



CodeFirst:Girls Beginners Coding course - Front end Web development

Week 4 - GitHub & Version control

What we'll cover this week:

1. Recap of what we've learnt so far
2. Version Control & Using GitHub
3. Create your group repository
4. Publishing on GitHub Pages
5. How Github Pages Works
6. Conflict scenario!
7. How to avoid and reduce merge conflicts
8. Homework: Create GitHub repository for the course project

WHAT WE'LL COVER THIS WEEK

1. Recap of what we've learnt so far
2. Version Control & Using GitHub
3. Create your group repository
4. Publishing on GitHub Pages
5. How Github Pages Works
6. Conflict scenario!
7. How to avoid and reduce merge conflicts
8. Homework: Create GitHub repository for the course project

© CodeFirst.Girls 2017

² Slide 2

Recap session: (20 mins)

TOPICS WE'VE COVERED IN WEEKS 1, 2 & 3

Session 1: Getting going + HTML

- HTML syntax

Session 2: CSS

- Tags, Selectors and Attributes, Stylesheets

Session 3: Frameworks, UX

- Frameworks, Libraries, APIs, User Experience

Task: Find a partner and together take a quick look through the session notes from the last 3 sessions. If you're unclear on any of the concepts work through them with your partner and an instructor

© CodeFirst Girls 2017

3

Slide 3

Run through the topics covered in previous weeks (HTML, CSS and Frameworks / UX).

Task: find a partner and together take a quick look through the session notes from the last 3 sessions. If you're unclear on any of the concepts work through them with your partner and an instructor

Version Control & Using GitHub

GITHUB &
VERSION CONTROL

4

Slide 4

So, what is Version Control and why is it important?

Have you ever worked in a group on a Powerpoint presentation or Word document?

What is that process like? Let's not think about Google Slides or Google Docs for the moment.

Let's start with two people - you and your partner are presenting on the controversial topic of the future of the UK's relationship with the European Union. You've got a structure for the presentation and started delegating the work; You'll work on the first section discussing the state of the relationship, and your partner will start working on the possible future scenarios.

You both go away and make the slides and handout notes separately and meet again in a week. But when you meet, there's a blocker that's completely irrelevant to your material:

You've decided to make your presentation with a black-background theme, in Calibri, and your partner has chosen a completely different colour scheme, used different fonts, and you've both made about 10-15 slides, organised in a different style from one another, and different types of handouts.

Yes, you could have waited for one person to start making the slides and emailed it over, or decided a "style" from the start, but that would have added an extra meeting and maybe a couple of days of waiting back and forth. Either way, you need to spend at least a couple of hours coordinating, or fiddling with the styling of the slides and handouts.

What if there were a way to automate the merging of your material and choose what to keep and what not to keep?

What is Version Control?

- It lets you track your files over time
- If you mess up, you can easily get back to a previous working version
- It's the key to collaborative software development
- You can work in teams on the same project
- You can manage conflicts in code on the same file

Slide 5

Version Control

The answer is Version Control, also known as Revision Control or Source Control. It lets you track your files over time. But why would you care about that?

Because if you mess something up along the way, you can easily get back to a previous working version.

Version control is the key to collaborative software development.

- You can work in teams on the same project easily and manage conflicts in code on the same files;
- You can work on different versions all at the same time;
- You can decide what you want to keep and what you want to bin at the end.

Why do we need version control?

**It's powerful and it can do a lot
more than simply file sharing**

Slide 6

So why do we need a version control system (VCS)?

You may think that sharing a folder on a service like Dropbox or Google Drive might be enough when you are collaborating with other people. Sharing files and folders is quick and simple, but there are a whole bunch of things they can't do.

A version control system is much more powerful and can do a lot more than simply file sharing. In fact it is necessary if you want to do any of the following:

The advantages of version control

Backup and Restore

- Files are being saved as they are being edited
- You can jump back to any moment in time
- Even if it was 10 years ago



Slide 7

The advantages of version control

- **Backup and restore:** Files are being saved as they are being edited. This means you can jump back to any moment in time. Even if it was 10 years ago.

The advantages of version control

Synchronization

- Share files with people
- Stay up-to-date with latest version



Slide 8

- **Synchronization:** It lets people share files and stay up-to-date with latest version.

