

yr2 sem1 webdev mobile assignment 2

by Jericho Jermaine Uzzell

FILE	12014_JERICO_JERMAINE_UZZELL_YR2_SEM1_WEBDEV_MOBILE_AS SIGNMENT_2_308223_1794148637.DOCX (248.25K)		
TIME SUBMITTED	20-JAN-2021 07:52PM (UTC+0000)	WORD COUNT	1180
SUBMISSION ID	142862516	CHARACTER COUNT	10056

University of Wales Trinity Saint David

School of Computing

Web and Mobile Technologies

ACCA5031

Jericho Uzzell

1702201

12/01/2021



Prifysgol Cymru

Y Drindod Dewi Sant

University of Wales

Trinity Saint David

1. Introduction; [page 4]
2. The site; [page 3-11]
 - 2.1. Spotify stats; [page 3-6]
 - 2.1.1. Screen shots [page 3];
 - 2.1.2. Methodology [page 3];
 - 2.1.3. Annotated code [page 5-6];
 - 2.2. A random joke; [page 7]
 - 2.2.1. Screen shots;
 - 2.2.2. Methodology;
 - 2.2.3. Annotated code;
 - 2.3. Most popular song; [page 8-9]
 - 2.3.1. Screen shots [page 8];
 - 2.3.2. Methodology [page 8];
 - 2.3.3. Annotated code [page 9];
 - 2.4. U.K Population; [page 10-11]
 - 2.4.1. Screen shots [page 10];
 - 2.4.2. Methodology [page 10];
 - 2.4.3. Annotated code [page 10-11];
- 2.5. Conclusion; [page 12]
3. Bibliography; [page 12]

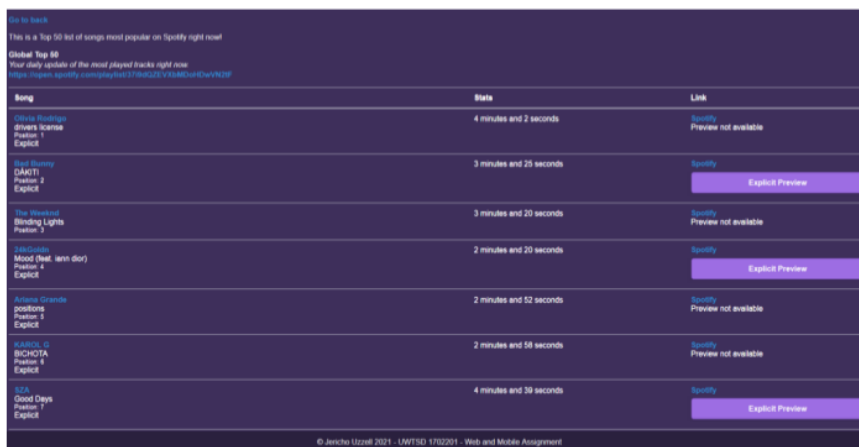
1. Introduction

This document will review a website that has been developed by the student. The website must conform to some requirements, those being: to have a minimum of three pages, demonstrate mobile features such as single page application, transitions, panels, colapsables and more, site style css edits, perform AJAX request on public data, perform AJAX request on site data; the student appended a mobile version of the site they created previously. With the mobile version having different features and does not include previous features.

2. The site

2.1 Spotify stats

2.1.1. Screen Shots



Song	Date	Link
Circles Post Malone Position: 1 Explicit	4 minutes and 2 seconds	Spotify Preview not available
Savage Megan Thee Stallion Position: 2 Explicit	3 minutes and 25 seconds	Spotify Explicit Preview
The Rock Show The Rock Position: 3 Explicit	3 minutes and 20 seconds	Spotify Preview not available
Mood (feat. 21 Savage) Drake Position: 4 Explicit	2 minutes and 20 seconds	Spotify Explicit Preview
Positions Ariana Grande Position: 5 Explicit	2 minutes and 52 seconds	Spotify Preview not available
ROCKSTAR Cardi B Position: 6 Explicit	2 minutes and 58 seconds	Spotify Preview not available
Good Days JAY-Z Position: 7 Explicit	4 minutes and 30 seconds	Spotify Explicit Preview

2.1.1. Methodology

The method used for this page was to query Spotify using the students API key they generated so that information from a public playlist could be displayed. The request calls a PHP server file which then calls a file on Spotify's server -- the returned data is then processed and is displayed in a table like format.

