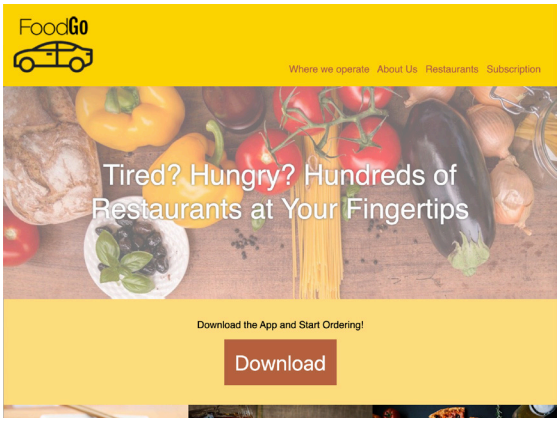
Create a Responsive Design

So far in the class we've discussed creating a sophisticated layout using HTML and CSS. However, we never really stopped to think about what size our design should be. We live in a world where people can access a website from many different type of devices. Where it's a desktop computer with a large monitor or a smart tv, a medium sized device like a laptop or tablet or a small device like a smart phone. Each device has different dimensions, yet they are all accessing the same webpage. In order for our webpage to work we'll need to make our webpage responsive. In this lab we are working with a company called FoodGo, a food delivery service. We already have the webpage set for a traditional browser, but we want to make sure it works on a mobile device like an Android or iPhone.

Overview



Take a moment to look at the *FoodGo-Preview.pdf* you can see the website as it will look as a traditional webpage on a desktop browser. We can also see how the same webpage will look on smaller devices like smartphones. You can see that the layout changes a bit. For the mobile devices multiple columns become single columns. The navigation buttons also increase in size for the mobile device. You can get a better sense of the layout by looking the *FoodGo-Wireframe.pdf*.

Take a moment to look at the HTML for the webpage under *foodgo.html*. You can see that page is broken into a *<header>*, *<main>* and *<footer>*. The header has two majors sections, the main has three major sections and the footer has two major sections.

```
<body>
    <header>
        <section id="logo">
            <img src="FoodGo-Logo.png">
        </section>
        <section id="nav">
            <a href="#">Where we operate</a>
            <a href="#">About Us</a>
            <a href="#">Restaurants</a>
            <a href="#">Subscription</a>
        </section>
    </header>
    <main>
        <section id="splash">
            <span class="splashtext"> Tired? Hungry?
            Hundreds of <br>Restaurants at Your
            Fingertips </span>
```

Open style.css and you can see how the CSS is broken into major styling sections including: typography, background, box-model and layout. Notice that the header, main and footer are using CSS grids to create columns.

```
/* == Typography == */

body {
    font-family: 'Helvetica', 'Arial', sans-serif;
    font-size: 14pt;
    line-height: 18pt;
}

h2 {
    font-size: 30pt;
}

header,
footer {
    color: #b56441;
}
```

Also note that flex box is being used to place the *#splash* and *#app* text in the vertical center of their sections. It is also being used to place the navigation bar text at the bottom of the *#nav* section.

Responsive Units

One of the issues of going between devices is that each device has it's own height and width. That makes using finite measurements like pixels and points harder to transfer from one device to another.

For a desktop a width of 600px may be too short, for a mobile device a width of 600px may be too large. The same goes for measuring types using points. 60pts may work great for a large browser on a desktop, but may be too large for a smaller device like a smartphone. Thus, we are going to need to make our units more fluid.

Fluid Typography

Not all type is created equal. Users like to control the type sizes for their devices, this is usually done their brower or device settings. However if we set type to an exact amount such as 14pts or 20pt, that takes that control away from the users. They are not able to resize their type as it is set in set instead by the CSS. To fix this issue we can use a fluid or flexible unit such as using rems to measure type. REM stands for a responsive em. An em is a relative measurement based off the default size of type for a browser or device. 1 em is equal to the default font size of a device. Every device has it's own default type size. If the default font size for a browser is 14 points, than 1em would equal 14 points. If a browser's default type size was 16 points than 1em would equal 16points.

We can convert points to rems by using this formula

$$\frac{\text{Target}}{\text{Context}} = \text{REM}$$

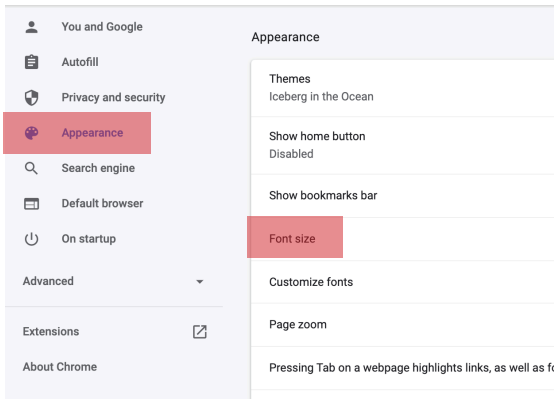
The ***'target'*** in this case is the size we would use in points. The 'context' is what we think the average person's device is set to. For instance, we may say that the average person's browser is using a default font size of 14pts. Thus, our context would be 14pt. Figuring out the context is a bit of guess work and possibly doing some research.

