Fawaz Shah

System Life Cycle – “Countdown” Web App

Analysis

This web app has one main purpose: to display a countdown timer to a specific datetime event. Past this time, the countdown should end and a suitable message will be displayed (i.e. “It is now past the deadline.”).

Design

The logical flow behind the app is handled using JavaScript and JQuery. The days, hours, minutes and seconds for the countdown will be calculated and stored as variables. The actual countdown will be a string.

This is the pseudocode for calculating the countdown (endDateTime and currentDateTime are given in milliseconds since Jan 1st 1970):

```
if currentDateTime < endDateTime:

        timeDifference = endDateTime - currentDateTime
        days = floor(timeDifference ÷ (1000×60×60×24))

        remainder = timeDifference – 1000×60×60×24 × days;
        hours = floor(remainder ÷ (1000×60×60))

        remainder = timeDifference – 1000×60×60×24 × days – 1000×60×60 × hours;
        minutes = floor(remainder ÷ (1000×60))

        remainder = timeDifference – 1000×60×60×24 × days – 1000×60×60 × hours –
        1000×60 × minutes
        seconds = floor(remainder ÷ 1000)

        countdown = days + " days " + hours + " hours " + minutes +  “minutes " + seconds +
        " seconds until the deadline."
        print(countdown)

else:

        countdown = "The deadline has passed."
        print(countdown)
```

(^^^ this code loops infinitely and the page is updated every 0.5s)

The design of the app and any animations is handled with CSS3 (and HTML).

Implementation

I started by focusing on the logical flow of the countdown process. I developed the algorithm as seen above and checked it against a pre-existing countdown website. While coding, I found out early on that you couldn't mix JQuery code and JavaScript code, therefore the app was made using pure JavaScript.

The fade-in animations of the HTML contents were to be created using CSS3 transitions. One quirk I found out is that for transitions to work on Chrome and Safari, the prefix "-webkit-" must be added before each CSS property. Therefore I had to make two lots of each property, one for general browsers and one for Chrome and Safari.

Testing

Since my app would only have to work with one pre-set countdown date, I didn't test it with any others. If the app asked the user to input their own countdown date, I would have had to systematically test it with **normal data**, **erroneous data** and also **boundary data,** e.g. 29th February (happens only every 4 years) or any date that involves accounting for daylight saving (± 1 hour). However, since the app uses JavaScript's built-in date/time functionality, the app should take these things into account already.

Evaluation

This app fulfils all its necessary requirements, and I succeeded at adding in everything that I intended to at the start (such as animations). Ideally I would have allowed the user to enter their own countdown date, although this would have required server-side processing, which at the moment is unavailable to me. I would have performed extra testing if the countdown date/time were to be changed by the user.