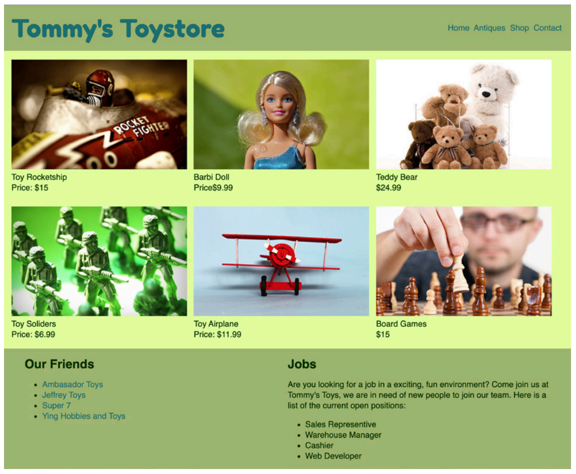
Using CSS Grids and Flexbox for Web Layout

In the past labs we've discussed using CSS floats and CSS positioning to lay out content in a webpage. Today we will discuss a two additional CSS layout methods, CSS grids and flexboxs. In this lab we'll use both to create the layout for our client, Tommy's Toys.



Tommy's Toystore

Home Antiques Shop Contact

Toy Rocketship
Price: $15

Barbi Doll
Price$9.99

Teddy Bear
$24.99

Toy Soliders
Price: $6.99

Toy Airplane
Price: $11.99

Board Games
$15

Our Friends

- Ambasador Toys
- Jeffrey Toys
- Super 7
- Ying Hobbies and Toys

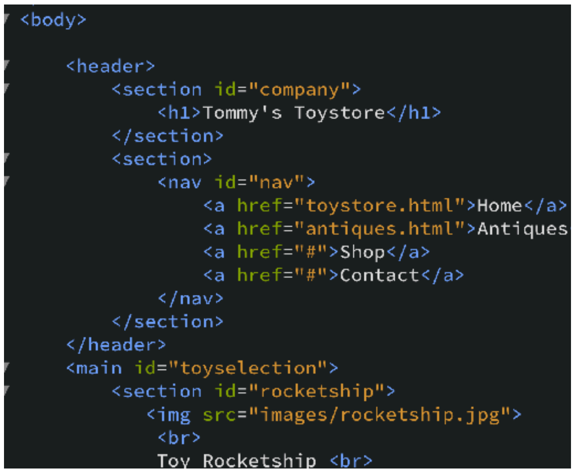
Jobs

Are you looking for a job in a exciting, fun environment? Come join us at Tommy's Toys, we are in need of new people to join our team. Here is a list of the current open positions:

- Sales Representive
- Warehouse Manager
- Cashier
- Web Developer

Overview

Take a moment to open the *toystore.html* file and *style.css* file into your coding program. If you look at the HTML we have a basic layout that is structured with a *<header>*, *<main>* and *<footer>*. The header have two *<sections>*, one for the company title and one for the navigation. Inside the *<main>* each toy has it's own *<section>* with the toy image and price. The footer has two major *<section>*, one for friends of the store and the other posting about job oppurtinties.



```
<body>

    <header>
        <section id="company">
            <h1>Tommy's Toystore</h1>
        </section>
        <section>
            <nav id="nav">
                <a href="toystore.html">Home</a>
                <a href="antiques.html">Antiques
                <a href="#">Shop</a>
                <a href="#">Contact</a>
            </nav>
        </section>
    </header>
    <main id="toyselection">
        <section id="rocketship">
           <img src="images/rocketship.jpg">
            <br>
            Toy Rocketship <br>
```

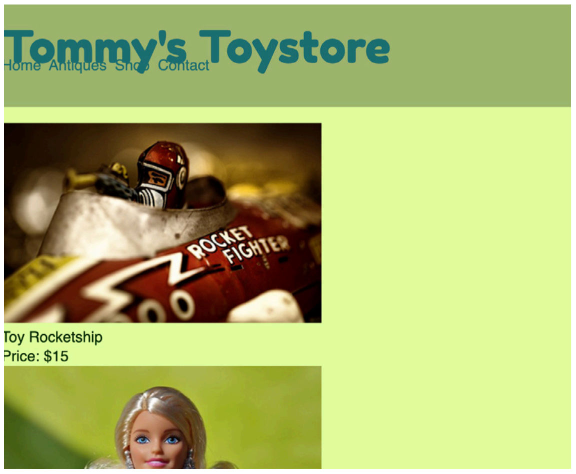
Take a moment to look at the CSS and you'll see that the fonts and backgrounds have been set. You can look at the *Wireframe.pdf* to see the general layout of what we want to create. In this lab we do not need to edit the existing CSS, although we will add to it.

Web Fonts

Let's start with some practice of previous ideas. In this design we want to a web font to add visual interest to the project.

Embed the Google Font '*Fredoka One*' into the project into the toystore.html and add CSS to *style.css* to stylize the h1 to '*Fredoka One*'

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



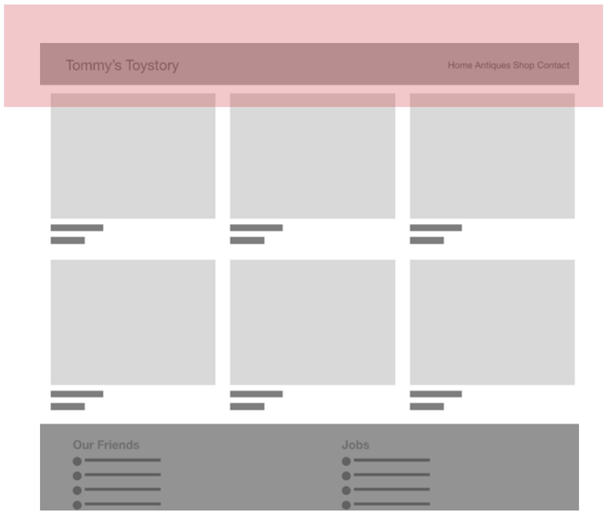
Tommy's Toystore

Home Antiques Shop Contact

Toy Rocketship
Price: $15

CSS Grids

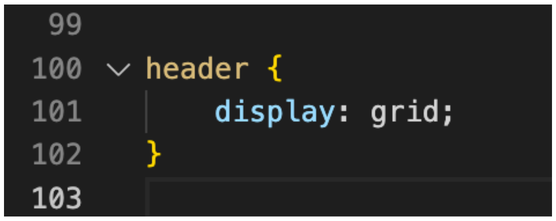
CSS grids takes a page and divides it into rows and columns, this is similar to an Excel spreadsheet. The concept of columns has existed in design and page layout for many years, and now is incorporated into CSS.



If you look are *Wireframe.pdf*, page one you'll see that the header section should be into two columns with the company name on the left and navigation bar on the right. To quickly create columns we can use CSS grids.

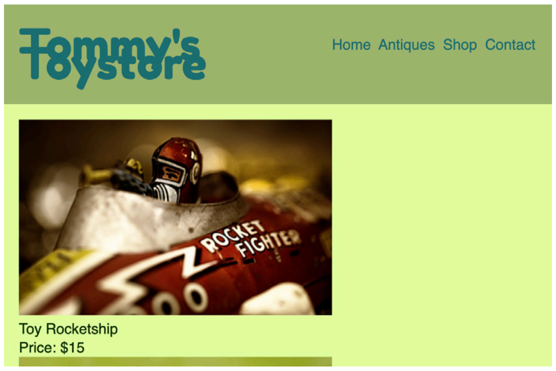
Let's begin by creating a grid. In the CSS set the header to display: grid as shown below

```
header {
  display: grid;
}
```



```
header {
    display: grid;
}
```

This sets the *<header>* block to become a grid. When something is set to become a grid, it's direct children become the columns. In this case there are two direct children of the *<header>*, which are the two *<section>* tags. These will become the columns.



In order for the browser to know how many columns to create we must add a CSS property called grid-template-columns. This allows us to specify how many columns and what their widths are.

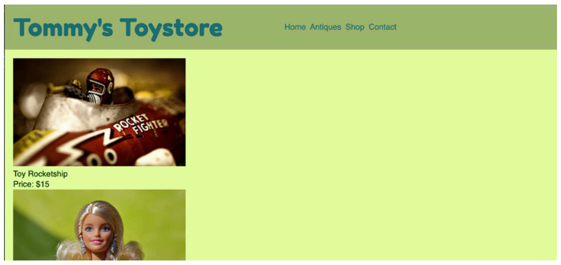
Let's create two columns that are both 400px wide. We would write the code like this:

```
header {
  display: grid;
  grid-template-columns: 400px 400px;
}
```

This states the first column will be 400px and the second column will also be 400px.

Save and Preview

Sometimes though, its easier to set the width to relative width like 50% as opposed to finite width like 400px. The reason why is that it becomes easier for the webpage to readjust itself if the user makes the webpage larger or smaller. It also translates better begin many different devices. So let's change it from 400px to 50%;



```
header {
  display: grid;
  grid-template-columns: 50% 50%;
}
```

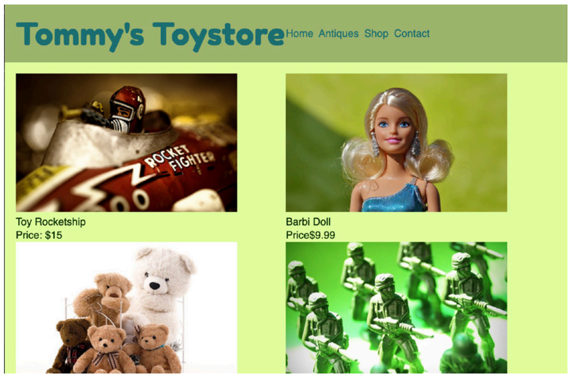
Save and Preview.

Layout the Toys

Now that we've laid out the <header> let's do the same for the <main> section which contains the toys. The <main> is not affected by our previous grid because it is not a direct child of the <header>, so to affect the <main> let's set it to a grid as well.

```
main {
  display: grid;
}
```

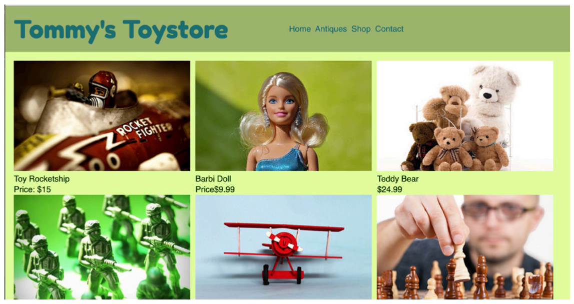
If you look at Wireframe.pdf, page one you'll see the client was thinking about laying out all of the various toys into two columns. So let's use grid-template-columns and set the width to 50% each.



```
main {
  display: grid;
  grid-template-columns: 50% 50%;
}
```

Save and preview. You'll see you have two columns and that since the grid has six children, that you've created three rows with them. Your webpage should look like the image on the right.

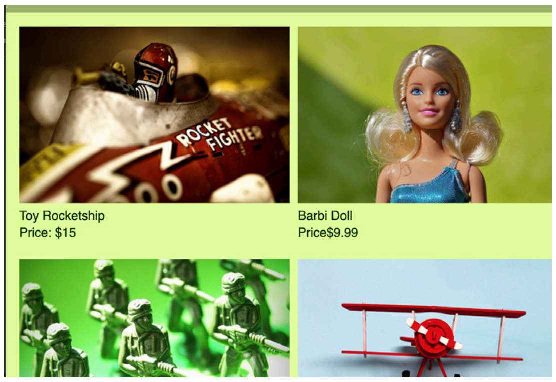
After the client looked at the result they decided that two columns was not working because customers could not see enough of the products, so they decided they wanted the toys to be displayed in three columns, instead of two. Believe it or not this is not too hard to fix, we'll simply go back to the CSS and change 50% to 33% and add a third column.



```
main {
  display: grid;
  grid-template-columns: 33% 33% 33%;
}
```

Save and Preview. You see that you have three columns now instead of two.

If you view your page you'll notice that the rows are fairly tight with each other. We can fix that by using a CSS property called grid-row-gap. The value is the space between our rows.

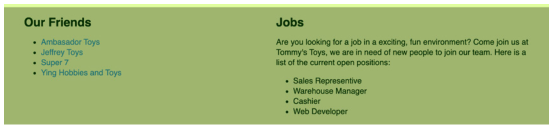


```
main {
  display: grid;
  grid-template-columns: 33% 33% 33%;
  grid-row-gap: 25px;
}
```

This means you'll have 25px between each row in our design.

Layout the Footer

Now you have a basic idea of how to use CSS grids and create columns set the <footer> to have two columns, each 50% of the page.



