ClickTime Project Documentation

(The-Race-Around-The-World)

Angelo Reyes

Repo Link: ***https://github.com/angiereyes99/The-Race-Around-The-World***

Table of Contents

1 Introduction 3
1.1 Assumptions Made 3
1.2 Project Overview 3
1.3 Technical Overview 3
1.4 Summary of Work Completed 3
2 Development Environment 3
3 How to Build/Import your Project 3
4 How to Run your Project 4
5 Implementation Discussion 4
5.1 Class Diagram 5
6 Project Reflection 6
7 Project Conclusion/Results 6

1 Introduction

1.1 Assumptions Made

Some assumptions I made for the problem were, when you click the start button, the time you started will be displayed in the history table and the history table will only be updated once you hit “resume”. And hitting resume will add entries to the time table. Another assumption I made, was recording entries in the time table will always display time, time zone, and the longitude and latitude. For the time zone aspect, I assumed it was okay to display the full date and the area we are in (i.e. Pacific).

1.2 Project Overview

The program in this repository is a stopwatch that uses a stop, resume, and reset button to interact with the watch and record data that includes time, time zone, and longitude and latitude measurements in a time table.

1.3 Technical Overview

Using Node.js, HTML/CSS, and Express, I implemented a frontend interface displaying a stop watch that records data in a history time table whenever the user clicks on the start and resume button. The program is run locally on localhost:8080 by running commands on the terminal. The server is ran on a js file named “app.js”. The time recorded is displayed on a history time table which also displays the elapsed time, time zone, longitude, and latitude.

1.4 Summary of Work Completed

The key highlights of the work I completed was mainly involved in the watchTime.ejs file. Almost all of the HTML elements were initialized in the JavaScript file along with their action events. I created the mandatory buttons start, stop, resume and reset in the file, that would be passed on to HTML. I was able to develop action events for each button and also used the geolocation API to obtain user’s longitude and latitude. I also used the Date() function to get current dates and time zone to be displayed in the time table when the start and resume button were clicked on.

2 Development Environment

I developed Click Time’s stop watch in the text editor Brackets, and ran the program locally using the terminal and its command: “node app.js”.

3 How to Build/Import your Project

To import this project (Assuming you have git installed in your device), clone the SSH or HTTPS key in the GitHub link above in the title. Open your terminal or GitBash and type: **“git clone -SSH or HTTPS keys you copied -“**

Once you have successfully cloned the repo, you should be able to see the folder in your computer’s desktop.

Afterwards, go on your terminal and direct yourself to the project folder. Follow the directions found in the README file in the repo and hit “node app.js” to run the program on localhost:8080.

