Lab: Responsive Webpage

Lab Overview

In this lab you were hired to code a responsive webpage for the a music streaming company called *AudioFire*. Their website allows users to choose music and read reviews / music news. This lab will cover topics including: HTML, CSS layout and converting a webpage into a responsive design that will work on all major device platforms.

Take a moment to look over the *mockup.pdf / mockup.fig* for the webpage. You'll see that the design is a mix rows and columns. Some areas have two, three or even five columns.

You'll notice that for medium devices like tablets that there are less columns. You'll also notice that a menu icon appears at the upper right hand corner. This icon is meant to be interactive where users can click on with their finger to open up the navigation bar.

You'll also notice that for small devices like smart phones that the design changes again where everything has become a single column, there is also an additional message to download the official app.

Creating a complex responsive webpage like this is not easy, so we'll do it in steps. We will follow the general steps in the order below.

HTML Layout
CSS Text / CSS Backgrounds
CSS Layout
CSS Colors / Backgrounds
CSS Padding / Margins / Display
CSS Odds and Ends
CSS Responsive

Begin Coding

Start by creating a new HTML document. Add in all the standard HTML basic page tags including: *<!doctype html>*, *<head>*,

<title>, and *<body>*.

Save it into the save folder that stores all of lab files you'll need for this assignment. Save it as *music.html*.

Header Layout

Begin by laying out the *<header>*, *<main>* and *<footer>*. These are our main areas of our webpage.

You can see from the design that in the *<header>* section it is divided into three columns, one for the company logo, one for the navigation bar and one for the social media.

```
<div>
    <nav>
        <a href="">Stream</a>
        <a href="#">Albums</a>
        <a href="#">Features</a>
        <a href="#">Podcast</a>
        <a href="#">Features</a>
    </nav>
</div>
```

Inside the *<header>* create three divs. In the first div put the *logo.png* file. Add a class of company-logo to the div.

In the second div add a *<nav>* container and add five hyperlinks, use the mockup and *Website-Text* document for assistance. Add a class of nav to the parent div.

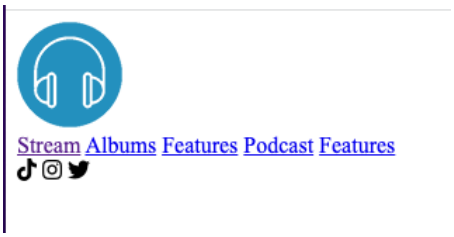
```
<div class="social-media">
    <i class="fa-brands fa-tiktok"></i>
    <i class="fa-brands fa-instagram"></i
    <i class="fa-brands fa-twitter"></i>
</div>
```

Font Awesome

In this lab we will use a lot icons. It will help to use the Font Awesome CDN to help us quickly add in icons.

Go to the Font Awesome website, log into your account and use the font awesome CDN script and add it to the *<head>* section of your HTML document.

Now that Font Awesome is installed find and copy and paste the code for *Tik Tok*, *Instagram* and *Twitter* icons into the last div in the header. Add a class of *.social-media* to it.



Save and your webpage should look roughly like image on the right.

Main

Now that our header is complete, let's add the main content. In the *<main>* tag add a *<section>* tag and add a class of stream.

Inside the *<section>* add an *<h2>* and add *"Stream Now"* to it.

```
<div>
    <figure>
        <img src="elvis.jpg">
        <figcaption>
            <h3>Elvis</h3>
            <p class="album-name">Elvis Presley</p>
            <p class="album-year">1956</p>
        </figcaption>
    </figure>
</div>
```

Next let's add the albums. Create a * * and call it .albums. Inside the *.albums* div add a new * *. This will hold the album. Because the album has both an image and text related to it let's use a *<figure>* / *<figcaption>*. Put the *<figure>* inside the * *. Add the *elvis.jpg* image to the *<figure>*. Add a *<figcaption>* within the *<figure>*. Add a paragraph for the album title and a paragraph for the year the album was released. Give both paragraph descriptive class names such as *.album-title* and *.album-year* so that you can select the text later to style with CSS if you like. Put the *Elvis Presley* album and year into the correct locations.

Now that we have one album, copy and paste the * * four more times within the *.albums* div. Replace each *<figure>* with the correct album art, album title and year. Use the mockup / website document as a reference.

Save and your webpage should look roughly like the image on the right.

Reviews

Now that we have the Albums section let's create a new a section to place the review content.

Create a new *<section>* tag and add a class of .reviews. Inside the *<section>* add an *<h2>* and add *"Album Reviews"* to it.

Next let's add the review articles. Create a * * and call it *.review-articles*. Inside the *.review-articles* div add an *<article>* tag. Add the iframe code for the Jessie Ware video and place it inside the *<article>* tag.

Add an h3 and add Jessie Ware.

Finally, use the *Website-Text* document to add the article information to to the *<article>* tag.

Create a new *<article>* tag and repeat the same process for *Beach House*. Use the mockup and *Website-Text* document

```
ion class="reviews">
h2>Album Reviews</h2>
iv class="review-articles">
  <article>
      <iframe width="560" height="315" src="https://www.youtube.com/embed/YAaUn5sn1T
          title="YouTube video player" frameborder="0"
          allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscop
          picture-in-picture; web-share"
          allowfullscreen></iframe>
      <h2>Jessie Ware</h2>
      <p>Jessie Ware hinted at leaving the dancefloor behind after 2020's What's You
      Pleasure, but on
          "Pearls," the single announcing her upcoming album, That! Feels Good!, the
          singer has her
          feet still planted. Assisted by the likes of Clarence Coffee Jr. (Dua Lipa
          Beyoncé) and
          Stuart Price (Kylie Minogue, Madonna), "Pearls" is ascendant and evangelis
          disco that
          feels fuller than the smoky club tracks of her last album. Gliding over
          springy keys and
          cheery percussion, Ware trades her sinuous purrs for jubilant cries that
          showcase her
          dazzling upper range. The harmonies plume like steam, the production is de
          and
          fortissimo, and Ware sets the intention to chase a little danger. "Offfff,
          let me go! Let me
          dance!" she belts, as if pushing through a scrum of bouncers, never soundi
          so warm-blooded
          and rapturous.</p>
```

as a reference. When you are done your webpage should look like the image on the right.

```
<footer>
    <div>
        <a href="#">About Audio Fire</a>
        <a href="#">Join Us</a>
        <a href="#">Press</a>
    </div>
    <div>
        <a href="#">Investor</a>
        <a href="#">Terms</a>
        <a href="#">Help</a>
    </div>
    <div>
        <div class="footer-social-media">
            <i class="fa-brands fa-tiktok"></i>
            <i class="fa-brands fa-instagram"></i>
            <i class="fa-brands fa-twitter"></i>
        </div>

    </div>
</footer>
```

Footer

Inside the *<footer>* section add three s. Inside the first * * add the first three hyperlinks. In the next add the remaining three hyperlinks. Use the mockup and Website-Text document as a reference. In the last * * repeat the social-media code from the header. Give the footer social media a class of *.footer-social-media* so that you can easily select the two navigation bars easily.

Save and your webpage should look roughly like the image on the right.

CSS

Now that the HTML structure has been set, let's lay out the base line CSS for the webpage. Let's start with creating the layout for the desktop version of the webpage.

Create a new document and label it *style.css*. Use the *<link>* tag to attach the CSS document to the music.html file.

Layout

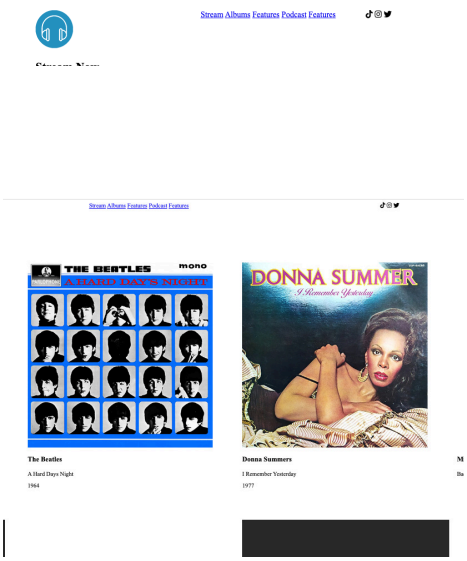
Let's begin with the layout. If you view the mockup you can see the header section is divided into three columns. Let's set that using CSS grids.

