7.5 Timers

Timeouts

Some events are related to time instead of user actions. Ex: A website may wish to display an advertisement 10 seconds after the webpage loads or display inventory data that updates at regular intervals. A ***timer*** is a general name for techniques to execute JavaScript code after some amount of time has occurred.

A web browser is able to execute a function after a time delay using `setTimeout()`. The ***setTimeout()*** method takes two arguments: a function and a time delay in milliseconds (1/1000th of a second). The browser calls the function after the time delay. `setTimeout()` returns a unique integer identifier that refers to the timeout that was created, and the timeout can be canceled by passing the identifier to ***clearTimeout()***.

PARTICIPATION ACTIVITY 7.5.1: Showing a daily special with setTimeout().

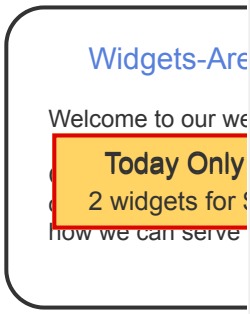
1 2 3 2x speed

```
<div id="special">
  <h1>Today Only!</h1>
  <p>2 widgets for $10!</p>
</div>
```

```
#special {
   display: none;
   border: solid red 1px;
   ...
}
```

```
let timerId = setTimeout(showSpecial, 3000);

function showSpecial() {
   let special = document.getElementById("special");
   special.style.display = "block";
}
```



After 3 seconds, the browser calls showSpecial() and makes the visible.

Captions

1. A webpage contains a with a daily special that is not yet visible.
2. setTimeout() tells the browser to call showSpecial() in 3 seconds.
3. After 3 seconds, the browser calls showSpecial() and makes the visible.

Feedback?

PARTICIPATION ACTIVITY 7.5.2: Timeouts.

Refer to the animation above.

1) How many times is `showSpecial()` called when the webpage is loaded just once?

- 0
- 1
- Every 3 seconds

Correct

`showSpecial()` is only called once because `setTimeout()` is only called once when the page is loaded. The timeout only calls the function.

2) What modification to `setTimeout()` displays the daily special 5 seconds after the page loads?

- `setTimeout(showSpecial, 5)`
- `setTimeout(showSpecial(), 5000)`
- `setTimeout(showSpecial, 5000)`

Correct

5000 ms is equivalent to 5 seconds.

3) Is the code below logically equivalent to the code in the animation?

```
let timerId = setTimeout(function()
{
   let special =
document.getElementById("special");
   special.style.display = "block";
}, 3000);
```

- Yes
- No

Correct

The first argument to `setTimeout()` can be an anonymous function or named function. The anonymous function performs the same logic as `showSpecial()`.

4) Suppose the code below is inserted immediately after the call to `setTimeout()`. What is different when the webpage loads?

```
clearTimeout(timerId);
```

- ○ No change.
- ◉ The daily special never appears.
- ○ The daily special appears immediately.

Correct

`clearTimeout()` cancels the timeout created by `setTimeout()`, so `showSpecial()` is never called.

5) What is missing to hide the daily special after the special has displayed for 10 seconds?

```
function showSpecial() {
   let special =
document.getElementById("special");
   special.style.display = "block";
   ____;
}

function hideSpecial() {
   let special =
document.getElementById("special");
   special.style.display = "none";
}
```

- ◉ `setTimeout(hideSpecial, 10000)`
- ○ `hideSpecial()`
- ○ `clearTimeout(timerId)`

Correct

`hideSpecial()` is called after 10,000 ms to hide the special. The return value from `setTimeout()` is not saved in a variable because the timeout ID is not needed.

Feedback?

Intervals

A web browser is able to execute a function repeatedly with a time delay between calls using `setInterval()`. The ***setInterval()*** method takes two arguments: a function and a time interval in milliseconds (*t*). The browser calls the function every *t* milliseconds until the interval is canceled. The `setInterval()` method returns the interval's unique integer identifier, and the interval identifier can be passed to the ***clearInterval()*** method to cancel the interval.

PARTICIPATION ACTIVITY 7.5.3: Animating a ball with setInterval().

1 2 3 4 5 2x speed

```
<img src="ball.png" id="ball" alt="ball"
   style="position:absolute; left:0">
```

```
let ballImage;
let timerId;

function startMoving() {
   ballImage = document.getElementById("ball");
   timerId = setInterval(moveBall, 10);
}

function moveBall() {
   let left = parseInt(ballImage.style.left);
   ballImage.style.left = left + 5 + "px";
}
```

The interval calls moveBall() every 10 ms, animating the ball to the right. Eventually the ball is no longer on the screen.