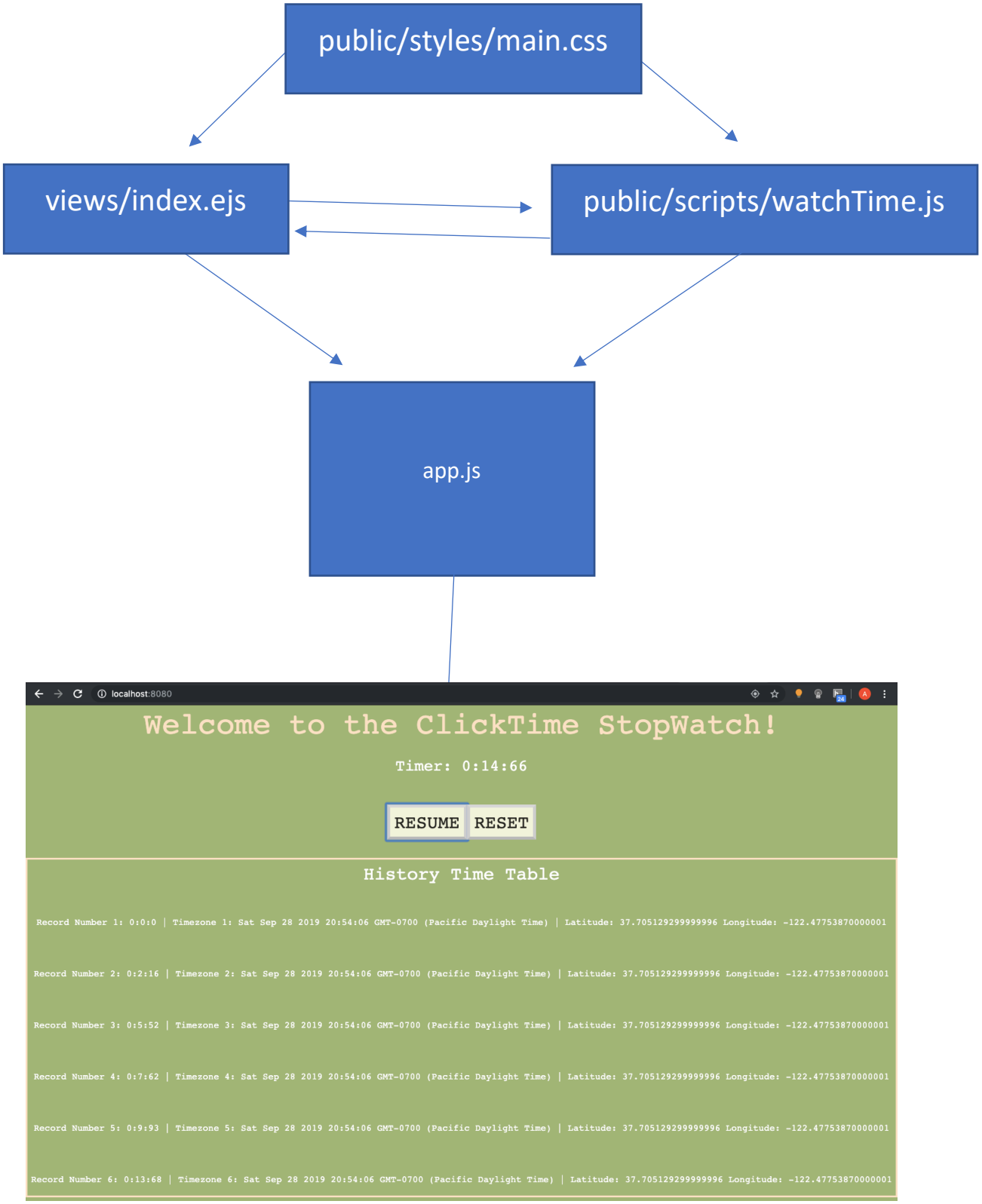
4 How to Run your Project

After connecting the server to localhost:8080, the stop watch interface should appear on the site. To simply run the program, all you have to do is hit the “START” button and the timer will start as well as enter in an entry in the history time table.

5 Implementation Discussion

The program is primarily developed through my implementation in the watchTime.js. I implemented all the HTML button, label, and tag elements in the file, as well as all the object action events. I also implemented a table that will serve as the history time table for the recorded time. I created a timer(), start(), and reset() function. For the timer function, I created the id “timer” and used innerHTML to display the “Timer:” label. It also displays the minutes, seconds, and milliseconds when the user hits start or resume. The start() function also holds the navigator. Function that runs when the start() function is true. I used text nodes and appendChild() functions to display the recorded data in history time table. For the reset() button, I take in the variable that holds the text nodes of the time table, and loops through each element so that it can delete every child element. This way, it also deletes the time table and timer.

5.1 Class Diagram



6 Project Reflection

The overall experience of this project was a good one, as I was able to work on my JavaScript skills as well as learn more about the geolocation API. I enjoyed developing the stop watch and all its functions and to see it work the way I wanted it to.

I also enjoyed the trial and errors throughout the project as there were numerous times I was stuck in a problem and had a hard time fixing it. These errors improved my self-sufficiency because I had to be able to use my problem-solving techniques to conquer certain problems!

Another important aspect I reflected on was the choice of code I had to use because even if the program runs properly, you want to make sure your code is readable and maintainable. With this thought in mind, I tried to implement a majority of the program in JavaScript. This left my HTML template with little lines of code and made it presentable.

7 Project Conclusion/Results

I was able to create the program correctly in every function in the way I interpreted the problem and the assumptions I made. All buttons work properly and enters the correct data in the history time table. A little fix I would try to make is that sometimes when you run the program and hit the start button, there could be a delay in the timer and history time table. I would go back and try to see why that is, but the program runs smoothly for the most part.