It was possible to have the API key on the client end (in the JavaScript) however was opted against for security reasons. This table gives information on the playlist with a link to it, information on the song, the artist, its position, duration and links for the artist and song.

This page also has a preview button, this will be discussed later in the previous page.

2.1.2. Annotated code

```
let audio=null;
function getSpotify() {

$.post('/allqueries.php',{type:'getSpotifyData'},function(data){
    console.log(data);
    if(data.substring(0,1)!='{'){swal({title: "Failed to load
script!",text: "Check the console of the website for more info.",type:
"warning"});return;}

    data=JSON.parse(data);

    if(!('result' in data)){swal({title: "Error, result not
found.",text: data['message'],type: "error"});return;}
    if(data['result']==0){$('#spotify_table
tbody').html('<tr><td colspan="3">'+data+'</td></tr>');return;}

    if(!('data' in data)){swal({title: "Error, data not
found.",text: data['message'],type: "error"});return;}

    if(!$$.isPlainObject(data['data'])&&$.isArray(data['data'])){swal({title:
"Error, data not array.",text: data['message'],type: "error"});return;}
    let out='';

    $('#spotify_track_name').text(data.data.name);
    $('#spotify_track_desc').text(data.data.description);

    $('#spotify_track_url').text(data.data.external_urls.spotify);

    $('#spotify_track_url').attr('href',data.data.external_urls.spotify);

    console.log(data.data.tracks.items);
    //preview not available for mobiles...they do not allow
audio to play :( ;
    $.each(data.data.tracks.items,function(k,i){
        out=out+'<tr>';
        out=out+'<td><b><a target="_blank"
href="'+i.track.artists[0].external_urls.spotify+'">'+i.track.artists[0].name+
'</a></b><br/>'+i.track.name+'<br/><small>Position:
'+(k+1)+'</small>'+(i.track.explicit?'<br/>Explicit':'')+'</td>';
```

```
out=out+'<td>'+milisecondToHours(i.track.duration_ms)+'</td>';
        out=out+'<td><a target="_blank"
href="'+i.track.external_urls.spotify+'>Spotify</a>'+(!isMobile())&i.track.pr
eview_url!=null?'<br/><button class="btn btn-block btn-sm
'+(i.track.explicit?'btn-warning':'btn-info')+' mt-2 spotify_preview"
songname="'+i.track.name+" songurl="'+i.track.external_urls.spotify+"
artistname="'+i.track.artists[0].name+"
artisturl="'+i.track.artists[0].external_urls.spotify+" key="'+i.track.id+"
preview_url="'+i.track.preview_url+">'+(i.track.explicit?'Explicit
':'')+'Preview</button>':(i.track.preview_url==null?'<br/>Preview not
available':''))+'</td>';
        out=out+'</tr>';
    });
    $('#spotify_table tbody').html(out);
});
}
function milisecondToHours(duration) {
    var seconds = Math.floor((duration / 1000) % 60),
        minutes = Math.floor((duration / (1000 * 60)) % 60),
        hours = Math.floor((duration / (1000 * 60 * 60)) % 24);

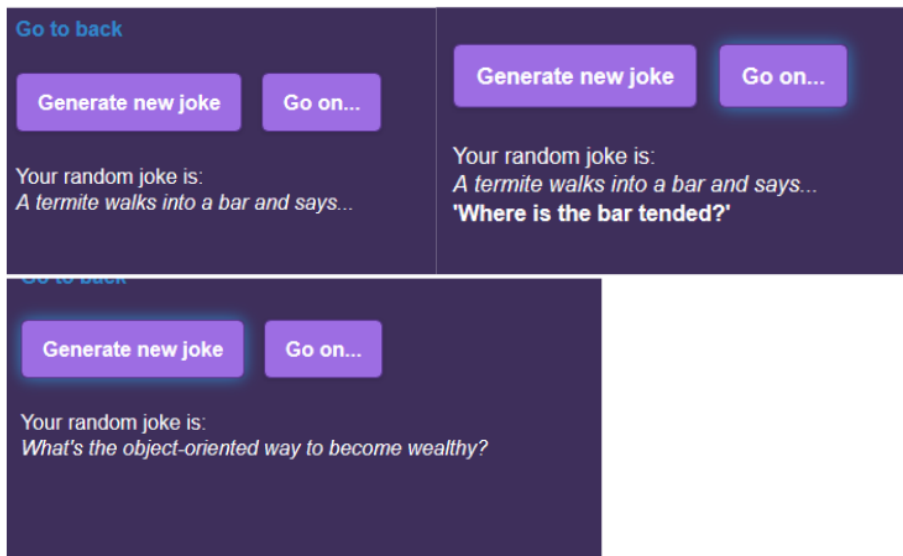
    hours = (hours < 10) ? hours : hours;
    minutes = (minutes < 10) ? minutes : minutes;
    seconds = (seconds < 10) ? seconds : seconds;

    return (hours==0?'':' '+hours) + minutes + " minutes and " +
seconds+" seconds";
}
$(document).on('click', '.spotify_preview',function(){
    $('.spotify_preview').attr('disabled',true);
    let sound=$(this).attr('preview_url');
    if(audio!=null && audio.duration > 0 && !audio.paused)
{audio.pause();}
    audio = new Audio(sound);
    audio.play();
    this.blur();
    addPreview($(this));
    setTimeout(function(){
        $('.spotify_preview').removeAttr('disabled');
    }, 2000);
});
```

```
    });  
    function addPreview(event) {  
  
$.post('/allqueries.php',{type:'addPreviewSpotify',id:event.attr('key'),songname:event.attr('songname'),songurl:event.attr('songurl'),artistname:event.attr('artistname'),artisturl:event.attr('artisturl')},function(data){  
    console.log(data);  
    if(data.substring(0,1)!='{'){swal({title: "Failed to load script!",text: "Check the console of the website for more info.",type: "warning"});return;}  
  
        let prevname=event.text();  
        event.text('Added to preview tally');  
        setTimeout(function(){  
            event.text(prevname);  
        },2000);  
        getPreview();  
    });  
}  
function isMobile() {  
    if( /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera Mini/i.test(navigator.userAgent) ) {return true;}  
    return false;  
}
```