Open up ***style.css*** and take a look at the ***body {}***. You'll see that the base text is set to 14pt. Let's convert this to rems. For this lab we'll say that average person default text size or context is 14pt. Our formula would be this:

$$\frac{\text{14pt}}{\text{14pts}} = \text{1 rem}$$

Now let's set our line height to rems as well. Keeping 14pt as our context our formula would look like this:

$$\frac{18pt}{14pts} = 1.29\ rem$$



Convert the rest of the font measurements in style.css to rems. Do not change the pixel measurement yet, we'll do that next.

To test the responsive fonts go to settings to change the type size. Type can be changed for Chrome under *Preference > Apperance > Font Size*. For Firefox you can change this under *Preferences > General > Language and Apperance > Fonts and Colors*. You shoudl see all of your type scale as a group.

Responsive Padding, Borders and Margins

In addition to making our type responsive, it helps if other elements are made responsive as well. This includes padding, margins and borders. Typically these CSS properties are measured in pixels, but we can convert those to rems as well. The formula is a slightly different. First we'll convert pixels to points and then divide by the context, just like for type. The formula will look like this:

$$\frac{Target \times 3/4}{Context} = REM$$

In the style.css find #splash {}. Currently the height of the id is set to 400px. We will still use 14pts as our context.

$$\frac{400px \times 3/4}{14pt} = 25\ rem$$

Convert the rest of the font measurements in *style.css* to rems.

Responsive Media

Open *responsive.css* into your code editor. You'll notice that there are two CSS stylesheets. In web development you can have as many stylesheets attached to a HTML file that you want. The reason for having more than one stylesheet is better organization of our CSS. In this lab we will put all of the CSS related to responsive web design in the response.css documnet.

Although we can set the CSS width and font-size to rems, one thing we can't convert to rems is the width and height of media items like images, audio and video. The reason we can't convert is that the pixel width is baked into the image itself. This can cause issues when the page gets smaller and the images do not. The result is that images and media may become larger than the containers or even the page they sit in. To fix this we can use a simple CSS hack.

```
img {
  max-width: 100%;
}
```

What this says is that an image's maximum width can not be larger than 100% or the width of the element it is inside of. The result is as the page gets smaller the images will scale as well.

Add the code to *responsive.css*. This hack would also work for audio elements and video elements such as embedded YouTube videos and Soundcloud audio files.

Responsive Design

Responsive web design means a design that can change depending on the device that the webpage is being displayed on. This means a design for a large desktop / laptop monitor will probably not work for a smaller smartphone. This include layout such as columns, which items to include and not include. Also keep in mind that most smaller devices have a touch interface where as larger devices typically use a mouse. Okay, let's get started.

Add Viewport

Open foodgo.html and add the following code into the <head> section.

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
```

This tells browsers not to zoom in or zoom out when viewing webpages that have responsive designs built into them.

Add @media queries

Media queries are instructions to the browser when to show specfic CSS. For instance the following

@media screen and (min-width:600px){}
Means only use the following CSS if the browser is wider than 600px.

@media screen and (max-width:600px){}

Means only use the following CSS if the browser is less than 600px wide.

Small Devices

Let's set up how the browser should look for small devices. Small devices for this lab are browser less than 600px wide. In responsive.css add the following:

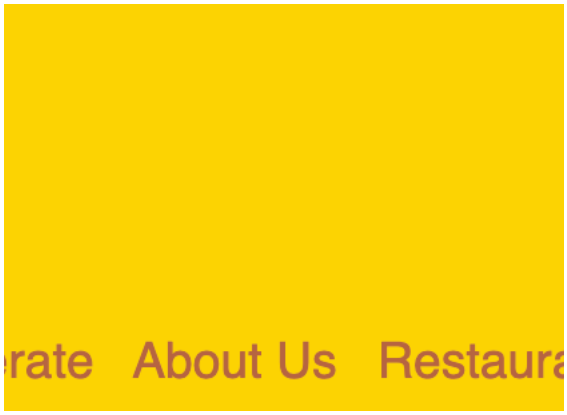
```
@media screen and (max-width:600px){

}
```

Let's look at *FoodGo-Wireframe.pdf* page 2 to see how the mobile layout should look. You can see that for desktops the *<header>* has two columns where as for mobile the logo and navigatio bar are set to a single column. To have this change we will reset the grid-template-columns CSS property from two column at 50%

each to a single column that is 100%.

The code would look roughly like this:

```
@media screen and (max-width:600px){
    header {
    grid-template-columns: 100%;
  }
}
```



Save and preview. You can test the responsiveness of your webpage by using responsive checker in your browser. To access the responsive checker in Google Chrome open the Developers Toolbar by right-clicking and choosing *View > Developer > Developers Toolbar*, or by choosing '*Inspect Element*' when right-clicking on a webpage. Choose the icon on the lower left that looks like a smartphone / tablet. This will give you responsive mode. You can click on the sides to resize the webpage without needing to resize your browser. Bring it below 600 and you should see the design change from one column to two columns.

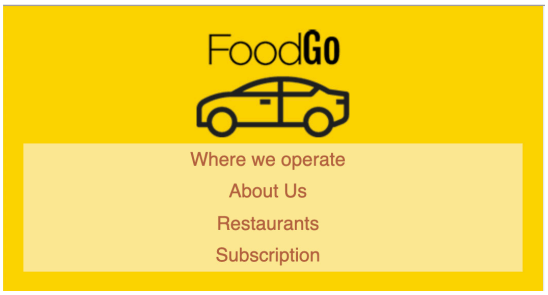
Readjust Navigation Bar

In mobile view we may want to alter our navigation bar, one reason is that these links may be too small for individuals on a smartphone or tablet to click on. We can readjust the links by changing the CSS.

We'll select the navigation links by using *#nav a*. Then change the hyperlinks from inline elements to blocks. Let's also add a *background-color* to make them stand out a bit, we'll use orange. This will make the hyperlinks extend the width of the page.

```
#nav a {
  display: block;
  background-color: orange;
}
```

Save and preview. Your page should something like the image on the right.



Add *text-align: center* and *padding* to give the buttons more space.

Add *text-align: center* to the container holding the logo, which is *#logo* to put the logo image in the center of the page.

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

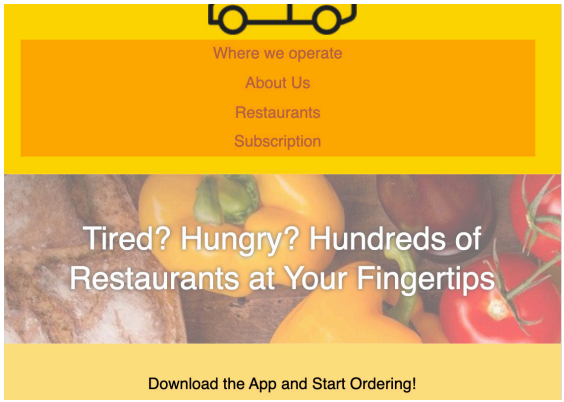
Resize the Splash Text

For large desktop browser the splash text works well, but as the page gets smaller it becomes too overwhelming. Will fix that by resetting the text for *#splash* for browser smaller than 600px.

Inside your *@media* for small devices use *#splash* to set the font-size to 2rem and line-height to 2.5rems.

```
#splash {
    font-size: 2rem;
    line-height: 2.5rem;
}
```

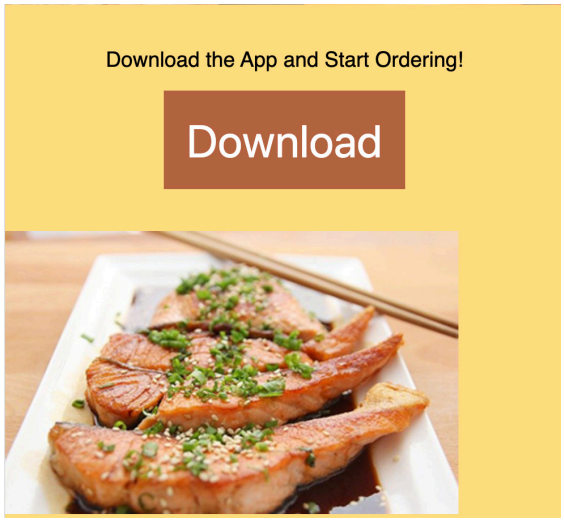
Save and you'll see for small devices the text gets smaller. The issue is now the height is too tall, there is too much extra room on the top and bottom. Let's readjust the height using the CSS height property from 21.4rem to 10.7rem.




```
#splash {
    font-size: 2rem;
    line-height: 2.5rem;
    height: 10.7rem;
}
```

Re-adjust <main> Layout

If you look at FoodGo-Wireframe.pdf you'll see that the #app area and restaurants images are being displayed as a single column. For desktop the design is using CSS grids and having three columns, we'll need to reset this to one column.



Set *<main>* using *grid-template-column:100%* to go to one column for mobile view.

