Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to create a countdown timer

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 5

Aim: Write a JavaScript program that will create a countdown timer

Prerequisites:

- For this **Javascript Lab**, it is assumed that you have a prior knowledge of HTML coding. It would help if you had some prior exposure to object-oriented programming concepts and a general idea on creating online applications.
- To understand this experiment, you should have the knowledge of the following JavaScript programming topics:

✓ JavaScript Math floor()
✓ JavaScript Date and Time
✓ Javascript setInterval()

Editor:

1.	NotePad
2.	Visual studio code

Theory:

A countdown timer is an accurate timer that can be used for a website or blog to display the countdown to any special event, such as a birthday or anniversary. For countdown timer you know the following methods;

1) The clearInterval() Method
2) The setinterval() Method
3) The setTimeout() Method
4) The clearTimeout() Method

1) Window setInterval() :

Definition and Usage:

The setInterval() method calls a function at specified intervals (in milliseconds).

The setInterval() method continues calling the function until clearInterval() is called, or the window is closed.

1 second = 1000 milliseconds.

Note:

To execute the function only once, use the setTimeout() method instead.

To clear an interval, use the **id** returned from setInterval():

```
myInterval = setInterval(function, milliseconds);
```

Then you can to stop the execution by calling clearInterval():

```
clearInterval(myInterval);
```

Syntax

```
setInterval(function, milliseconds, param1, param2, ...)
```

Parameters

Parameter	Description
function	Required. The function to execute
milliseconds	Required. The execution interval. If the value is less than 10, 10 is used

param1, *param2*, ...	Optional. Additional parameters to pass to the *function* Not supported in IE9 and earlier.

Return Value

Type	Description
A number	The id of the timer. Use this id with clearInterval() to cancel the timer.

Examples

Display "Hello" every second (1000 milliseconds):

```
setInterval(function () {element.innerHTML += "Hello"}, 1000);
```

```
<html>
<body>
<h1>The Window Object</h1>
<h2>The setInterval() Method</h2>
<p id="demo"></p>
<script>
const element = document.getElementById("demo");
setInterval(function() {element.innerHTML += "Hello"}, 1000);
</script>
</body>
</html>
```

OR

Call displayHello every second:

```
setInterval(displayHello, 1000);
```

```
<html>
<body>

<h1>The Window Object</h1>
<h2>The setInterval() Method</h2>

<p id="demo"></p>

<script>
setInterval(displayHello, 1000);

function displayHello() {
  document.getElementById("demo").innerHTML += "Hello";
}
</script>

</body>
</html>
```

Output:

2) clearInterval()

Definition and Usage

The clearInterval() method clears a timer set with the setInterval() method.

Note

To clear an interval, use the id returned from setInterval():

```
myInterval = setInterval(function, milliseconds);
```

Then you can to stop the execution by calling clearInterval():

```
clearInterval(myInterval);
```

Syntax

clearInterval(*intervalId*)

Parameters

Parameter	Description
intervalId	Required. The interval id returned from setInterval().

3) setTimeout()

Definition and Usage

The setTimeout() method calls a function after a number of milliseconds.

1 second = 1000 milliseconds.

Note:

The setTimeout() is executed only once.

If you need repeated executions, use setInterval() instead.

Use the clearTimeout() method to prevent the function from starting.

To clear a timeout, use the **id** returned from setTimeout():

```
myTimeout = setTimeout(function, milliseconds);
```

Then you can to stop the execution by calling clearTimeout():

```
clearTimeout(myTimeout);
```

Syntax:

```
setTimeout(function, milliseconds, param1, param2, ...)
```

Parameters

Parameter	Description
function	Required. The function to execute.
milliseconds	Optional. Number of milliseconds to wait before executing. Default value is 0.
param1, *param2*, ...	Optional. Parameters to pass to the *function.* Not supported in IE9 and earlier.

Return Value

Type	Description
A number	The id of the timer. Use this id with clearTimeout(id) to cancel the timer.

4) clearTimeout()

Definition and Usage

The clearTimeout() method clears a timer set with the setTimeout() method.

Note:

To clear a timeout, use the **id** returned from setTimeout():

myTimeout = setTimeout(*function*, *milliseconds*);

Then you can to stop the execution by calling clearTimeout():

```
clearTimeout(myTimeout);
```

Syntax:

```
clearTimeout(id_of_settimeout)
```

Parameters

Parameter	Description
timeout id	Required. The id returned by the setTimeout() method.

JavaScript Math floor()

The Math.floor() function rounds down a number to the next smallest integer.

Example

```
let number = 38.8;

// round number to nearest smallest number
let roundedNumber = Math.floor(number);

console.log(roundedNumber);

// Output: 38
```

Math.floor() Syntax

The syntax of the Math.floor() function is:

```
Math.floor(x)
```

floor(), being a static method, is called using the Math class name.

Math.floor() Parameters

The Math.floor() function takes in:

- x - A number

Math.floor() Return Value

- Returns the largest integer less than or equal to a given number.
- Returns 0 for null.

