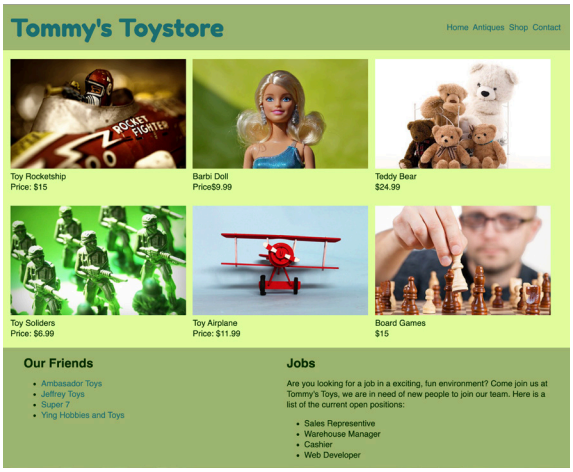
Using CSS Grids and Flexbox for Web Layout



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 methods, CSS grids and flexboxs. In this lab we'll use both to create the layout for our client, Tommy's Toys.

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.

CSS Grids

CSS grids takes a page and divides it into rows and columns, similar to an Excel spreadsheet. The idea 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;
}
```

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%;
}
```

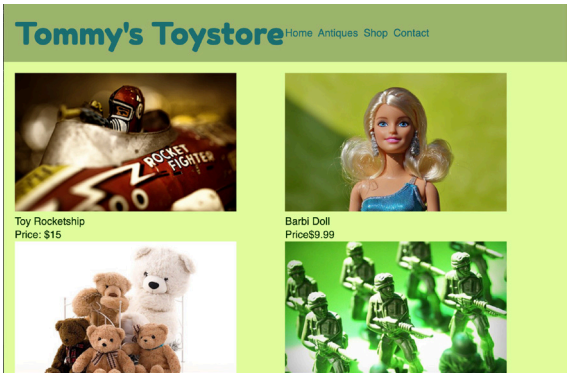
Save and Preview.

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;
}
```

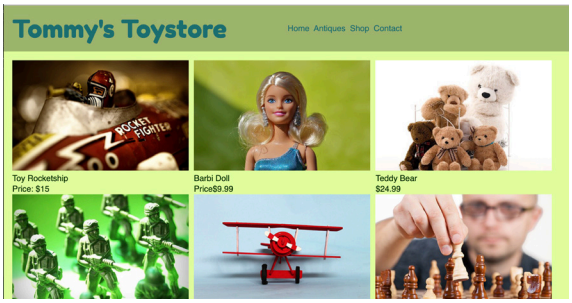
If you look at Wireframe.pdf, page one you'll see the client was thinking about laying out all of the various toys into two columns. So let's use grid-template-columns and set the width to 50% each.




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

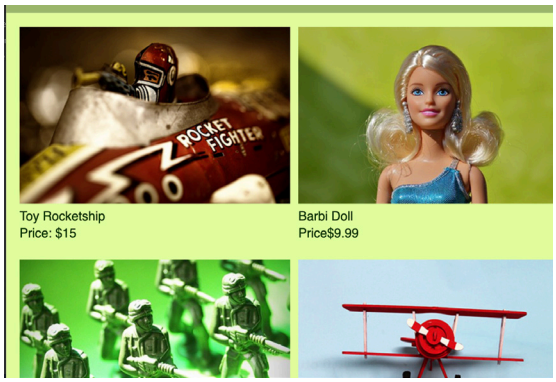
Save and preview. You'll see you have two columns and that since the grid has six children, that you've created three rows with them. Your webpage should look like the image on the right.

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.



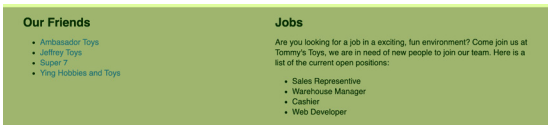
If you view your page you'll notice that the rows are fairly tight with each other. We can fix that by using a CSS property called grid-row-gap. The value is the space between our rows.

```
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

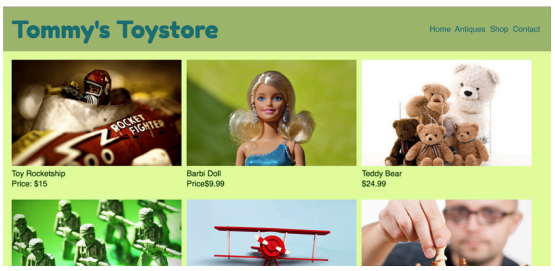
Now you have a basic idea of how to use CSS grids and create columns set the <footer> to have two columns, each 50% of the page.



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 with an id of #antiques. Inside #antiques are the sections we will layout with a grid.

Begin by setting #antiques to a CSS grid and give it 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 third 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 let's look at the example at the right. Imagine I have two columns. There is a start place to the far left, that is position 1. Between the first and second column is position 2 and at the far right is position 3.

If I wanted to make a column span more than a single column you must specify the start position and the end position. To make a column go across two columns you would put:

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

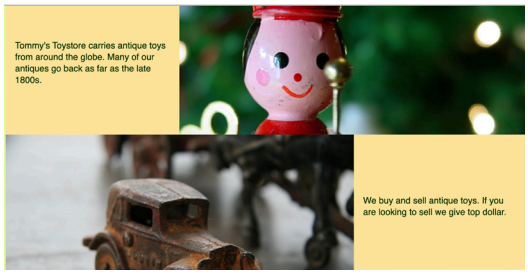
Let's try the same thing for the first row of our design. You'll notice that the #splash area of the toy soldier aspans two columns. In our page since there are three columns the positions look like the image on the right.

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;
}
```

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

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 material is known as flex box. 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 section of images and you want to make sure they are laid over equidistant from another another there was no easy way to do that.

Flex box is a technique that turns a container like a * *, *<section>* or *<main>* into a special '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;

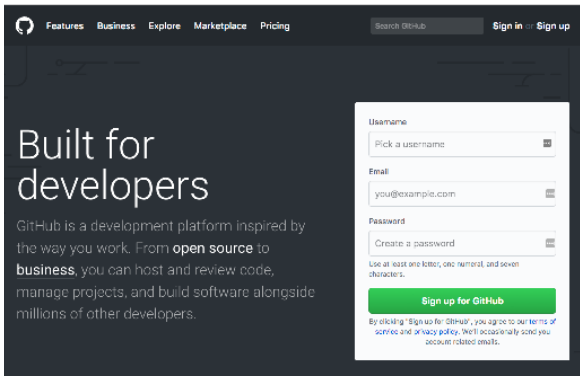
```
#splashSection {
   display: flex;
}
```

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.

Save and preview the result 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.

Learn to Use Github



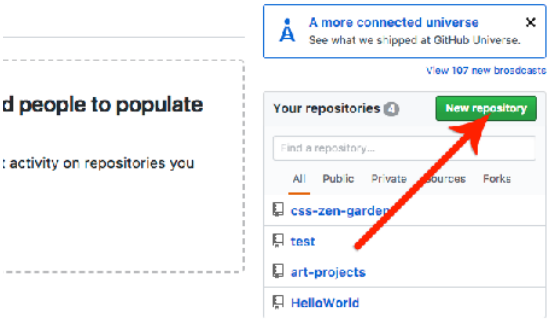
Github is a great resource that is used by many developers. It allows projects to be easily worked on multiple people without issues of workers having to hunt down the latest files, or people overriding each other's work. It also can be used to host webfiles as a server.

We will create a Github server repository and upload our this lab to it.

Setup a Github account

Begin by setting up a Github account. You can do this for free at their website http://github.com. The account is free.

Create a Repository



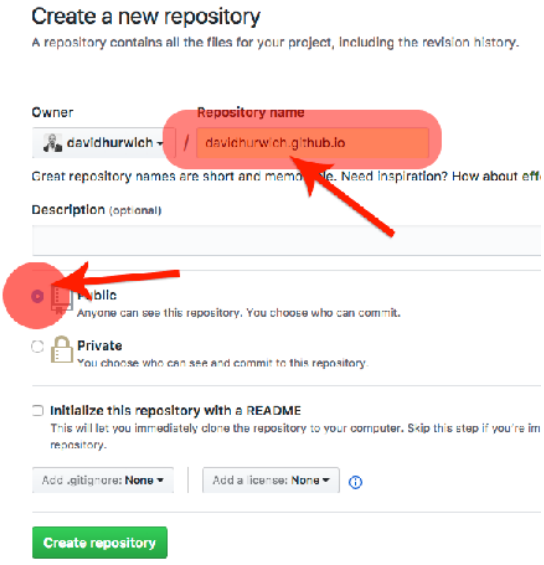
A repository is a place to store files. These can be any files, HTML documents, CSS documents, images, audio, video and more. Think of a repository as a project folder.

To create a new repository click on the big shiny green button marked "+ New Repository".

After this you'll be asked to name the repository. When naming your repository label it

github-username.github.io

This repository will be your server, or the place you upload web files to in order to make them live.



Uploading Files to Github

There are many many ways to upload files to Github, you can use a terminal, but in this class we will use a desktop application called GithubDesktop. It will make the process MUCH easier for you.

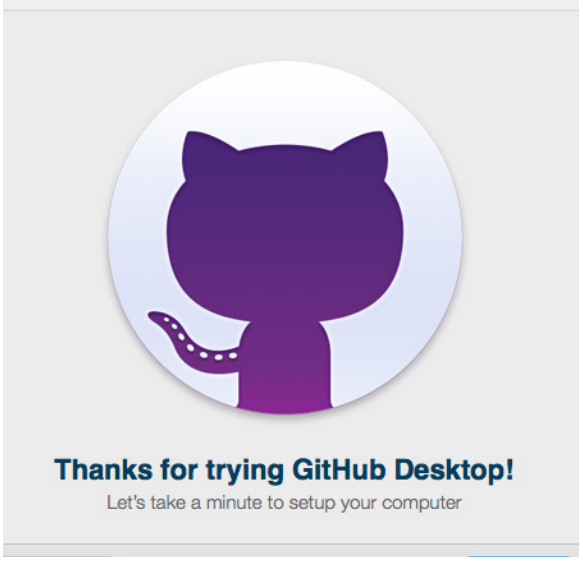
If you are working on your computer you can download GithubDesktop for free at ou can download GithubDesktop for free at https://desktop.github.com/

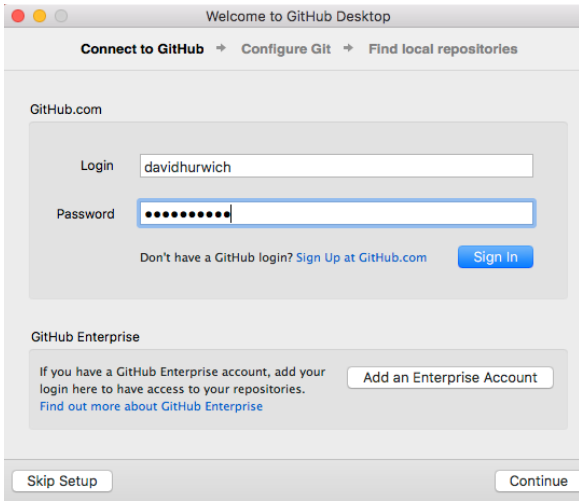
Github Desktop Application Login

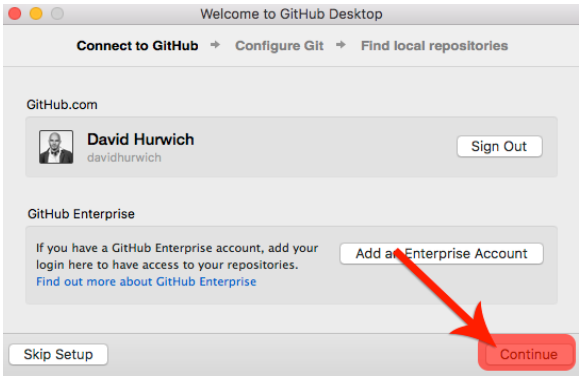
After launching the Github Desktop application begin by logging into the application, typically when you launch the program for the first time you'll be asked to login. Use your github.com user and password to login into the Github Desktop application.

If you are not asked to login during the startup process go to *Github Desktop > Preferences > Accounts* to login into application.

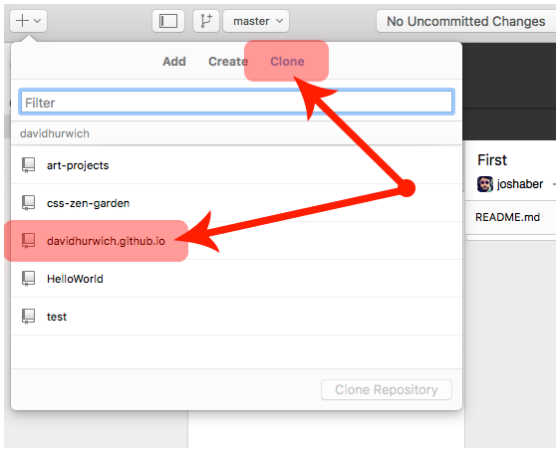
Logging in is important so that you can share files from your Github Desktop app to github.com.







Clone your username.github.io Folder



The easiest way to upload your files to Github server is to create a synch folder. When files are added to this folder it will allow them to be uploaded to your Github server.

Click on the "+" sign in the upper left hand corner and choose the option marked "Clone". You'll see it lists all of your Github directories. Choose the "username.github.io" folder to clone. You'll be asked to choose a place to clone the folder. For the easiest workflow choose the Desktop.

The Github Workflow

The Github Workflow is the following:

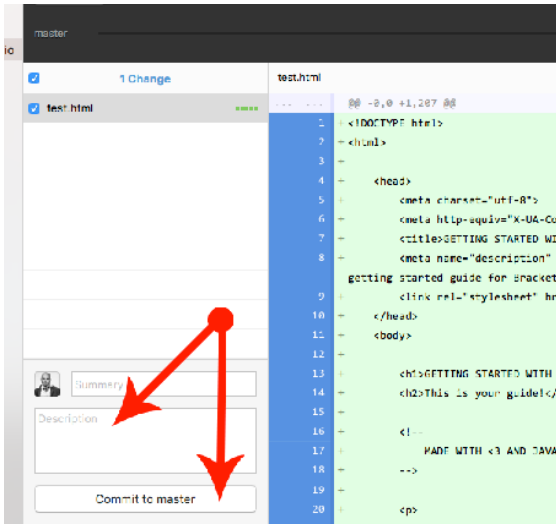
- Add
- Commit
- Synch

You add files to your cloned folder. These are the files that you want to upload to your Github Server.

Before you upload you must commit to the upload, or tell Github you are uploading and adding new file.

Finally, you synch or actually upload the files.

Add Files to Your Cloned Folder



The cloned folder acts as a synch folder. Think of it like your local Dropbox folder. Any file added to it will eventually be uploaded to your Github Server.

In this example take the files for this lab and add them to your cloned folder.

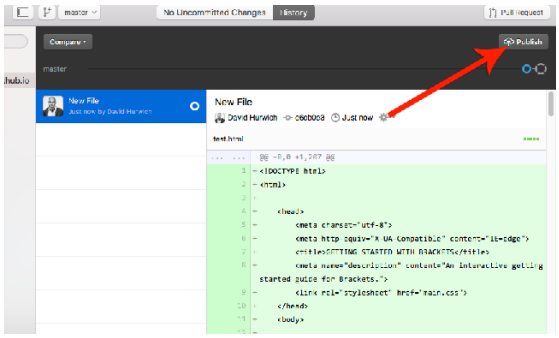
If you go back to the Github Desktop Application you'll notice that it shows there is a change or update to the upfolder.

Commit Changes

If you want to update your username.github.io server any new files or changes must be committed. The act of committing tells Github you are going to make changes.

Go to commit a new or updated file go to the bottom and click on the 'Commit to Master' button. Before clicking that you'll need to add a log note. The log note describes the new changes or updates. For instance, you could say "Adding new file" or 'Updating index.html file'.

Synch New / Updated Files



Now that we've added files and committed files. Let's synch or update the files on the github.io server.

Click the button in the upper right hand side marked Publish / Synch.

Publishing or Synching your file sends it back to your username. github.io server. You can double check that the Github publishing worked by logging into your Github.com account and looking in your repository. You should see the file uploaded.

Check Your File Online

Now that your files have been uploaded to your Github server they are live, or can be found on the internet.

You should be able to view them by going the following URL

username.github.io/filename

In this case insert your own username and insert the name of the uploaded HTML page, including the .html.

Congradulations, you are now publishing to the web like a pro!

Submission

Submit the completed zipped folder to iLearn. Also upload a copy of the files to your Github server and include a valid URL to the file.