The advantages of version control

Short-term undo

If you mess up, go back to the last known good version



Slide 9

- **Short-term undo:** If you messed up a file you can throw away all the changes you made and go back to the last known good version in the database.

The advantages of version control

Long-term undo

- If you mess up, go back to the last known good version
- Even if it is quite some time ago



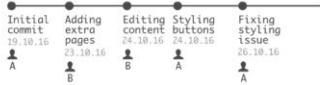
Slide 10

- **Long-term undo:** What if you messed up really badly? Imagine if a change you made a year ago has a bug you only discover today. You can jump back in time to see what changes were made then.

The advantages of version control

Track Changes & Ownership

- Every file change receives a message
- The person making the change is logged as well
- If someone deserves credit: you know who
- If someone needs to be blamed: you know who



Slide 11

- **Track Changes:** As updates are being made to files you add a message explaining what changes you have made. These messages will make it easier to see how a file has evolved over time and why.
- **Track Ownership:** At the same time the messages are being added to changes, the name of the person making the change is added as well. If someone needs to be blamed for the mess, you know who it was. Or you could give someone credit too for a job well done.

The advantages of version control

Sandboxing, branching & merging

- Insurance against yourself
- Test a new feature in isolation
- When everything is tested and OK, add it back into the big pool



Slide 12

- **Sandboxing:** This is a form of insurance against yourself. When you are making a big change you can do so in an isolated area. That way you can test your work and iron out any mistakes before adding it back into the big pool.
- **Branching and merging:** The same as sandboxing but on a larger scale. You can branch a copy of your code into a separate area to modify it in isolation. Later it can be merged back into the common work area.

The main elements of version control

Repository

- The basic unit of GitHub
- A single project folder containing all the files of a project

Branches

- Every repository has at least one branch called Master
- Branches allow you to work on different parts of a repository at the same time

Slide 13

The main elements of version control

The version control system we are going to learn about is Git. It's very powerful and popular. Hard line developers use Git on the command line, and that is where Git is the most powerful, but we'll go down a slightly easier route.

Please be aware that Git and GitHub are not the same thing. Git is the version control system and GitHub is a website where you can view coding projects that use Git.

Before we can start using Git we need to understand what the main elements are of a version control system.

Repositories

- A repository is the basic unit of GitHub. Usually it is a single project or folder containing all the files needed for that project. That includes code files, image files, etc.
- A repository should also include a README file and a license. The README file needs to explain what the project is about and how you might want to use it, if it is something public. The license should outline what the rules and restrictions are in case someone wants to use the code.

Branches

- Branching is how you can work on different parts of a repository at one time.
- When you create a repository it will by default have one branch called Master.

The main elements of version control

Commit

- A change in a repository
- Every commit needs an associated message describing the change
- Commits help others understand what has been done to a repository

Issue

- A note on a repository about something that needs attention

Pull Request

- A request to the repository owner to include your code in their project

Slide 14

Commits

- Each time when you save a change it will be registered. On GitHub this process is called a commit.
- Every time you commit you need to write an associated message. This is a description of the changes you have made and why. It doesn't have to be a full essay, the trick is to keep them short, but descriptive.
- Commit messages will help you and others understand what has been done and why. All the commits of a project will read like the history of the project.

Issues

- An issue is a note on a repository about something that needs attention.
- It could be a note about a bug that needs fixing, a feature request, a question, or anything else.

Pull Requests

- Pull Requests are the heart of collaboration on GitHub. When you make a pull request you are asking the owner of the repository to include your contribution. In other words, you are asking them to merge your work into their branch.

WHAT ARE WE USING?

Git

- The technology of version control

Github

- A company/website where you can store your code, and connect to using git

Github Desktop

- A GUI developed by the folks at Github to make the steep learning curve just a little bit easier! It's still quite early stages but we love it!

© CodeFirst Girls 2017

15

What are we using today?

Git - this is the technology of version control

Github - this is a website (and larger company) wherein you can store your code in their cloud online. You connect and control it using git protocols.

Github desktop - this is a GUI that has been developed by the folks at github to make the fairly steep learning curve a little bit easier.

Creating a repository

Create your group repository

Time to roll up our sleeves and get ready for some tricky stuff. We are going to create a repository for your website project. And then we're going to commit some stuff and publish the repository to GitHub.