Create a CSS rule for the header tag. Set it to three columns using grid-template-columns.

```
header {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
}
```

Save and preview.

Add the remaining columns for the layout. Create an equal five column layout for .albums. An equal two column layout for



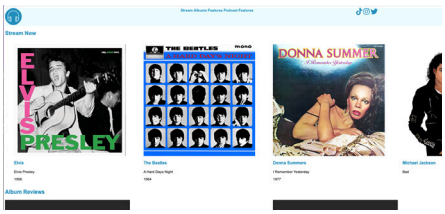
.review-articles. Use the gap property to leave a bit of space between the articles. Finally add a three column layout for the footer. Although the footer has three columns set the columns so that column 1 and column 2 are 25% of the width and column 3 is the remaining 50%.

Save and the design should look roughly like the image on the right.

Fonts / Colors / Background Colors

Use the *mockup.fig* file to assist with layout out the typography, font colors and background colors.

Use CSS inheritance to layout out the Arial font to the entire webpage.

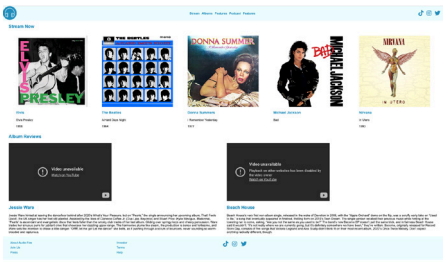


Use font-sizes in the mockup to assist with the layout.

Set the h2s, h3, navigation links and Font Awesome icons to the dark blue.

Set the header and footer background color to the light blue.

Make hyperlinks bold.



Margins / Paddings / Displays

To get a get a tighter webpage fit, remove padding and margin from the *<body>* tag.

Use *margin-right* to add space between social media icons as well as navigational links.

Add padding to *<header>* and *<footer>*

Make the navigational links move vertically by changing them in *display: block*. Separate them using *margin-bottom*.

Final Touches

To have the navigation bar and the social-media be in the vertical center turn *.nav* and *.social-media* into flexboxes. Then use align-items: center to place them in the vertical centers.

Whew! Good work, when you are done you should have a sophisticated layout with rows and columns.

Reponsive

Now that we have our design let's make sure that the design is responsive so it can work on all major platforms, including desktop computer, tablets and smart phones.

Responsive Measurements

Make sure all of your type related CSS properties such as font-size, line-height are set to a responsive measurement such as rem.

Make sure all widths are set to a responsive unit such as rem, fr or %.

Make sure all paddings and margins are set to responsive measurements such as rems.

Responsive Media

Preview *music.html* into a browser. Resize the width of the browser and you'll see that the media elements such as images and videos do not resize. The lack of resizing hurts the responsiveness of the webpage.

To fix this issue set all media such as img, video, and iframe (YouTube videos) to *max-width: 100%*.

Media Queries

Media Queries allow developers to set our webpage to use different CSS in different settings. In this case the property we will use is the width of the browser, since that will indicate what device the user is most likey using.

In the *mockup.pdf* take a look at tablet layout of the website. You'll notice that the website is laid out differently than it was for the desktop view. For instance, instead of five albums being laid out per row, it is set to three albums.

Remember that designs do not work at all sizes, you must think about what is a good design for a user at that size device.

Create a new css document and call it responsive.css. Use *<link>* to attach it to *music.html* AFTER the style.css link. This document will hold all of your responsive css.

To create media queries we need to know the width dimensions we want to affect. In this lab we will say that medium sized devices like tablets fall between 480px and 955px.

