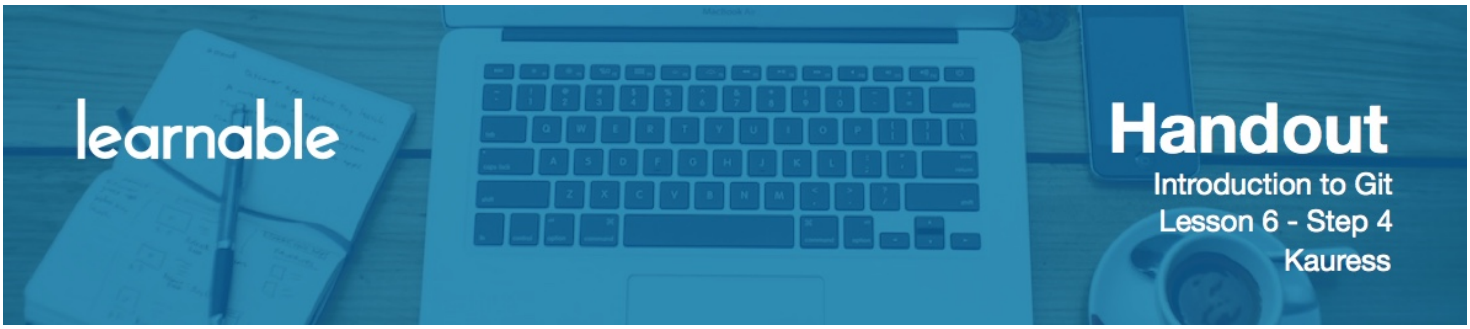

Introduction

A common reason why pull requests can fail is when you have conflicting changes in your files (conflicting information in the same part of the same file). Git will pick up on these conflicts and ask you to resolve them in your code editor manually. So let's go ahead and play out this scenario.

Work Through a Failed Pull Request

Go to GitHub and now let's edit the *business-casual.css* file on behalf of Lara. Suppose she didn't like the color for the `brand` class and replaced it:

```
.brand {color:#40E0D0;}
```

Commit the change with a message "changed color to another".

Once that's done come back to your code editor and go to your CSS file. So, you decided to change the color of brand class once again:

```
.brand {color:#fab11c;}
```

Save the file and run

```
git add .
git commit -m "Changed color to a mustard shade"
```

From previous experience you know that you should pull any remote changes before pushing local changes. Go ahead and do that with

```
git pull origin master
```

It tells you that the merge failed because there are conflicts in our *business-casual.css* file which we have to fix manually. Go to your CSS file in the code editor. The conflict area is marked by the angular brackets (`<`). `HEAD` denotes the commit for our own branch, and below that we have Lara's color choice. The lines (`=`) divide the conflict.

Now, at this point, you will discuss with your teammate which option to keep, and then delete the color that you don't want. Imagine that we discussed this with Lara and we've decided to keep our color and delete hers. Once that's done, go head back to your terminal and run

```
git status
```

It's telling us that we should add and commit our changes again. Type in

```
git add .
git commit -m "Kept mustard color"
```

Lastly push our commit with

```
git push origin master
```

Going to GitHub, you can now see our latest commit, where we kept the mustard color.