ACTION

If any of the students have not yet confirmed their email address for their GitHub account then they need to do so now. As long as the email address is not confirmed any communication between the GitHub desktop app and GitHub online will run into problems.

IMPORTANT!

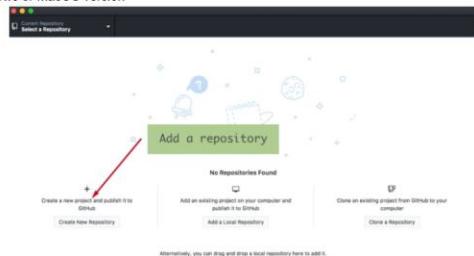
You can run this session either as a demo or codealong. Every student should use the project they started in week one so that they will all have a project published on GH pages by the end of the class. Every student must have got someway to achieving this before moving onto section two, collaboration.

Feel free to adjust this according to the level your class is at, however you must ensure that by the end of this session everyone has initialised, committed and published their website from week two, and started a new project for their group work.

Don't worry too much about the demo videos, they are included for reference for the students when working through the slides themselves.

TASK: Creating a repository

Go to
Download the Windows or MacOS version



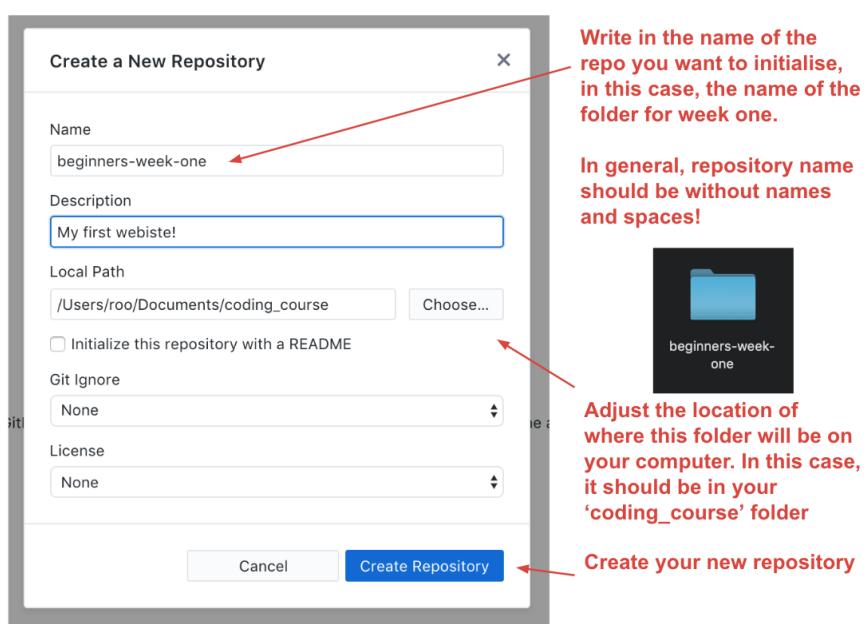
Either