Write your media query as such:

@media screen and (min-width: 480px) and (max-width: 955px){}

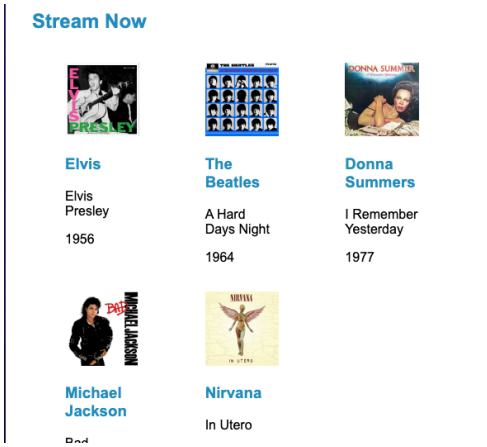
Reset the Layout

Let's start with the layout of the albums. Currently *.albums* is set to five columns. Inside the new media query set *.albums* to three columns as such

.albums {
* grid-template-columns: 1fr 1fr 1fr;*
}

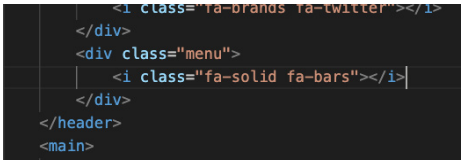
Save and test in a browser, you should see that when you browser gets smaller that the layout changes to three columns.

Change the column layout for *.review-articles* to a single column and change the footer to have two columns instead fo three.



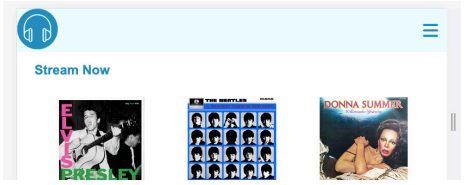
Navigation Bar

You'll notice that in tablet view that navigation bar and social media menu is replaced by a menu icon. In most smaller websites the menu icon is trigger that causes the navigation bar to appear. That trigger requires Javascript which we are not going to discuss in this course, but we can at least have the correct setup.



In the HTML go to the *<header>.* Add a four column div after the social-media. Call the div *.menu*. Go to Font Awesome and copy and paste the HTML code for the bars, or the menu icon.

In your media query change the header to go from a three column layout to a two column layout.



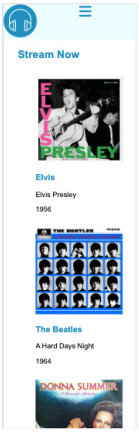
Save and preview and you'll see that the navigation bar gets a bit akward. We can fix this using display. Inside the medium device media query set *.nav* and *.social-media* to display: none for this media query. This will hide the navigation bar and social media for this size.

Small Devices

Now that we've set the medium / tablet devices, let's create a media query for small devices such as smartphones. Set it that the maximum width or max-width is 480px.

@media screen and (max-width: 480px){}

Set the layout of the *<header>*, *.albums*, *.review-articles* and *<footer>* to a single column.



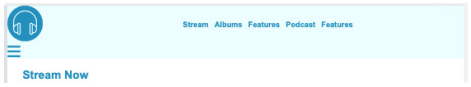
Use *display: none* to hide the header navigation nar and social media as we did for medium devices.

Set the section titles to be aligned to the center for small devices.

Set the footer items to be aligned to the center for small devices.

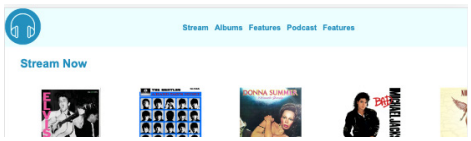
Large Devices

Finally, let's set a media query for large devices. It should be a minimum of 955px.



@media screen and (min-width: 955px){}

For large devices make sure your layout is working correctly where the header has three columns, .albums is 5 columns, *.review-articles* is two columns and the footer is three columns.



For large devices we do not need the menu icon so use *display: none* to hide it.

Submission

Whew! That was a lot of work, but the webpage looks great! Preview the webpage at different sizes and see that the design responds and changes to adapt!

Rename the root folder with your name on it, zip the folder and submit it to Canvas for grading.