Captions

1. The HTML and CSS place a ball image on the left side of the browser screen.
2. When startMoving() is called, setInterval() creates an interval that calls moveBall() every 10 ms.
3. left is assigned the image's left CSS property, converted into an integer.
4. The image's left CSS property is assigned a px value 5 more than before, moving the ball 5 pixels to the right.
5. The interval calls moveBall() every 10 ms, animating the ball to the right. Eventually the ball is no longer on the screen.

Feedback?

PARTICIPATION ACTIVITY 7.5.4: Intervals.

Refer to the animation above.

1) The code below calls `moveBall()` less frequently than the animation does.

```
timerId =
setInterval(moveBall,
20);
```

- (●) True
- ( ) False

Correct

An interval delay of 20 ms means `moveBall()` is called half as frequently as when the interval delay is 10 ms.

2) How many times is `moveBall()` called to move the ball 100 pixels?

- ( ) 5
- (●) 20

Correct

`moveBall()` moves the ball 5 pixels each time. Moving the ball 100 pixels takes 100 / 5 = 20 calls to `moveBall()`.

3) How long does the ball take to move 100 pixels?

- ( ) 1 second
- (●) 200 ms

Correct

Moving the ball 100 pixels takes 100 / 5 = 20 calls to `moveBall()`. The interval is 10 ms, so 20 calls × 10 ms = 200 ms.

4) `moveBall()` is not called anymore after the ball moves off the screen.

- ( ) True
- (●) False

Correct

The interval continues to run since `clearInterval()` is never called.

5) What is missing to stop `moveBall()` from being called after the ball reaches the browser edge?

```
function moveBall() {
   let left =
parseInt(ballImage.style.left);
   if (left + ballImage.width
> document.body.clientWidth) {
      ____;
   } else {
      ballImage.style.left =
left + 2 + "px";
   }
}
```

- (●) clearInterval(timerId)
- ( ) clearTimeout(timerId)

Correct

`clearInterval()` cancels the interval so `moveBall()` is no longer called.

6) The code below moves the ball in the same direction as the animation above.

```
function moveBall(distance) {
   let left =
parseInt(ballImage.style.left);
   ballImage.style.left = left
+ distance + "px";
}

setInterval(function() {
   moveBall(-5);
}, 20);
```

- ○ True
- ◉ False

Correct

The anonymous function calls `moveBall()` with -5, so `moveBall()` subtracts 5 each time from the ball's left position, moving the ball to the left instead of moving the ball to the right.

Additional arguments can also be passed to a timer function by specifying the arguments in a `setTimeout()` or `setInterval()` function call after the millisecond argument. Ex: `setInterval(moveBall, 20, -5);`

Feedback?

PARTICIPATION ACTIVITY 7.5.5: Intervals.

Modify the JavaScript to create a countdown timer.

1. Add code to `startbutton`'s click event handler to start an interval that calls the `countdown()` function every second.
2. Store the unique identifier returned by `setInterval()` in `countdownTimerId` so the interval can be canceled.
3. Add code to `countdown()` to clear the countdown interval.
4. Add code to `stopbutton`'s click event handler to clear the countdown interval.

HTML | JavaScript

```
1 <body>
2    <p>Enter the countdown starting number, then click the start b
3    <input type="number" id="number" min="0" value="5">
4    <input type="button" id="startbutton" value="start">
5    <input type="button" id="stopbutton" value="stop" disabled>
6 </body>
7
```

Render webpage Reset code

Your webpage

Enter the countdown starting number, then click the start button.

5 start stop

Expected webpage

Enter the countdown starting number, then click the start button.

5 start stop

▶ View solution

Feedback?

CHALLENGE ACTIVITY 7.5.1: Timers.

530096.4000608.qx3zqy7

Jump to level 1

1 2 3

Write a setInterval() function that increases the count by 1 and displays the new count in counterElement every 200 milliseconds. Call clearInterval() to cancel the interval when the count displays 4.

HTML JavaScript

```
let count = 0;
let counterElement = document.getElementById("counter");
counterElement.innerHTML = count;


let intervalId = setInterval(function() {
    count++;
    counterElement.innerHTML = count;

    // Check if the count is 4, and if so, clear the interval
    if (count === 4) {
        clearInterval(intervalId);
    }
}, 200);
```

1 2 3

Check Next **Done**. Click any level to practice more. Completion is preserved.

✔

✔ Checking counterElement.innerHTML when interval is halfway complete.

Yours 2

✔ Checking counterElement.innerHTML when interval is canceled.

Yours 4

✔ Checking counterElement.innerHTML shortly after interval is canceled.

Yours 4

Feedback?

Exploring further:

- Scheduling: setTimeout and setInterval from javascript.info
- JavaScript Timing Events from w3schools

How was this section? **Provide section feedback**