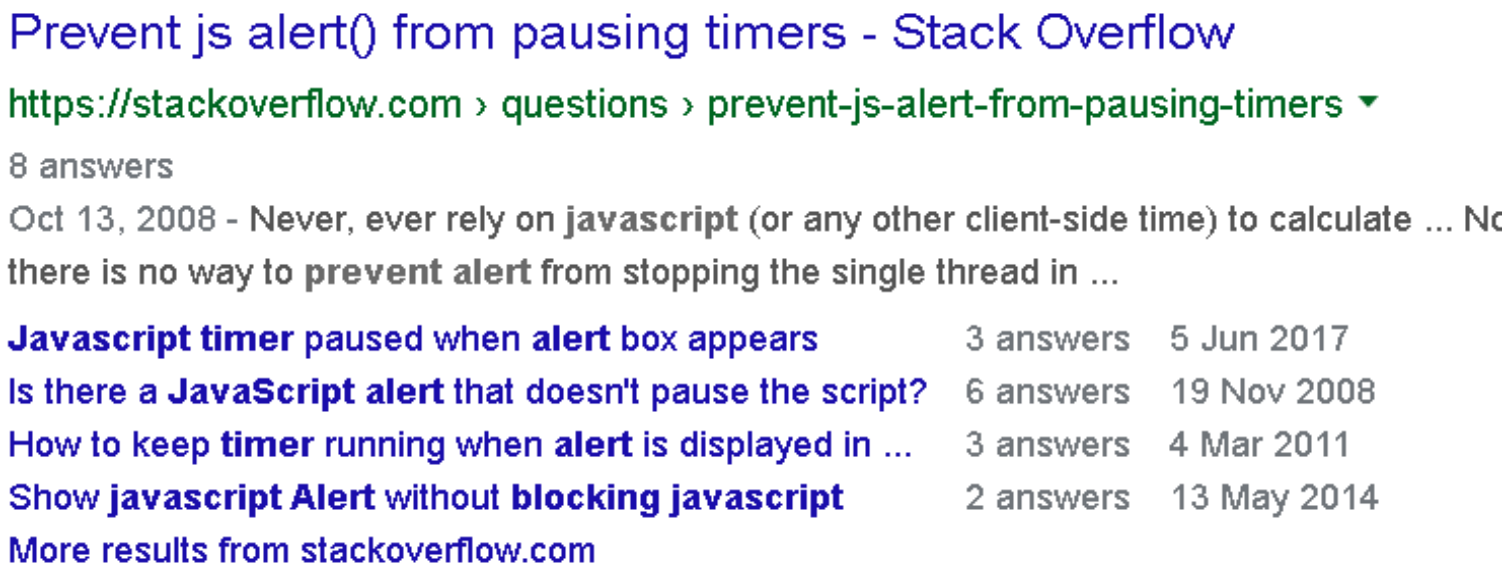
alert + setTimeout

- Does alert() block the timer?
- Google returns discussions 10+ years ago, but alert() no longer blocks (haven't find out when)



alert() does not Block Timer

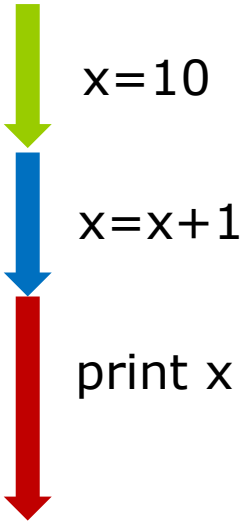
- setTimeout("alert('First Alert')", 0);
  setTimeout("alert('Second Alert')", 5000);
- When the 1st alert is displayed, click OK after 10 sec
- If 1st alert blocks the 2nd timer, then the 2nd alert will be displayed after 5 sec
- In fact, the 2nd alert is displayed immediately
- Conclusion:
  - 2nd timer runs while alert() is running
  - However, the 2nd alert is indeed blocked by the 1st alert

alert() Blocks JavaScript Thread

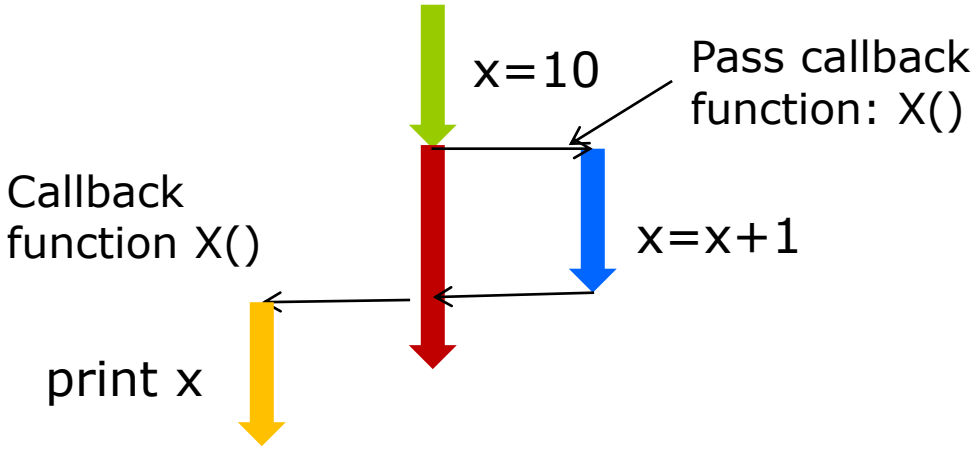
- JavaScript is single threaded (up to 2019)
- alert() blocks the thread, and the whole page halts; the only action allowed is to click "OK"
- Some people say this is desirable because it forces the user to focus
- If you do not want blocking (e.g., video continues to play), use DIV to emulate the alert box
  - The displayed message and style can be customized
  - jQuery provides customizable "alert" box

Synchronous vs Asynchronous Execution

- Synchronous operation: the thread executes the operation and when it is done continues with the next operation



- Asynchronous operation: the thread hands off the operation to another process, continues with the next operation immediately
- The calling process provides a callback function to "receive" the result



Alert() in Single-Threaded JavaScript

- In principle, if JavaScript is single-thread, so when alert() blocks the thread, setTimeout() runs prior to alert() will be blocked too
- The only solution is to run the timer "outside" the JavaScript thread as an asynchronous operation
- "…functions like setTimeout and setInterval are not part of the ECMAScript specs or any JavaScript engine implementations. Timer functions are implemented by browsers and their implementations will be different among different browsers."