3.1 A random joke

3.1.1. Screen Shots



3.1.1. Methodology

This script automatically runs when the page loads. When it loads by default, it calls a function which then sends an AJAX request off to the website -- this site does not require any API key so a direct link within the JavaScript is fine.

Clicking the generate new joke button does the exact same function as it does on page load. As an added bonus, if you wanted to try and guess the joke, the punchline is hidden which means you must click the go on button to show it.

3.1.2. Annotated code

```
function generateJoke() {  
    $.get('https://official-joke-  
api.appspot.com/random_joke',{},function(data){  
        // console.log(data);  
        // data=JSON.parse(data);  
        $('#page_randomjoke_text').text(data.setup);  
  
        $('#page_randomjoke_text_punchline').hide().text(data.punchline);  
    });  
}  
$(document).on('click','#page_randomjoke_generate',function(){  
    generateJoke();  
});
```

4.1 Most popular song

4.1.1. Screen Shots

The first screenshot shows a table with two columns: 'Song' and 'Previews'. The table lists the following songs and their preview counts:

Song	Previews
Bad Bunny DÁKITI	8 previews
Tiësto The Business	4 previews
Boza Hecha Pa' Mi	4 previews
24kGoldn Mood (feat. Justin Bieber)	3 previews
Tate McRae You Know I'm Not	3 previews
Don Lipa Cumbiando (feat. Jhay Cortez)	3 previews
Bad Bunny La Bachata (feat. Jhay Cortez)	2 previews

The second screenshot shows a modal window titled 'This pages information!'. It contains the following text:

This page gathers the information of the Preview Button on the [Spotify](#) page.

