|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Timers** |  |

|  |
| --- |
| **Your Tasks (Mark these off as you go)** |
| * Explain the need for timers * Apply the *setTimeout()* function * Apply the *setInterval()* function * Stop a timing event * Have Ms. Pluska check off the above tasks * Brainstorm a program * Receive credit for the group portion of this lab |

* **Explain the need for timers**

Programmers use timing events to delay the execution of code or to repeat code at a specified interval. There are two native functions in the JavaScript library used to accomplish these tasks: *setTimeout()* and *setInterval()*.

The *setTimeout()* function is used to delay the execution of the passed function by a specified amount of time.

The *setInterval()* function is used to specify the time interval for which a function should be repeated.

Both *setTimeout()* and *setInterval()* allow us to make our applications more interesting by controlling the timing of our functions.

* **Apply the *setTimeout()* function**

You use *setTimeout()* to delay the execution of a function by a specified amount of time. There are two parameters that you pass to *setTimeout()*: the function you want to call, and the amount of time in milliseconds. (There is 1000 milliseconds(ms) in 1 second. Ex: 5000 ms = 5 seconds.) *setTimeout()* will execute one time after the specified time has elapsed.

|  |
| --- |
| var timer;  delayTimer();  function delayTimer() {  timer = setTimeout(delayedFunction, 3000);  }  function delayedFunction() {  alert(“Three seconds have elapsed.”);  } |

|  |
| --- |
| (a) Write a function called *gameOver* that alerts the user with the message “Game Over!” and their corresponding score.  (b) Write another function called *alertUser* that calls the function in part (a) after 10 seconds has elapsed.  (c) Call the function (b) |
| var score = 10; |

* **Apply the *setInterval()* function**

You use setInterval() to specify a function to repeat with a time delay between executions. Again, two parameters are required for setInterval(): the function you want to call, and the amount of time in milliseconds. setInterval() will continue to execute until it is cleared.

|  |
| --- |
| var timer;  repeatEverySecond();  function repeatEverySecond() {  timer = setInterval(sendMessage, 1000);  }  function sendMessage() {  var d = new Date();  document.body.innerHTML = d.toLocaleTimeString();  } |

|  |
| --- |
| (a) What does the code above do? |
|  |
| (b) Create a function called *updateTime* that deducts one from the *gameTime* and displays the new time on the body of the page  (c) Create a function called *countDown* which calls *updateTime* every second  (d) Call the function you wrote in part (c) |
| var gameTime = 10; |

* **Stop a timing event**

There are two corresponding native functions to stop the above timing events: *clearTimeout()* and *clearInterval()*.

You may have noticed that each timer function is saved to a variable. When the set function runs it is assigned a number which is saved to this variable. This generated number is unique for each instance of a timer. This assigned number is also how timers are identified to be stopped. For this reason, you must always set your timer to a variable.

To stop a timer, call the corresponding clear function and pass it the timer ID variable that matches the timer you wish to stop. The syntax for *clearInterval()* and *clearTimeout()* are the same and are illustrated below,

|  |
| --- |
| var timeoutID;  delayTimer();  function delayTimer() {  timeoutID = setTimeout(delayedFunction, 3000);  }  function delayedFunction() {  alert(“Three seconds have elapsed.”);  clearAlert();  }  function clearAlert() {  clearTimeout(timeoutID);  } |

|  |
| --- |
| (a) Write a function called *gameOver* that alerts the user with the message “Game Over!” and their corresponding score.  (b) In the gameOver function write a line of code that clears the *timeOut* timer event  (c) In the gameOver function write another line of code that clears the *timeLeft* timer event  (e) Call the appropriate functions to make your game work |
| var gameTimer;  var timeLeft;  var score = 10;  var gameTime = 15;  function gameTimer(){  timeOut = setTimeout(gameOver, gameTime\*1000);  }  function countDown() {  timeLeft = setInterval(updateTime, 1000);  }  function updateTime(){  gameTime--;  } |

* **Have Ms. Pluska check off the above tasks**



Before you continue have Ms. Pluska check off the above tasks

Do not continue until you have Ms. Pluska’s (or her designated TA’s) signature \_\_\_\_\_\_\_\_\_\_\_\_

* **Brainstorm a program**

With your partner, brainstorm code that could be used to solve each of the following challenges. Write your code on a separate sheet of paper and attach it to this lab.

The program you wrote previously placed a hidden monster randomly on a grid. The objective of the game was to locate the monster. Your task is refine your game to meet the following criteria:

- The monster should appear at a different location on the grid every second

- If the monster is clicked the score should go up by 1

- If the monster is missed the score should go down by 1

- After 10 seconds your game should alert the user that the game is over and their final score.

- When the game is over, all timing events should be cleared

On a separate sheet of paper brainstorm how you will modify your program to incorporate the above features.

* **Receive Credit for the group portion of this lab**



* Indicate the names of all group members.
* Make sure both you and your partner have completed the above tasks
* Have Ms. Pluska check off the group tasks
* Submit your lab to the needs to be graded folder to receive credit for the group portion of this lab.
* Do not submit your lab until you have Ms. Pluska’s (or her designated TA’s) signature

\_\_\_\_\_\_\_