  In "JavaScript Timers: Everything you need to know" (2018)
- This explains why timers are not blocked by alert()
- May say: timers are run asynchronously by browser

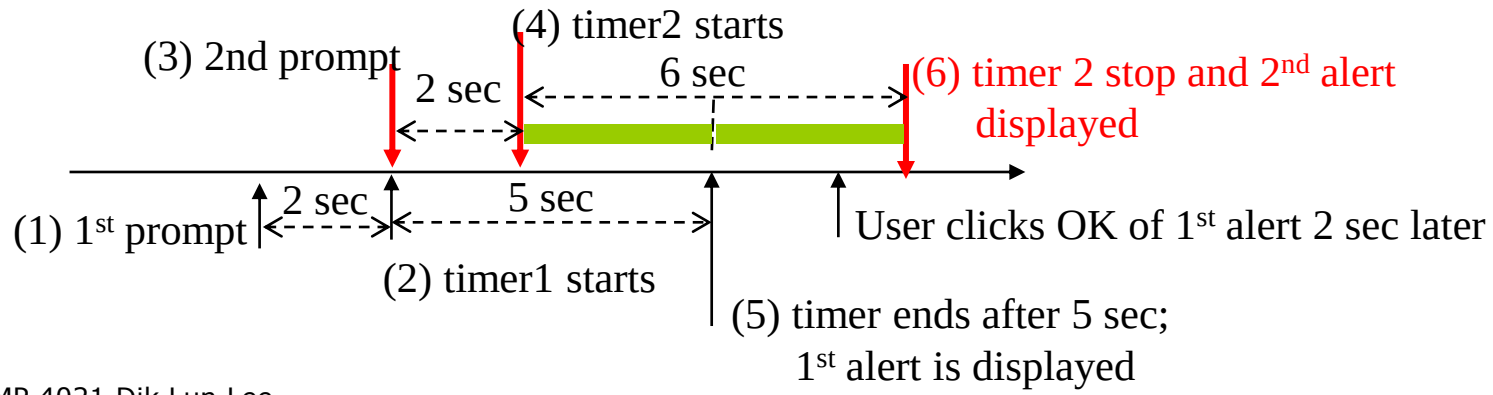
Fill in the Timeline

```
function set_things_up() {
        wait_duration=prompt("How long would you like to sleep?", "");       // (1)
        timer1=setTimeout("show_wake_up_message()", wait_duration );  // (2)

        wait_duration=prompt("How long until your next lecture?", "");       // (3)
        timer2=setTimeout("show_lecture_message()", wait_duration );  } // (4)

function show_wake_up_message() {
        alert("WAKE UP! WAKE UP! WAKE UP!!");   }     // (5)
function show_lecture_message() {
        alert("GO TO LECTURE! GO TO LECTURE!");   } // (6)
```

- The first timeout triggers an alert() blocking execution until "OK" is clicked
- Question: If timer1 is 5 sec and timer2 is 6 sec, user responds to a prompt in 2 sec, can you put 1-6 above on the following timeline?

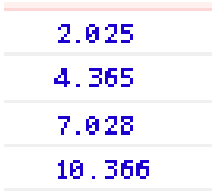


Run Example using console.log()

```
<script>
function set_things_up() {
   startTime=Date.now();
   wait_duration=prompt("How long would you like to sleep?", "");
   console.log((Date.now()-startTime)/1000);
   timer1=setTimeout("show_wake_up_message()", wait_duration*1000 );
   wait_duration=prompt("How long until your next lecture?", "");
   console.log((Date.now()-startTime)/1000);
   timer2=setTimeout("show_lecture_message()", wait_duration*1000 );  }
function show_wake_up_message() {
        console.log((Date.now()-startTime)/1000);
        alert("WAKE UP! WAKE UP! WAKE UP!!");   }
function show_lecture_message() {
        console.log((Date.now()-startTime)/1000);
        alert("GO TO LECTURE! GO TO LECTURE!");   }
</script>
```

Run code in TryIt

Output:



Wait Time is not Exact

```
<script>
setTimeout("console.log('1')", 0);
setTimeout("console.log('2')", 0);
console.log('3');
</script>
```

What is the output? 3, 1, 2

```
<script>
setTimeout("console.log('1')", 0);
setTimeout("console.log('2')", 0);
var start = Date.now();
while (Date.now() < start + 3000) {}
console.log('After busy wait');
console.log('3');
</script>
```

What is the output?

```
After busy wait
3
1
2
```

The two setTimeout() were executed outside the main thread. When finished, the two console.log() are queued for the main thread until the third console.log() is finished

Sleep (or Stupid) Sort

```
<script>
var dataSort="";
function appendNum(num) {
dataSort = dataSort + " " + num;
console.log(dataSort);
}

var data=[100, 5, 2, 9];
for (i=0; i<data.length; i++) {
setTimeout(appendNum, data[i], data[i]);
}
</script>
```

What does this script do?

Run it on TryIt