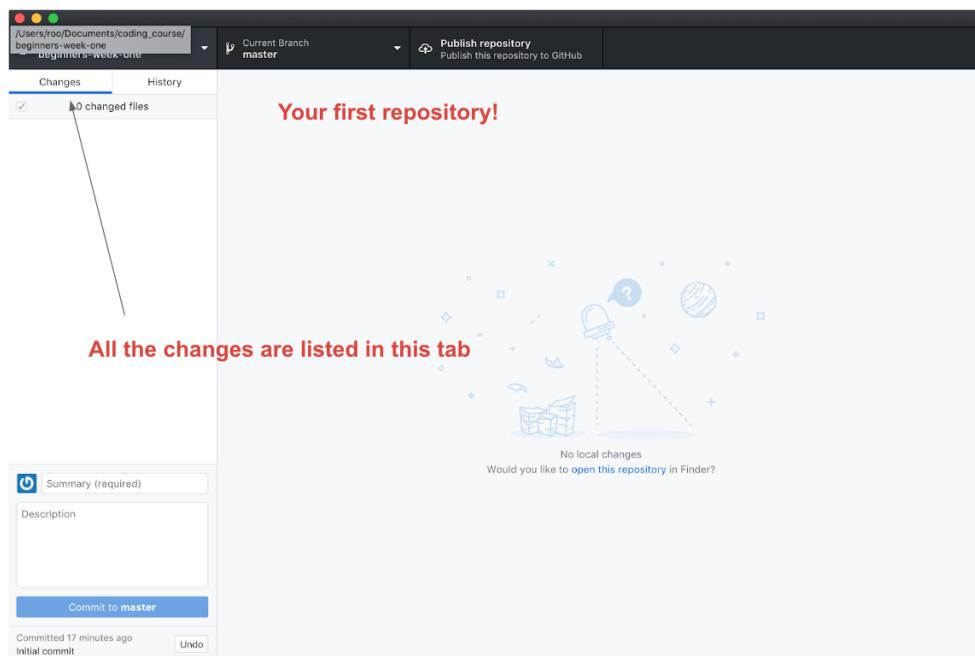
OR

16

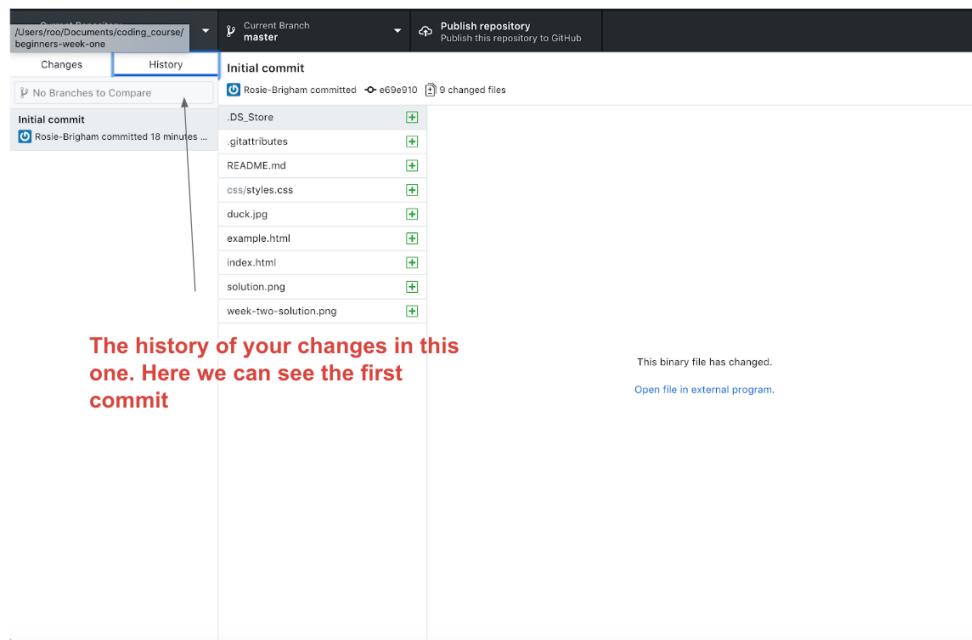
1. Open up the GitHub Desktop client on your computer
2. You should see something like this
3. Click on the **plus sign** in the top left hand corner of the screen



1. On the **Create** tab of the pop-up window type the name of your repository. In this case, because we are initialising a folder that is already there, we need to write in the folder name **exactly** as it is.
2. Check in the **Local Path** box that it has the correct path name for your week one project.
3. Click the **Create Repository** button.
4. The GitHub Desktop client will now create your folder at the location you wanted and, turn it into a git repository.



You now have a repository! You can look at all the changes you have made in the 'changes' tab



© CodeFirst:Girls 2017

20

And you can see the history of your changes in the 'history' tab.

First Commit

MAKE A SMALL CHANGE IN YOUR INDEX.HTML FILE

```
<!DOCTYPE html>
<html>
  <head>
    <title>First site</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
  </head>
  <body>
    <h1>Hello Universe!</h1>
    ...

```

© CodeFirst:Girls 2010

23

To make our first commit, we need to first make a small change to our project. Lets go over to the index.html file in atom and change 'Hello World' to 'Hello Universe'.

```

Current Repository: beginners-week-one
Current Branch: master
Publish repository: Publish this repository to GitHub

Changes (1) History index.html
1 changed file: index.html

index.html @@ -11,7 +11,7 @@
 11 11 </head>
 12 12 <body>
 13 13 <div>
 14 14 - <h1>Hello World</h1>
 15 15 + <h1>Hello Universe</h1>
 16 16 <h2>Welcome to my site!</h2>
 17 17

```

Any changes appear on the changes tab - additions in green, subtractions in red

```

Update index.html
Description

Commit to master
Committed 27 minutes ago Initial commit Undo

```

24

Here we can see this change in the changes tab in desktop. Any additions to the file will be highlighted in green, and subtractions will be highlighted in red.

```

Current Repository: beginners-week-one
Current Branch: master
Publish repository: Publish this repository to GitHub

Changes (1) History index.html
1 changed file: index.html

index.html @@ -11,7 +11,7 @@
 11 11 </head>
 12 12 <body>
 13 13 <div>
 14 14 - <h1>Hello World</h1>
 15 15 + <h1>Hello Universe</h1>
 16 16 <h2>Welcome to my site!</h2>
 17 17

Commit to master
updated hello world to hello universe
Committed 27 minutes ago Initial commit Undo

```

Select the box for the files that you want to commit here

Write your commit message and description here, then the commit button.

25

1. Tick the check box for the files that you want to commit (here we only have one line change in one file) but you can have multiple

2. Then write a commit message and a description for the commit (if you want) and press the commit button

Make your commit messages descriptive, as when you look at the history of commits it will help you understand what you did. ('added a thing' is never helpful a month on when trying to find out where a bug may have been introduced!) Some developers also recommend to write them in the present tense. This makes it easier to read when you go through the history of a project later.

The screenshot shows a GitHub commit history interface. At the top, there are tabs for 'Changes' and 'History'. The 'History' tab is selected, indicated by a blue underline. Below the tabs, a message says 'No Branches to Compare'. A single commit is listed under the heading 'updated hello world':

updated hello world
Rosie-Brigham committed just now
Initial commit
Rosie-Brigham committed 29 minutes ago

A red arrow points from the text 'Once committed, you can see it in the history tab' down to the commit message 'updated hello world'.

Below the commit message, the commit details are shown:

updated hello world to hello universe
Rosie-Brigham committed 29 minutes ago

index.html

11	11	@@ -11,7 +11,7 @@
12	12	</head>
13	13	<body>
14	14	<div>
	-	- <h1>Hello World</h1>
15	14	+ <h1>Hello Universe!</h1>
16	15	<h2>Welcome to my site!</h2>
17	17	

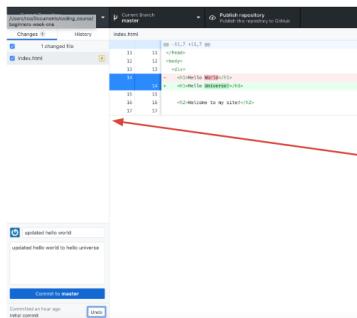
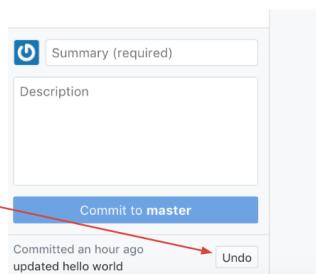
Once committed, you can see it in the history tab

You will then be able to see the commit in the 'history' tab!

Undoing a commit

If you ever need to undo a commit simply press the undo button.

This only works if you have *not* pushed it up to github (more on this on the next slide)



Here you can see there is once again only one commit on the history tab, and the changes that were committed are back on the changes tab.

29

One of the benefits of using a version control system is that you can go back to a previous version of your project if you need to. This can be because of a bug that needs fixing, or maybe you made a mess of a file when trying something clever, or maybe the client you are working for decides that all the work you did in the past month is not what he wanted after all.

Whatever the reason, you'll be glad to be using version control when you are in a situation like that. There are two ways of reverting to an older version. Let's start with the easy way.

If you have just done a commit and realised you didn't want to do it, or maybe you didn't include all the files you needed to (or included the wrong ones) then undoing it is simple. Simply press the 'undo' button! When you click the Undo button it will put your repository back in the state it was just before you did the commit. Meaning that your changes will still be there, but they are listed as being uncommitted changes. Make whatever you need to do and then commit as usual.

However, if you have already pushed the changes up to github 'undo' will get you in a muddle and you will need to 'revert' it instead. More on that later...

Publishing to github

What we have done so far is initialising a current project with git on your local machine. We need to publish it to our GitHub account. That's what we'll do next.

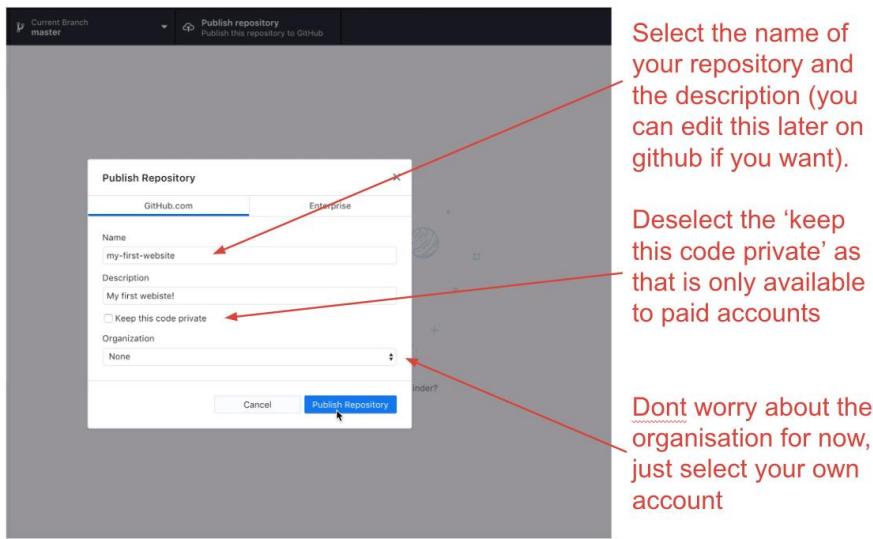
The screenshot shows a code editor interface with a diff view for the file 'index.html'. The left pane shows the commit history with one change: 'updated hello world to hello universe'. The right pane shows the diff details, specifically line 14 where the text 'Hello World' has been replaced by 'Hello Universe'. At the top right of the editor, there is a 'Publish repository' button with the sub-label 'Publish this repository to GitHub'.

To publish a repository on github
simply press the 'publish
repository' button



31

Simply click on the **Publish** button in the top right corner of the screen.



© CodeFirst:Girls 2017

32

In the pop-up window you should see the name of your repository and your Github account name. If you want to you can write a description for your repository.

Click the **Publish Repository** button. You should now get a message saying it is publishing to GitHub.

Overview **Repositories** 85 Stars 8 Followers 19 Following 4

Find a repository... Type: All ▾ Language: All ▾ New

my-first-website My first website! ★ Star
HTML Updated 9 minutes ago

monumental_server basic server backend thing for monumental work ★ Star
Ruby Updated 10 days ago

playing-with-desktop delete me soon ★ Star
Updated on 14 Nov

Organizations

- DEV
- IMPACT HACK
- ...
- ...
- ...

Then head over to your GitHub profile to see your new project!

1. Go to your account on <https://github.com>
2. Click on the arrow next to your profile image in the top right corner and select Your profile from the drop down menu.
3. Then click on the Repositories tab and you should see your new repository at the top of the page.

Rosie-Brigham / my-first-website

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

My first website! Edit

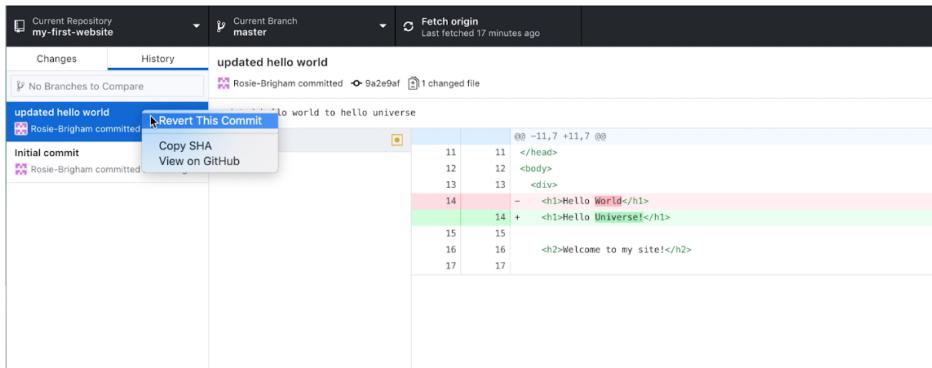
Manage topics

2 commits	1 branch	0 releases	0 contributors																											
Branch: master New pull request	Create new file Upload files Find file Clone or download ▾																													
<p>Rosie-Brigham updated hello world ... Latest commit 9a2e9af 12 minutes ago</p> <table border="1"> <tbody> <tr> <td>css</td> <td>Initial commit</td> <td>an hour ago</td> </tr> <tr> <td>.DS_Store</td> <td>Initial commit</td> <td>an hour ago</td> </tr> <tr> <td>.gitattributes</td> <td>Initial commit</td> <td>an hour ago</td> </tr> <tr> <td>README.md</td> <td>Initial commit</td> <td>an hour ago</td> </tr> <tr> <td>duck.jpg</td> <td>Initial commit</td> <td>an hour ago</td> </tr> <tr> <td>example.html</td> <td>Initial commit</td> <td>an hour ago</td> </tr> <tr> <td>index.html</td> <td>updated hello world</td> <td>12 minutes ago</td> </tr> <tr> <td>solution.png</td> <td>Initial commit</td> <td>an hour ago</td> </tr> <tr> <td>week-two-solution.png</td> <td>Initial commit</td> <td>an hour ago</td> </tr> </tbody> </table>				css	Initial commit	an hour ago	.DS_Store	Initial commit	an hour ago	.gitattributes	Initial commit	an hour ago	README.md	Initial commit	an hour ago	duck.jpg	Initial commit	an hour ago	example.html	Initial commit	an hour ago	index.html	updated hello world	12 minutes ago	solution.png	Initial commit	an hour ago	week-two-solution.png	Initial commit	an hour ago
css	Initial commit	an hour ago																												
.DS_Store	Initial commit	an hour ago																												
.gitattributes	Initial commit	an hour ago																												
README.md	Initial commit	an hour ago																												
duck.jpg	Initial commit	an hour ago																												
example.html	Initial commit	an hour ago																												
index.html	updated hello world	12 minutes ago																												
solution.png	Initial commit	an hour ago																												
week-two-solution.png	Initial commit	an hour ago																												

All the files in your repository are listed here, together with the last commit message for each file

If you click on the repository name you will see the files in the repository. For each file the message of the last commit for the file will be shown too.

Reverting a commit



The screenshot shows a GitHub repository named 'my-first-website'. The 'History' tab is selected, showing a commit from 'Rosie-Brigham' titled 'updated hello world' (commit 9a2e9af). Below it is an 'Initial commit' by the same user. A context menu is open over the 'updated hello world' commit, with the option 'Revert This Commit' highlighted. The diff view shows changes made to 'index.html': line 14 was updated from '<h1>Hello World</h1>' to '<h1>Hello Universe</h1>'. Line 16 was added with the text '<h2>Welcome to my site!</h2>'.

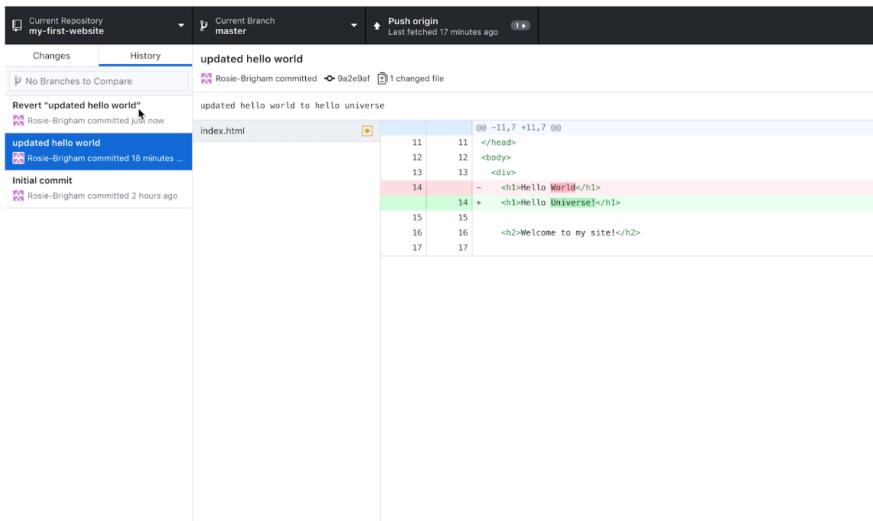
If you have already pushed a commit up to github, you cannot simply 'undo' it, instead you need to revert it. Simply right click on the commit and select revert

