeout(function()

  {

    console.log(`after index seconds(s): ${i}`);

  }, i \* 1000);

}

//after index seconds(s): 4

//after index seconds(s): 4

//after index seconds(s): 4

//after index seconds(s): 4

What we wanted to do in the loop is to copy the value of  `i` in each iteration at the time of iteration to display a message after 1, 2, and 3 seconds.

The reason you see the same message after 4 seconds is that the callback passed to the `setTimeout()` a closure. It remembers the value of `i` from the last iteration of the loop, which is 4.

In addition, all three closures created by the for-loop share the same global scope access the same value of `i`.

Using IIFE Solution

for (var i = 0; i <= 3; i++) {

  ((i) => {

    setTimeout(function () {

      console.log(`after index seconds(s): ${i}`);

    }, i \* 1000);

  })(i);

}

//after index seconds(s): 0

//after index seconds(s): 1

//after index seconds(s): 2

//after index seconds(s): 3

Using let solution

for(let i = 0; i <= 3; i++)

{

  setTimeout(function()

  {

    console.log(`after index seconds(s): ${i}`);

  }, i \* 1000);

}

//after index seconds(s): 0

//after index seconds(s): 1

//after index seconds(s): 2

//after index seconds(s): 3