Every click to preview the song will add a 'listen count' to the statistic. These songs, although from the Top 50 current songs, demonstrates the intestes in them.

At the bottom of the modal, there is a 'Close information' button. To the right of the modal, a partial view of the song list is visible, showing 'Bad Bunny DÁKITI', 'Tiësto The Business', 'Boza Hecha Pa' Mi', and '24kGoldn'.

4.1.1. Methodology

The preview page relies entirely on the Spotify playlist page. This function works off the amount of times a user has clicked on the preview button for a certain song. The count is incremented by one and if the file does not exist or the preview song is not in the list, the script automatically adds it.

The preview button on the Spotify by sends an AJAX request to a PHP server file on the local website which does all of the counting.

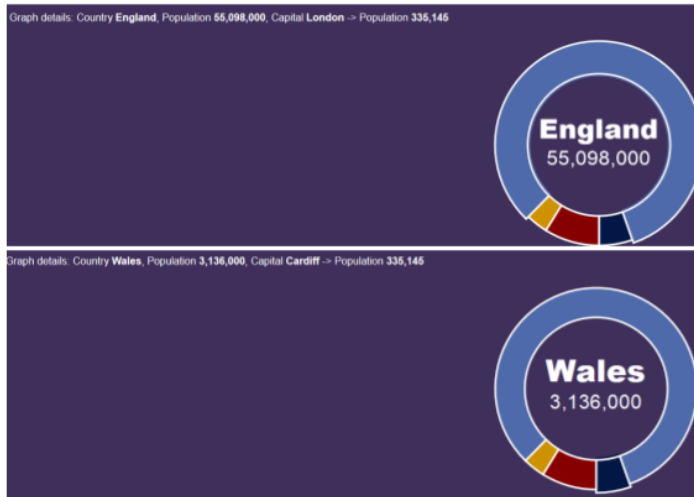
The preview page sends an AJAX request to a locally stored file on the local website requesting for its contents, where it will show in a table the song, artist and preview times. A text file was chosen because of the simplicity of counting and creating elements however, in a real world environment, a database would be ideal for tracking who clicked what, database relationships and ease-of-use.

4.1.2. Annotated code

```
function getPreview() {  
  
$.get('/resource/previewspotify.txt?q='+Math.random(0,1),{},function(data){  
    console.log(data);  
    if(data==' ' || data=='[]') {return false;}  
    data=JSON.parse(data);  
    let popular=new Array();  
  
    data.sort(function (a, b) {  
        return b.count - a.count;//desc ... for asc use ->  
return a.count - b.count  
    });  
  
    // console.log(data);  
    let out='';  
    $.each(data,function(k,i){  
        out=out+'<tr>';  
        out=out+'<td>';  
        out=out+'<b><a href="'+i.artisturl+'"  
target="_blank">'+i.artistname+'</a></b>';  
        out=out+'<br/><small><i><a  
href="'+i.songurl+'"' target="_blank">'+i.songname+'</a></i></small>';  
        out=out+'</td>';  
        out=out+'<td>'+i.count+'  
preview'+(i.count==1?'':'s')+'</td>';  
        out=out+'</tr>';  
    });  
    $('#mpps_table tbody').html(out);  
    });  
}
```

5.1 U.K population

5.1.1. Screen Shots



5.1.1. Methodology

This AJAX request pulls information about the country population. This AJAX request has a multidimensional array within it which allows for a country and its capital city to be given to the script. This script places that information into a pie-chart with an added ability to view the information when you however over a section.