STEPS:

Steps of a countdown timer are:

1. Set a valid end date.
2. Calculate the time remaining.
3. Convert the time to a usable format.
4. Output the clock data as a reusable object.
5. Display the clock on the page, and stop the clock when it reaches zero.

Step 1 : Set a Valid End Date
The Valid end date and time should be a string in any of the formats understood by JavaScript's Date.parse() method.
To create a countdown timer using Javascript, we first need to declare a variable that holds the date and time that we want our countdown timer to run down to. We create a **Date** object and then call the **getTime() function** on this object. The **getTime() method** makes the **countDownDate** variable hold the milliseconds since Jan 1, 1970, 00:00:00.000 GMT.

```
var deadline = new Date("dec 31, 2017 15:37:25").getTime();
```

Step 2 : Calculate Remaining Time

First we calculate the time remaining by subtracting the deadline by current date and time then we calculate the number of days,hours,minutes and seconds. The Math.floor() function is used to return the largest integer less than or equal to a given number.

```
var now = new Date().getTime();
var t = deadline - now;
var days = Math.floor(t / (1000 * 60 * 60 * 24));
var hours = Math.floor((t%(1000 * 60 * 60 * 24))/(1000 * 60 * 60));
var minutes = Math.floor((t % (1000 * 60 * 60)) / (1000 * 60));
var seconds = Math.floor((t % (1000 * 60)) / 1000);
```

Step 3: Output the result
In the code below the result is given as output by id="demo"

```
document.getElementById("demo").innerHTML = days + "d " + hours +
"h " + minutes + "m " + seconds + "s ";
```

Step 4: Write some text if the countdown is over
If the countdown timer is over then “expired” will be displayed on the screen.

```
    if (t < 0) {
        clearInterval(x);
        document.getElementById("demo").innerHTML = "EXPIRED";
    }
}, 1000);
```

Program:

```
        <!-- Display the countdown timer in an element -->
<p id="demo"></p>

<script>
// Set the date we're counting down to
var countDownDate = new Date("Sep 1, 2022 15:35:25").getTime();

// Update the count down every 1 second
var x = setInterval(function() {

  // Get today's date and time
  var now = new Date().getTime();

  // Find the distance / Timeleft between now and the countdown date
  var distance = countDownDate - now;

  // Time calculations for days, hours, minutes and seconds
  var days = Math.floor(distance / (1000 * 60 * 60 * 24));
  var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
  var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
  var seconds = Math.floor((distance % (1000 * 60)) / 1000);

  // Display the result in the element with id="demo"
  document.getElementById("demo").innerHTML = days + "d " + hours + "h "
  + minutes + "m " + seconds + "s ";

  // If the count down is finished, write some text
  if (distance < 0) {
    clearInterval(x);
    document.getElementById("demo").innerHTML = "EXPIRED";
```

```
  }
}, 1000);
</script>
```

Output:

```
0d 0h 0m 23s
```

EXPIRED

Applications of Countdown Timer

- Used during Events to display the time left for its commencement.
- Used by online commerce websites to display time left for an ongoing sale.
- Used by websites during promotions
- Used in Car racing games,football games etc
- Used in Auction Websites to display the left for placing bids.

Benefits of making a countdown timer in JavaScript than using plugins

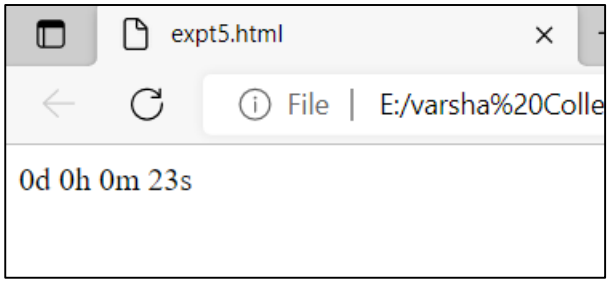
- The code will be lightweight because it will have zero dependencies.
- The website will perform better because there won't be any need of loading external scripts and style sheets.
- The user gets more control because he has built the clock to behave exactly the way he wants it to rather than trying to bend a plugin according to his will.

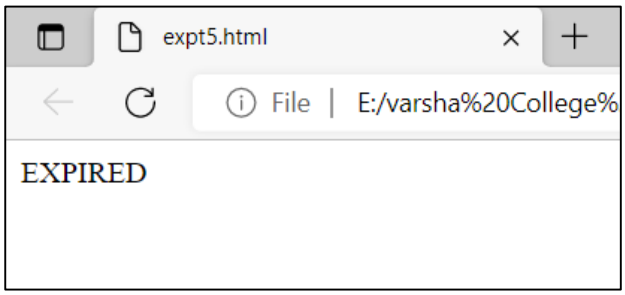
Program:

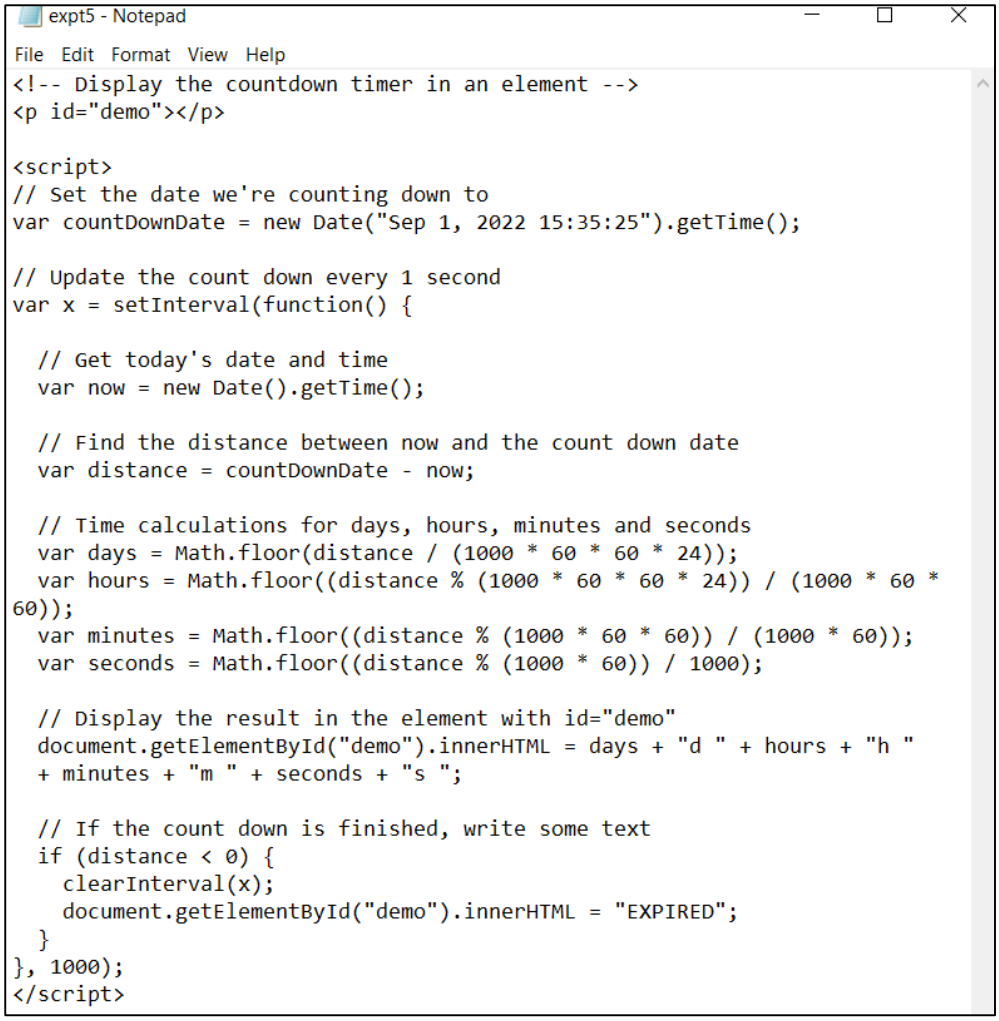
```
<!-- Display the countdown timer in an element -->
<p id="demo"></p>

<script>
// Set the date we're counting down to
var countDownDate = new Date("Sep 1, 2022 15:35:25").getTime();

// Update the count down every 1 second
var x = setInterval(function() {

  // Get today's date and time
  var now = new Date().getTime();

  // Find the distance between now and the count down date
  var distance = countDownDate - now;

  // Time calculations for days, hours, minutes and seconds
  var days = Math.floor(distance / (1000 * 60 * 60 * 24));
```

```
  var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
  var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
  var seconds = Math.floor((distance % (1000 * 60)) / 1000);

  // Display the result in the element with id="demo"
  document.getElementById("demo").innerHTML = days + "d " + hours + "h "
  + minutes + "m " + seconds + "s ";

  // If the count down is finished, write some text
  if (distance < 0) {
    clearInterval(x);
    document.getElementById("demo").innerHTML = "EXPIRED";
  }
}, 1000);
</script>
```

Screenshot's of Output:






expt5 - Notepad
File Edit Format View Help
<!-- Display the countdown timer in an element -->

<script>
// Set the date we're counting down to
var countDownDate = new Date("Sep 1, 2022 15:35:25").getTime();
// Update the count down every 1 second
var x = setInterval(function() {
// Get today's date and time
var now = new Date().getTime();
// Find the distance between now and the count down date
var distance = countDownDate - now;
// Time calculations for days, hours, minutes and seconds
var days = Math.floor(distance / (1000 * 60 * 60 * 24));
var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 *
60));
var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
var seconds = Math.floor((distance % (1000 * 60)) / 1000);
// Display the result in the element with id="demo"
document.getElementById("demo").innerHTML = days + "d " + hours + "h "
+ minutes + "m " + seconds + "s ";
// If the count down is finished, write some text
if (distance < 0) {
clearInterval(x);
document.getElementById("demo").innerHTML = "EXPIRED";
}
}, 1000);
</script>

Conclusion: