SVG in JavaScript Clock

132148

Multimedia Design & Applications

Computer Science

20/02/2017

1 Introduction

This report will demonstrate the implementation of a clock created with SVG in JavaScript. This will be done by describing the functionality of the main components that the clock provides, which includes analogue and digital clock displays, date, a stopwatch and an alarm. Visual components of the clock will also be described, such as implementation of different icons/fonts.

2 Design and Development

2.1 Media Elements

All visual elements were created using Raphaeljs. Every separate circle/rectangle is a vector object. Text, including numbers, also uses Raphaeljs (text) objects. Lab tutorials were used to learn how to use the Raphaeljs library [1].

2.2 Analogue & Digital Displays

To get the clocks to display the current time, a function called startTime() [1] is consistently called using the built-in setTimeout() Window function. startTime() gets the current time (using the 'Date' JavaScript object), assigns the seconds, minutes and hours to variables, then updates the displays accordingly. The 'animate' Raphaeljs function is used on the clock hands, in which a rotate property updates the hands to the correct positions depending on the second/minute/hour variable (this has been written so that every hand updates every second). The digital display is simpler; it just uses the values of the variables to display the time.

2.3 Changing the time/Date functionality

Figure 2.3.1

To change the time of the analogue/digital clocks, the yellow buttons just above the digital clock, from left to right, will increase the hour and minute respectively. They will wrap around i.e. clicking the minute button when it is displaying 59 will set it to 00, and also increase the hour. The pink button resets the time to the default current real time. The date function, which for default displays the current day, month and year, uses the same functionality, with an additional button to be able to decrement the year:

Figure 2.3.2

Four functions are used to update the minute, hour, day and month correctly so that they are all connected (e.g. going from 23 to 00 on the hour will increment the day, going from Dec to Jan will increment the year etc.). They all use the same methodology, so one example (for updating the minute) will explain how they all work:

1) getMinute() is given the current real time as a parameter. Initially, it will get the minutes of the current time and display that.
2) When the minute is increased manually (e.g. by clicking the minute increase button), a variable minuteIncr is incremented. This is then added on to the current real time in getMinute() to display the incremented time.

3) If the minute value (i.e. real minute time + minuteIncr) reaches 60, which will only happen when the minute is increased manually, the value will reset to 00 using the '% 60' operator. A hourIncr variable will then be incremented, updating the hour appropriately.
4) An if statement checks for two scenarios:
• If the current minute updates to 00 in real time, the hour would update in real time, therefore hourIncr should not be incremented (implemented using minuteIncr != 0).
• hourIncr should only be incremented once when the minute value becomes 00 (implemented by updating the Boolean shouldIncrHour at the appropriate times).

2.4 Stopwatch

The stopwatch provides functionality of an on/off button (green when on) and a reset button. A stopwatchUpdate() function is called whenever the stopwatch is on. A stopwatchCurrent variable takes the time from when the stopwatch was turned on (again) to the current time. This value (in milliseconds) is used to work out the time elapsed, which is added on to the previously saved time if there was one (using an array of values which is saved when the stopwatch is turned off).

To display the milliseconds, the elapsed time is converted to a string then the slice() JavaScript function is used to get just the last three digits. To display the seconds and minutes, the elapsed time is divided into the appropriate value e.g. if 23,500 milliseconds have elapsed, then 23.5 seconds have elapsed. To retrieve just the integer (23) and not the decimal (as the decimal is represented by the milliseconds in the display), the JavaScript parseInt() function is used.

2.5 Alarm

The alarm provides functionality similar to the date and time functionality; the hour and minute can be incremented to set the alarm (with the yellow buttons), and another button is used to turn the alarm on and off. When the minute and the hour of the alarm time and the clock time are equal, an alert pops up, implemented using the JavaScript alert() function. This will stay until

the times are no longer equal (i.e. when another minute has passed in the clock time). This condition is consistently checked in startTime() using an if statement.

2.6 Design choices/implementation

The overall colour scheme with bright colours was chosen as a more child-friendly design, as opposed to a professional design.

The digital displays were designed to imitate LCD displays with the green text on black background. The impact font, which is one of the browser friendly font families, was most appropriate for the imitation (the Raphaeljs attr() function on text was used to choose the font family as well as size and colour). The black backgrounds were given white borders using the attr() stroke property to further provide the illusion of a digital display.

The indents around the outer face of the clock are all separate objects, which were created by starting from one position, then using the transform rotate property of Raphaeljs to display each position appropriately.

The numbers on the outer face use the comic sans font family, which is browser friendly.

The Raphaeljs attr() 'r' property is used on the clock hand rect objects to provide rounded edges, and add to the child-friendly/playful design.

To provide their functionality, the buttons use the raphaeljs click() function to perform events (which have been described in previous sections).

2.7 External sources

A function getDays() is used to get the number of days in the current month and year. This is used when the days are updated to provide the correct modulus wrap-around value [2]. Also, to get the shortened name of the month and suffix of the day (e.g. 'nd' for 2), array values are accessed in accordance with the day and month values. Two Array JavaScript objects called suffixes and months are used [3]. Lastly, a function from the lab exercise was used to add one 0 (or two for milliseconds) if the values displayed were less than 10 [1].

3 Conclusion

Overall, I think the project is successful in its aim of providing a useful clock that can display different times and utilities for different purposes correctly. However, more functionality could have been implemented such as rotation in 3D, and also more visual elements (such as bitmap images). Also, the buttons could have been made more accessible by putting them outside the clock face; however this was more of a design choice as it was made to imitate a real clock, which would have the buttons on the face.

4 References

[1] P. Newbury. Labs. *Study Direct,* 2017. [Online] Available from: http://studydirect.sussex.ac.uk/course/view.php?id=27803&topic=2 [Accessed 19/02/2017]

[2] FlySwat. What is the best way to determine the number of days in a month with javascript?. *Stack Overflow,* 2008. [Online] Available from: http://stackoverflow.com/questions/315760/what-is-the-best-way-to-determine-the-number-of-days-in-a-month-with-javascript [Accessed 19/02/2017]

[3] Jesper. Get month name from Date. *Stack Overflow,* 2009. [Online] Available from: http://stackoverflow.com/questions/1643320/get-month-name-from-date [Accessed 19/02/2017]