# Assignment 1 JavaScript Project

|  |  |
| --- | --- |
| Project Title | StopWatch |
| Students Name and Roll Number | Hasnain Afzal 21014119-022  Uzair Ahmed 21014119-019 |
| GitHub Link | Paste GitHub link here. |
| YouTube Link | Paste YouTube video link here. |

# Project Description

**Working**

HTML Structure:

The HTML file includes a set of buttons for starting, stopping, resetting, and recording

laps.

There is a display area for the stopwatch time and an unordered list to display lap times.

JavaScript Logic:

The core of the stopwatch logic is encapsulated in the myStopwatch object, which stores

the state and data related to the stopwatch.

The updateDisplay function formats the elapsed time and updates the content of the

display element.

The pad function adds leading zeros to single-digit numbers for better formatting.

The startStopwatch function starts the stopwatch and returns a promise that resolves when

the stopwatch is stopped. It uses an interval to update the display every 100 milliseconds.

The stopStopwatch function stops the running stopwatch.

When the start button is clicked, it triggers the startStopwatch function, which

returns a promise. The then block logs a message to the console when the

stopwatch is stopped.

The lap button triggers the lapStopwatch function.

The stop button triggers the stopStopwatch function.

The reset button triggers the stopStopwatch and resetStopwatch functions.

Running the Stopwatch:

Clicking the "Start" button starts the stopwatch, and the display updates in real-

time.

Clicking the "Lap" button records lap times, and they are displayed in the lap

times list.

Clicking the "Stop" button pauses the stopwatch.

Clicking the "Reset" button stops the stopwatch and resets the timeandlaptimes.

# Detail about project’s components

**Section 1: Objects**

myStopwatch Object:

The myStopwatch object serves as a container for various properties and methods

related to the stopwatch. It encapsulates the state of the stopwatch, including the start

time, whether it's running, the elapsed time, and an array to store lap times.

Properties:

startTime: Represents the time when the stopwatch was last started.

running: Indicates whether the stopwatch is currently running or not.

elapsedTime: Represents the total elapsed time of the stopwatch.

lapTimes: An array to store lap times recorded during the stopwatch run.

Methods:

updateDisplay(): Updates the display of the stopwatch by formatting the elapsed time.

pad(number): A utility method to pad single-digit numbers with a leading zero.

startStopwatch(): Starts the stopwatch, updating the elapsed time at intervals.

stopStopwatch(): Stops the running stopwatch.

resetStopwatch(): Resets the stopwatch to zero and clears the lap times array.

lapStopwatch(): Records a lap time when the stopwatch is running.

updateLapTimesList(): Updates the lap times list in the HTML.

Stopwatch object

const myStopwatch = {

startTime: null,

running: false,

elapsedTime: 0,

lapTimes: [],

updateDisplay() {

},

pad(number) {

startStopwatch() {

},

stopStopwatch() {

},

resetStopwatch() {

},

lapStopwatch() {

},

updateLapTimesList() {

},

};

},

document.getElementById('startButton').addEventListener('click', () => {

myStopwatch.startStopwatch().then(() => {

console.log('Stopwatch stopped!');

});

});

document.getElementById('lapButton').addEventListener('click', () => {

myStopwatch.lapStopwatch();

});

document.getElementById('stopButton').addEventListener('click', () => {

myStopwatch.stopStopwatch();

});

document.getElementById('resetButton').addEventListener('click', () => {

myStopwatch.stopStopwatch();

myStopwatch.resetStopwatch();

});

# Section 2: Arrays

myStopwatch.lapTimes:

This array stores the lap times recorded during the stopwatch run. Each element in the

array represents the duration of a lap.

updateLapTimesList Function:

The updateLapTimesList function iterates over the myStopwatch.lapTimes array to

create a list of lap times in the HTML. It uses the array to dynamically generate the

content displayed on the webpage.

Code Related to Array

const myStopwatch = {

startTime: null,

running: false,

elapsedTime: 0,

lapTimes: [],

};

function lapStopwatch() {

if (myStopwatch.running) {

const lapTime = Date.now() - myStopwatch.startTime;

myStopwatch.lapTimes.push(lapTime);

updateLapTimesList();

}

}

function updateLapTimesList() {

const lapTimesList = document.getElementById('lapTimesList');

lapTimesList.innerHTML = '';

myStopwatch.lapTimes.forEach((lapTime, index) => {

const lapItem = document.createElement('li');

const minutes = Math.floor(lapTime / 1000 / 60);

const seconds = Math.floor((lapTime / 1000) % 60);

const milliseconds = lapTime % 1000;

lapItem.textContent = `Lap ${index + 1}: ${pad(minutes)}:${pad(seconds)}.${milliseconds}`;

lapTimesList.appendChild(lapItem);

});

}

**Section 3: Callback Functions**

setInterval Callback:

In the startStopwatch function, there is the use of setInterval, which takes a callback function to be

executed at specified intervals. This callback function updates the display and resolves the promise

when the stopwatch is stopped.