```
main {
   grid-template-column: 100%;
}
```

Save and you'll see that the design becomes one column for smaller sizes.

Fix Col-Span Issues

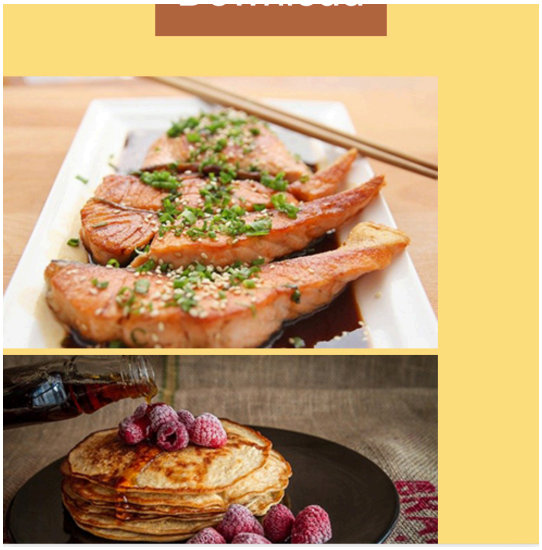
You might notice that only one of the restaurant images is appearing, why is that? The answer is that originally the #splash area and the #app area with the download button is using a column-span. This mean the app is really taking up multiple

columns, thus it is pushing down below to affect the restaurant columns.

To fix this issue we will resize the *#splash* and *#app* colums so that they only span one column instead of multiple when being displayed in mobile.

We'll use *grid-column: n / n* to give it the start marker and the end marker.

```
#splash {
  grid-column: 1/2;
}

#app {
   grid-column: 1/2;
}
```

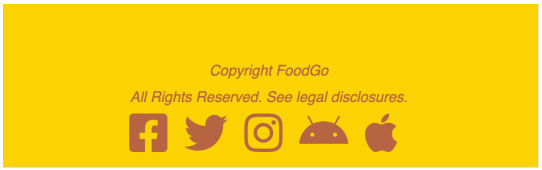


Save and your webpage should look like the image on the right.

Finally, let's put the images in the center of the page by applying *text-align:center* to the *<main>*

Fix the Footer

Now that you havae the basics set the *<footer>* so that each section gets it's own row. Also make sure that the text / social media icons are centered, but only for mobile view.



Your page should look like this when you are done.

Adding a Menu Icon for Small Devies

Since smartphones and tablets screens are smaller typically than desktop and laptop monitors we sometimes have to be selective of what we show in mobile menu. Many websites for smaller devices as menu icon, sometimes referred to as a hamburger menu that allows users to click on it and get additional options. This menu only appears when the page is small, since that is when users need it.

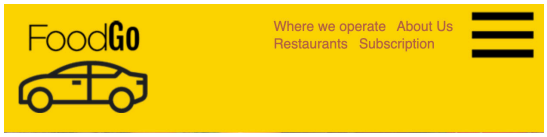
```
<div id="mobilemenu">
    <img src="menu-icon.png">
</div>
```

Let's add that to our website. In this lab the menu icon will not actually work since that would require Javascript, but we can at least add it for visual effect.

Go back to foodgo.html. In the lab files you'll notice an image called menu.png. Let's add it to our HTML. Place it in a * * above the *<header>*, give it an id of "*mobile-men*u".

Add CSS to the Mobile Menu Icon

Open style.css and use position: fixed, top:10px and right:0px to place the mobile menu icon on the upper right hand side of the webapge.




```
#mobile-menu {
   position: fixed;
   top: 10px;
   left: 10px;
}
```

Save and preview

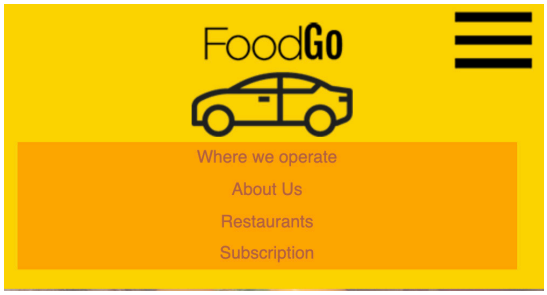
Make it Responsive

You'll notice that the mobile icon is appearing for both small browser widths and large one. We only want it appear for small one so we can fix that with a @media query.

Create a @media for large devices, in this lab it will be for devices LARGER than 601 pixels.

@media screen and (min-width:601px){}

For large devices set #mobile-menu to display: none, meaning it will not appear at large sizes.



Save and preview.

Submission

Students will submit a .zip folder with all the necessary files to iLearn.

Students will also upload the lab, including all necessary files to their Github accounts. Students will submit a link to the foodgo.html file and submit it along with their files on iLearn.

Extra Credit (Not Required) + 3

Add in a 'dark mode' for the webpage so that when the operating system is set to 'dark mode' that the background colors for foodgo.html become dark and text becomes light colors.

Look back at class examples or use this web tutorial for more information

https://davidwalsh.name/prefers-color-scheme