5.1.2. Annotated code

```
function getPopulation() {
    $('#population_graph_details').hide();

$.get('/resource/ukcountrypopulation.txt?q=' + Math.random(0,1), {}, function(data)
){
    console.log(data);
    if(data==' ' || data=='[]') {return false;}
    data=JSON.parse(data);

    let graph=new Array();
    $.each(data,function(k,i){
        // console.log(i.population)

graph.push({value:i.population,label:i.name,cname:i.capital.name,cpopulation:i
.capital.population})
    });

    document.getElementById('population_graph').innerHTML='';
    let mDonut=new Morris.Donut({
        element: 'population_graph',
```

```
        data: graph,
        parseTime:false,
        pointSize: 2,
        hideHover: 'auto',
        behaveLikeLine: true,
        xkey: 'data',
        labelColor: "#fff",
        colors: ['#031746', '#4f6aaa', '#ce9100',
'#860000','#323232'],
        ykeys: ['value'],
        labels: ['Value']
    });
    for(i = 0; i < mDonut.segments.length; i++) {
        mDonut.segments[i].handlers['hover'].push(
function(k,i){
            // console.log(graph[k]);
            $('#population_graph_details').show();

$('#population_graph_country').text(graph[k].label);

$('#population_graph_population').text(numberWithCommas(graph[k].value));

$('#population_graph_capital').text(graph[k].cname);

$('#population_graph_capitalpopulation').text(numberWithCommas(graph[k].cpopulation));

        });
    }
});
}
$(document).on('click','.ui-link',function() {
    if($(this).attr('href')== '#page_population')
{getPopulation();}
});
function numberWithCommas(x) {
    return x.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ",");
}
```

6. Conclusion

Concluding this report, two features required server sided code, which is not a bad thing. There was a lot of JQuery instead of pure JS however, JQuery allows for faster development and it is easier (to the student at least) to read. The Spotify API is good but linked in with the preview button, showing how many previews it has had on this website is even better. The student offered two options for each requirement, for the public AJAX they done Spotify and a random joke. For the local AJAX they done a preview button calculator and the U.K population. Their style appeared bland, it was not bright which could be bad due to people preferring a dark option.

7. Bibliography

<https://stackoverflow.com/questions/2901102/how-to-print-a-number-with-commas-as-thousands-separators-in-javascript>
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort
<https://stackoverflow.com/questions/3514784/what-is-the-best-way-to-detect-a-mobile-device>
<https://stackoverflow.com/questions/8489710/play-an-audio-file-using-jquery-when-a-button-is-clicked>
<https://stackoverflow.com/questions/19700283/how-to-convert-time-milliseconds-to-hours-min-sec-format-in-javascript>

yr2 sem1 webdev mobile assignment 2

GRADEMARK REPORT

FINAL GRADE

78/100

GENERAL COMMENTS

Instructor

WEBSITE

Range of code demonstrated - I don't think you necessary needed to use curl, but nevertheless Spotify API call is working well and is integrated well with the website. Good management of the sound files. Joke of the day is quite straightforward but working fine. I enjoyed the jokes in your demo video(!) Use of the text files, particularly the preview one is quite complex. It might have been a better approach to write the ids to a database and then increment the previews (and use a `json_encode()` from PHP to read it back - making a dynamic public JSON application) Overall, some very good work here. Several jQuery components have been used, although perhaps this could have been a bit more extensive. 48.5/60

Overall integration of website - Main page is a bit minimalist - I would have liked a bit more of a theme to the website rather than a demo of random components! The Spotify components are particularly well implemented however. 8.5/10

REPORT

Analysis of technical development - A good account of what you did both practically and technically. You need more of an analytical analysis - was this the best approach, would you do it differently next time etc 12/15

Annotated code listing - It's a bit hard to follow your code due to the tabbing. Comments (annotations) are a bit on the light side. 8.5/15

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12