Our Friends

- Ambasador Toys
- Jeffrey Toys
- Super 7
- Ying Hobbies and Toys

Jobs

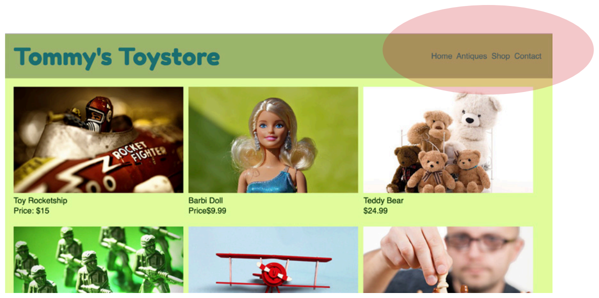
Are you looking for a job in a exciting, fun environment? Come join us at Tommy's Toys, we are in need of new people to join our team. Here is a list of the current open positions:

- Sales Representive
- Warehouse Manager
- Cashier
- Web Developer

You webpage should look roughly like the image on the right.

Align to the Grid

Although we have a layout of rows and columns sometimes we'll want to position something within that column. If you look at the navigation bar you'll see it sits in the middle of the page. That is because it is sitting on left of the right column, placing it roughly in the middle of the page. For our design we actually want the navigation bar to sit on the right side of the column. To fix this we can use previously taught concepts like text-align or float. Let's use text-align. In your CSS select the <nav> and use text-align to set it the right side of the column.

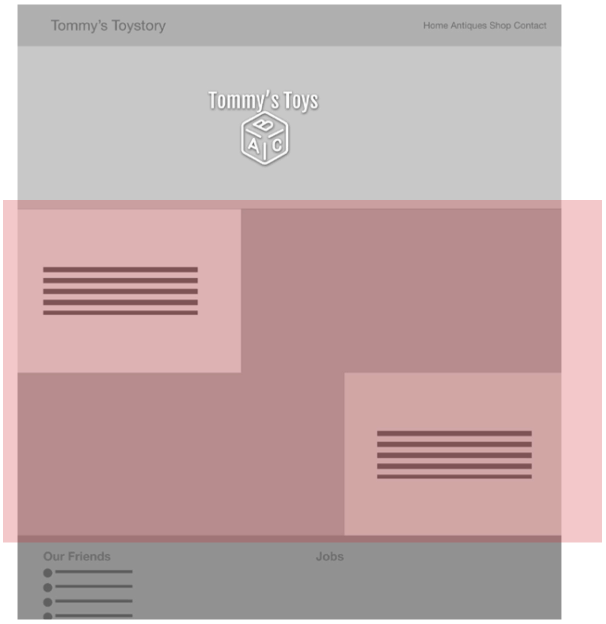


```
nav {
  text-align: right;
}
```

Save and Preview

Create Mulit-Span Columns

So, at this point you should have a basic understanding of how to create grids with columns. Let's up the ante a bit with a design that has columns that are uneven, that is to say where one row have columns of different length to the row below it. We'll be applying these ideas to *antiques.html*.



Take a moment to look over *Preview*, page two and you'll see that the in the middle of the page there are antique toys being displayed with tet. In the first row the text that is roughly 1/3rd of the page and an image that takes up 66% of the page, then the then second row starts with an image that is 66% and followed by text that is 33%. The final row and three columns, each with it's own image. So it's a bit hard to see how to easily do this. Would we need to make three separate grids? The answer is no, instead we can make columns that spans multiple columns.

Setup

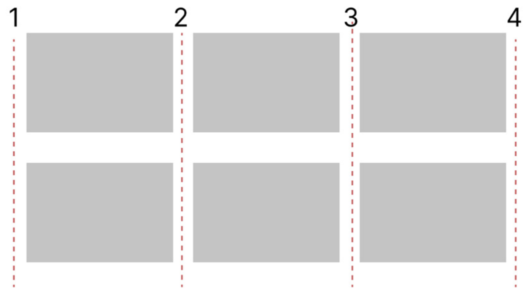
In *antique.html* you'll see that we have a *<div>* with an id of *#antiques*. Inside *#antiques* are the sections we will lay out using CSS grids.

Begin by setting *#antiques* to *display:grid* and have the layout have three columns, each being 33.33%.

The result should look something like the image on the right.

Colspan

You'll notice that the columns do not look like our Preview. That is because each row should not have three columns each. The first row should have two, the second should have two and the third row should have three.

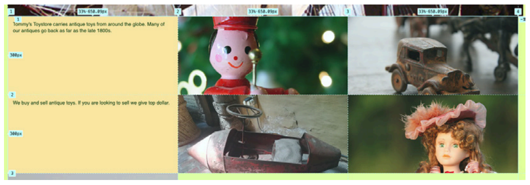


How do fix this? The answer is a *column-span*. A *column-span* allows a column to span over multiple columns. For instance, if there are three columns, a column can span: 1 column, 2 columns or 3 columns. If there were four columns, a column could span: 1 columns, 2 columns, 3 columns or 4 columns. To do this will use CSS *grid-column-start* and *grid-column-end*.

To understand how to use *grid-column-start* and *grid-column-end* we need to understand column grid and row grid markers. Column grid markers begin at the beginning of the grid and then go between each column. Finally there is a grid column marker at the end horizontal end of the grid. You can view this through the diagram.

If you are using the Google Chrome browser or Mozilla Firefox browser you can see the grid lines in the browser itself. When you preview the webpage right-click and go to Inspect. In the *Inspection* panel click on the small pill shape to the right of *#antiques* that says *"grid"*. The result will be that the grid column and grid row markers will be shown in the browser.

You can see that there markers at the start of the grid, the end of the grid, and in-between each column.

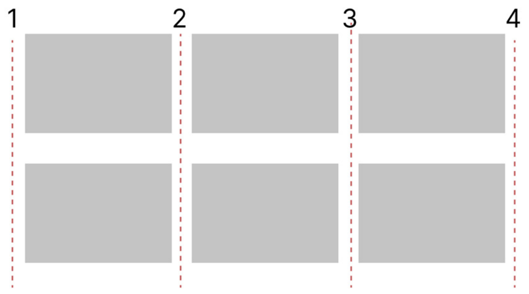

Tommy's Toystore carries antique toys from around the globe. Many of our antiques go back as far as the late 1800s.

We buy and sell antique toys. If you are looking to sell we give top dollar.

If I wanted to make a column span more than a single column you must specify the start column marker and the end column marker. To make a column go from the beginning of the the grid to 2/3rds inward you would put the following:

```
grid-column-start:1;
grid-column-end:3;
```

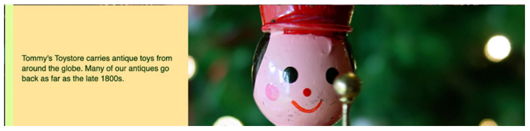
Apply Colspan in Our Design



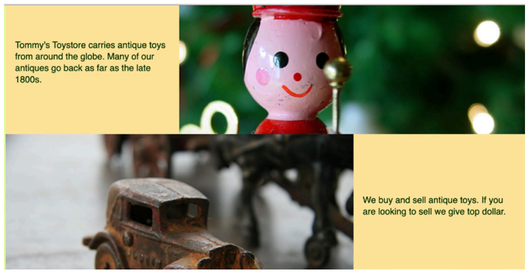
Let's try the same thing for the first row of our design. In the Preivew you'll notice that the #splash area of the toy soldier spans two columns. In our page since there are three columns the markers will look like the image on the right.

To select the text you'll see it inside a *<section>* with an id of *#splash2*. Set it to the following:

```
#splash2 {
  grid-column-start:2;
  grid-column-end: 4;
}
```


Tommy's Toystore carries antique toys from around the globe. Many of our antiques go back as far as the late 1800s.

Save and preview. It should look roughly like the image on the right.


Tommy's Toystore carries antique toys from around the globe. Many of our antiques go back as far as the late 1800s.

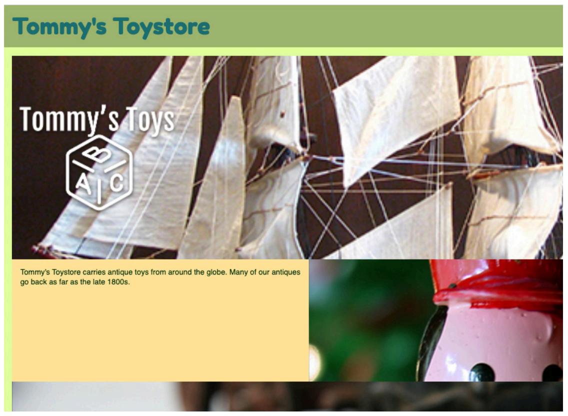
We buy and sell antique toys. If you are looking to sell we give top dollar.

Now make a column span for #splash3 to make it span multiple columns. You image should look the image on right.

Flex Box

The final method we will discuss for laying out content is known as *flexbox*. The reason flexbox was created was that developers found it difficult to lay out material easily. For instance, you could set something to the vertical center of a container by using margin-top or line-height, but what would happen if that container changes its height? Than the item inside would no longer be in the center.

Another issue that kept occuring was wanting to layout multiple items evenly. Let's say you a gallery <section> of images and you want to make sure they are laid out equidistantly from one another, there was no easy way to do that.


Tommy's Toystore

Tommy's Toystore carries antique toys from around the globe. Many of our antiques go back as far as the late 1800s.

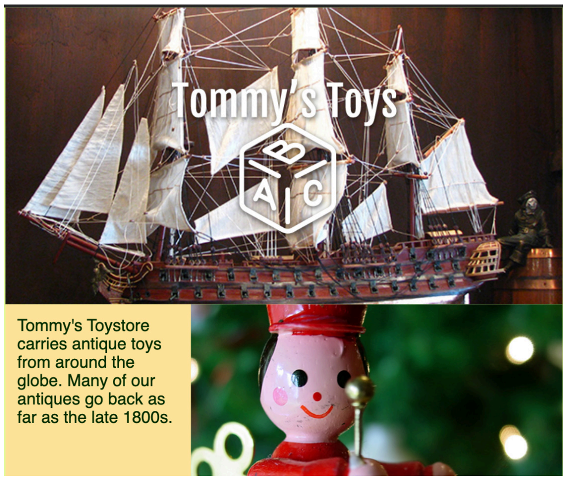
Flex box is a technique that turns a container like a *<div>*, *<section>* or *<main>* into a special flexible *box*.. That box can then lay out it items in the horizontal and vertical space easily.

Take a moment to look at the *Preview*, page 2 and notice that the Tommy's Toy logo is in the vertical and horizontal center of splash section.Make *#splashSection* a Flexbox

Let's make *#splashSection* a flex box. To do this set *#splashSection* to display: flex;

```
#splashSection {
   display: flex;
}
```

The *#splashSection* is now a flexbox. Now that it is a flex box let's set the logo to the vertical center. To do this we'll use *align-items*. *Align-items* can set things to the top of the box, or *flex-start*, center or the bottom of the box, *flex-end*. To place in the vertical center use the value *center*.



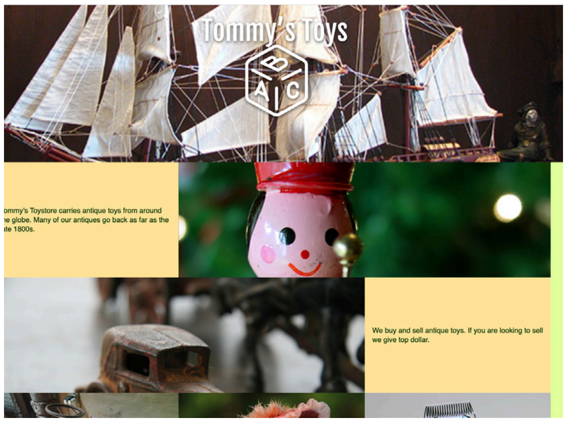
Tommy's Toystore carries antique toys from around the globe. Many of our antiques go back as far as the late 1800s.

```
#splashSection {
   display: flex;
   align-items: center;
}
```

Save and preview the result on the right.

Now that the vertical alignment has been set up, let's set the horizontal alignment to the center as well, we can do this by using another flex property called *justify-content*. Like with *align-items* we can use *flex-start* to place items at the start or left side of the container. *Flex-end* or the right side of the container, or use center for the horizontal center of the container.

```
#splashSection {
   display: flex;
   align-items: center;
   justify-content: center;
}
```



ommy's Toystore carries antique toys from around
e globe. Many of our antiques go back as far as the
te 1800s.

We buy and sell antique toys. If you are looking to sell we give top dollar.

Apply the same *display: flex*, *align-item*s and *justify-content* to *#antique-text-1* and *#antique-text-2.* The results would look like the image on the right.

Submission

Submit the completed zipped folder to iLearn.