Promise Callback:

The startStopwatch function also returns a Promise, and the resolve function in the promise is

essentially a callback that gets executed when the promise is resolved. This is where the code to

stop the interval and resolve the promise is placed.

Event Listener Callbacks:

Event listeners are attached to various buttons, and the callback functions associated with these

events handle user interactions. For example, the callback for the "Start" button click triggers the

startStopwatch function.

function startStopwatch() {

if (!myStopwatch.running) {

myStopwatch.startTime = Date.now() - myStopwatch.elapsedTime;

myStopwatch.running = true;

return new Promise(resolve => {

const timerInterval = setInterval(() => {

if (myStopwatch.running) {

myStopwatch.elapsedTime = Date.now() - myStopwatch.startTime;

updateDisplay();

} else {

clearInterval(timerInterval);

resolve();

}

}, 100);

});

}

}

document.getElementById('startButton').addEventListener('click', () => {

startStopwatch().then(() => {

console.log('Stopwatch stopped!');

});

});

document.getElementById('lapButton').addEventListener('click', () => {

lapStopwatch();

});

document.getElementById('stopButton').addEventListener('click', () => {

stopStopwatch();

});

document.getElementById('resetButton').addEventListener('click', () => {

stopStopwatch();

resetStopwatch();

});

# Section 4: Promises

startStopwatch Function:

The startStopwatch function returns a promise. This promise is used to signal the completion

of the asynchronous operation (interval) that updates the display.

resolve Callback in Promise:

The resolve function within the promise acts as a callback that is executed when the

asynchronous operation is complete. In this case, it clears the interval and resolves the

promise.

Event Listener for "Start" Button:

The startStopwatch function is called when the "Start" button is clicked. The then method is

used to specify a callback that will be executed when the promise is resolved. This is where

you can put code that should run after the stopwatch is stopped.

function startStopwatch() {

if (!myStopwatch.running) {

myStopwatch.startTime = Date.now() - myStopwatch.elapsedTime;

myStopwatch.running = true;

return new Promise(resolve => {

const timerInterval = setInterval(() => {

if (myStopwatch.running) {

myStopwatch.elapsedTime = Date.now() - myStopwatch.startTime;

updateDisplay();

} else {

clearInterval(timerInterval);

resolve(); }

}, 100);

}); }}

document.getElementById('startButton').addEventListener('click', () => {

startStopwatch().then(() => {

});

**Whole code of my Project**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Stopwatch</title>

</head>

<body>

<div id="display">00:00:00</div>

<button id="startButton">Start</button>

<button id="lapButton">Lap</button>

<button id="stopButton">Stop</button>

<button id="resetButton">Reset</button>

<ul id="lapTimesList"></ul>

<script>

const myStopwatch = { startTime: null, running: false, elapsedTime: 0, lapTimes: [],

};

function updateDisplay() {

const seconds = Math.floor(myStopwatch.elapsedTime / 1000); const minutes = Math.floor(seconds / 60);

const hours = Math.floor(minutes / 60);

const displayTime = `${pad(hours)}:${pad(minutes % 60)}:${pad(seconds % 60)}`; document.getElementById('display').innerText = displayTime;

}

function pad(number) {

return number < 10 ? `0${number}` : number;

}

solve();

function startStopwatch() {

if (!myStopwatch.running) {

myStopwatch.startTime = Date.now() - myStopwatch.elapsedTime; myStopwatch.running = true;

return new Promise(resolve => {

const timerInterval = setInterval(() => { if (myStopwatch.running) {

myStopwatch.elapsedTime = Date.now() - myStopwatch.startTime; updateDisplay();

} else { clearInterval(timerInterval); re

}

}, 100);

});

}

}

function stopStopwatch() { myStopwatch.running = false;

}

function resetStopwatch() { myStopwatch.elapsedTime = 0; myStopwatch.lapTimes = []; updateDisplay(); updateLapTimesList();

}

function lapStopwatch() {

if (myStopwatch.running) {

const lapTime = Date.now() - myStopwatch.startTime; myStopwatch.lapTimes.push(lapTime); updateLapTimesList();

}

}

function updateLapTimesList() {

const lapTimesList = document.getElementById('lapTimesList'); lapTimesList.innerHTML = '';

myStopwatch.lapTimes.forEach((lapTime, index) => { const lapItem = document.createElement('li');

const minutes = Math.floor(lapTime / 1000 / 60); const seconds = Math.floor((lapTime / 1000) % 60); const milliseconds = lapTime % 1000;

lapItem.textContent = `Lap ${index + 1}: ${pad(minutes)}:${pad(seconds)}.${milliseconds}`; lapTimesList.appendChild(lapItem);

});

}

document.getElementById('startButton').addEventListener('click', () => { startStopwatch().then(() => {

console.log('Stopwatch stopped!');

});

});

document.getElementById('lapButton').addEventListener('click', () => { lapStopwatch();

});

document.getElementById('stopButton').addEventListener('click', () => { stopStopwatch();

});

document.getElementById('resetButton').addEventListener('click', () => { stopStopwatch();

resetStopwatch();

});

</script>

</body>

</html>