Tomorrows Web - Development Document

Oliver Handford

# Concept

The concept for my Tomorrows Web project was to create an alarm that could be set on a PC, Laptop or Tablet and the alarm go off on the user’s mobile device. The thought process behind this is that a reminder could be set while the user was at their desk, without having to get their phone out and manually set the alarm.

# Specification

The specification for this application was the following:

* The user does not need to authenticate.
* The user receives a room code, while on their desktop, to ‘join’ on their mobile
* The user can input the room code on their mobile and ‘join’ the room.
* The user can set an alarm on their PC and send it to the mobile.
* The user can set an alarm as far in the distance as they like, using a datetime picker.
* The user can set a name for the alarm.
* On both devices, the user should be able to see a log of which devices have joined the room.
* The UI and frontend should be very simple and user friendly and intuitive.
* On the mobile, a countdown timer should show how long is left on the alarm.
* The phone should vibrate, if supported, and still work regardless of browser support.
* The entire project should be written in HTML, CSS, and JS.
* The project should use WebSockets.
* The project should use the Web Vibrations API.

# Research, Development and Technical Description

The first stage of this project was the decide the specification, which called for the use of WebSockets and the Web Vibrations API. I will review each one in the appropriate section.

## WebSockets

The WebSockets API was originally introduced to the web December 2009 by Google, when Chrome v4 was the first browser to support the standard (Wikipedia, 2021). With this, brought the ability to transmit real-time messages between a client and server, without having a make a separate request each time. The WebSockets API is supported by all modern browsers and is a cornerstone of the modern web. This made it a perfect choice for Tomorrows Web project.

The purpose of the WebSockets for my application is to transmit the data between the PC and the mobile in real-time. This means that no extra requests are required on the mobile client to receive the alarm data, or to join a room, where in previous systems a periodic request would have had to been sent to a server to check if the alarm data has been sent.

Due to the widespread use and implementation of this standard, no alternatives were tried. However, to simplify the setup for this project, I used SocketIO, a NodeJS package that creates a friendly wrapper around the WebSockets API (Socket.IO, 2021).

Socket.IO is used with a simple HTTP node server to display the web page, which is viewed by both the PC and mobile client, and also transmits all of the appropriate alarm data and room data between the connected clients.

As the alarm data is not sensitive information, and no authentication is required for this application, the alarm data and room data are not separated into private channels, but rather each client receives all information and checks if the currently joined room matches the received room data. In the context of this Proof of Concept, this is perfectly acceptable, however in a fully deployed solution, private channels would improve the amount of wasted traffic, wasted computation and privacy of the application.

## Vibrations API

The Vibrations API is a newer API, originally implemented in 2014, but only still sparsely adopted by browsers. This API allows an application to vibrate an attached mobile device, as if it was a native application.

The use for this in my application was to send vibrations when the alarm was sounded, to increase the user experience of the application. This was also chosen because, the Audio API, although much more widely supported has lots of limitations for playing sounds, particularly on mobile devices. Therefore, the Vibration API was chosen.

I also considered using the Notifications API, which sends a native notification, as well as a vibration. However, this was not chosen due to the more complicated setup to allow the application to send notifications when the page is not in focus and is something I would implement in the future.

## Application Basis

The application is a NodeJS project, with an Express HTTP server and a SocketIO server to run the HTTP and WS services. The Express server hosts a single HTML file, which acts as both the mobile and desktop client.

The HTML file, uses jQuery, MomentJS, CountdownJS and bootstrap-datetimepicker. The application also uses a custom theme of Bootstrap 3 for the frontend styling.

The entire client-side JavaScript is just under 100 lines of code, which handles almost all of the logic of the application.

# Critical Reflection

Overall, I feel like the proof of concept successfully demonstrates the key features of the application and logically how it would work, be implemented and its usability. However, the proof of concept clearly lacks the polish of a finished application.

The frontend design needs work to improve the user experience and the flow of the application. It also uses jQuery to hide and show elements based on the device type, which could now be done with CSS instead to speed up page loading times which is important when building a site designed for mobiles.

The server setup is a good, modern solution for this problem. However, not using private channels with WebSockets is a limiting factor of the proof of concept which could not be deployed in the finished product and would not scale well with more users.

Overall, to bring this concept to the next stage I would like to polish up the main flaws of the current system but also implement Progressive Web App (PWA) support to this site, with a manifest file and service workers. Implementing this as a PWA would mean that it could be ‘installed’ and act as a native app, and also be used offline, so a user could setup an alarm and then take their phone to an area with no connection and still receive the alarm as planned.

# References

Wikipedia (2021) *WebSocket.* Available at: <https://en.wikipedia.org/wiki/WebSocket#:~:text=In%20December%202009%2C%20Google%20Chrome,two%20revisions%20under%20Ian%20Hickson>. (Accessed: 25/05/2021)

Socket.IO (2021) *Socket.IO v4.0.* Available at: <http://socket.io> (Accessed: 05/05/2021)