Design Document

Thomas Moreno-Cooper

Coursework 2

University of Leeds

COMP1021 - Introduction to web technologies

Sc17tm@leeds.ac.uk

Introduction

Throughout this design document, it will be assumed that the reader is familiar to a slight degree with the content of the website when it was submitted as part of Coursework 1. If not, it broadly recounts my move from France to Leeds to study Computer Science. More detail will be given when appropriate when explaining the new features of the website. This document contains the design and explanation of the features as well as details on any changes made during implementation.

Additional features and design

The core purpose of the website being already explicit, the focus of these features is to add a certain degree of functionality to the website, making it slightly more interesting and personal.

Three core functionalities have been added, all using JavaScript to varying extents :

- A slideshow with pictures of my move from Grayson Heights to the Tannery
- A greeting message that changes depending on the local time of the user
- A map that shows exactly where I used to live in France.

Before the implementation of the slideshow, the page dedicated to my move from Grayson Heights to the Tannery contained two pictures stacked on top of each other. This is detrimental to the readability of the content and clunky to look at. It is necessary to design and implement some sort of feature that is both more elegant and doesn't add to much weight to the page.

The second feature of the website is a little greeting message on the welcome page. This may seem trivial at first, but it really adds a little charm to the website. The design is relatively simple since the methods in JavaScript already exist and the style format used will be the same as all the other paragraphs of the website to preserve a sense of continuity.

Lastly, the map will be a visual guide to show exactly where I live in the south of France as some people might not know where the Côte Bleue is or even what Marseille and Provence are. The layout will be relatively simple as it will follow the margins of the content part of the website and will be put in between two paragraphs.

A lot of html and JavaScript tutorials are available online, especially to implement certain features such as the photo slideshow. Hence the design process was streamlined to a certain degree. The real reflection and actual difficulty was around how exactly to implement them and understanding how the underlying algorithms of these features worked.

Implementation and evaluation

The implementation of a slideshow is a perfect answer to the old and awkward "stacked" presentation. Essentially, a div element is implemented for the slideshow "frame", containing

child div elements for the slides which each contain a single picture. The frame will in effect only display one picture, at a time, hence when one picture is loaded, the others must be “invisible”. We must set the CSS display property of the loaded picture to “display : block ;” i.e. display the picture, and the remainders to “display : none ;” i.e. the others are invisible. The JavaScript comes into play when handling the skipping from one picture to the next. All the pictures’ display properties are set to “display : none ;” with a for loop, a blank canvas is created and the desired picture is displayed by simply reverting its display property to “display : block;”. To navigate through the pictures, two simple buttons called prev and next are set to handle the onclick events “display next picture” or “display previous picture”.

Because the pictures are stored in an array, the JavaScript function checks if previous or next are beyond the boundaries of the array and reverts to the opposite end if necessary. For instance, if the current displayed picture was at the last element of the array and the browser clicks on next, the function will display the first picture of the array.

```
<script>
  var indexCount = 1;
  displaySlide(indexCount);

  function displaySlide (k) {
    var slides = document.getElementsByClassName("slide");
    if (k > slides.length) {
      indexCount = 1;
    }
    if (k < 1) {
       indexCount = slides.length;
    }
    for (var count = 0; count < slides.length; count++) {
      slides[count].style.display = "none";
    }
    slides[indexCount-1].style.display = "block";
  }

  function prevOrNextSlide(k) {
    displaySlide(indexCount += k);
  }
</script>
```

For the second feature, the browser’s local time is obtained with the Date object, we take the hours part with the inbuilt method and use simple Boolean logic to determine what time of the day it is.

```
<script>

  function greet() {
    var userDate = new Date();
    var x = userDate.getHours();
    if ((0 <= x) && (x < 12)) {
      document.getElementById("greetings").innerHTML = "Good Morning!";
    }
    else if ((12 <= x) &&  (x < 19)) {
      document.getElementById("greetings").innerHTML = "Good Afternoon!";
    }
    else {
      document.getElementById("greetings").innerHTML = "Good Evening!";
    }
  }

  greet();

</script>
```

Lastly, the little map will use the very helpful google maps API. Essentially, a little JavaScript creates the properties of the map such as what coordinates the map is centred around and the zoom, as well as any possible map markers through special methods recognised by the API.

The API is then summoned by a link and fed the properties via this link and displays the desired map. Quite frankly, the real difficulty of this feature's implementation was how to create the google API key, necessary to authenticate oneself to the google servers when summoning an API, as well as enabling the google maps API on the key.

```
<script>
  function myMap() {
    var myCenter = new google.maps.LatLng(43.353880, 5.197097);
    var mapProperties= {
    center: myCenter,
    zoom: 11,
    };
    var map = new google.maps.Map(document.getElementById("Map"),mapProperties);
    var marker = new google.maps.Marker({position:myCenter});
    marker.setMap(map);
  }
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCkkwzRCq4xzD0WMUpC
```

References

(1) Google (2018) *Google maps API documentation* Available from: https://developers.google.com/maps/documentation/ [Accessed 10 may]

(2) W3schools (2018) *html* Available from: https://www.w3schools.com/ [Accessed 10 may]