Use this, instead of the 'undo' function to revert a commit in the long ago past, to undo something that you have already published and pushed up to github

© CodeFirst:Girls 2017

36

When you have pushed up a commit you cannot simply 'undo' it, as it will mess up the history between your local and remote versions. Therefore, you will have to 'revert' it. This will create a new commit which reverses the work you did in the commit you want to undo. To do this, simply right click on the commit and select 'revert'



The screenshot shows a GitHub repository named 'my-first-website'. The 'History' tab is selected, showing a commit from 'Rosie-Brigham' titled 'Revert "updated hello world"' (commit 9a2e9af). Below it is a commit from the same user titled 'updated hello world' (commit 9a2e9af). A context menu is open over the 'Revert "updated hello world"' commit, with the option 'Revert "updated hello world"' highlighted. The diff view shows changes made to 'index.html': line 14 was updated from '<h1>Hello Universe</h1>' back to '<h1>Hello World</h1>'. Line 16 was removed with the text '<h2>Welcome to my site!</h2>'.

This will create a new commit, that reverts the work back to the original!

© CodeFirst:Girls 2017

37

As you can see, this creates a new 'revert commit', make sure you publish this by pressing the 'push origin' button.

Creating a branch

CREATING A BRANCH

- The first branch in any repository is master
- You can create a branch from any other branch you are in
- A branch is (initially) a copy of the current branch you are in
- Branches are useful for testing out features

© CodeFirst.Girls 2017

33

Up until now we have been using a version control system to commit our changes and to fix commits we should not have committed. But as we saw earlier, using a version control system is also a great way of adding new features in isolation to a project without jeopardising the current code.

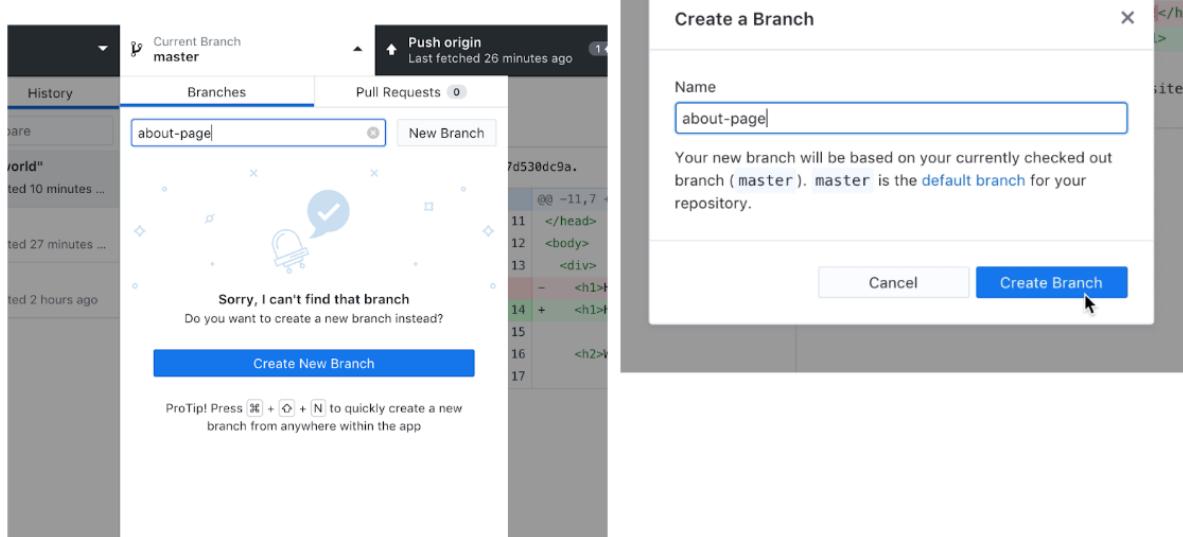
To do this you need to create a separate branch on your project.

Remember when you create a git repository it will also create your first branch with the name master. If you want to add a new feature to your project then you first create a different branch, do all your work, test it out thoroughly and then merge the feature branch back into the master branch.

When you create a new branch, git will take a copy of the current branch you are in, usually the master branch. This new branch will sit alongside the main branch until you are ready to merge it back into the mainstream.

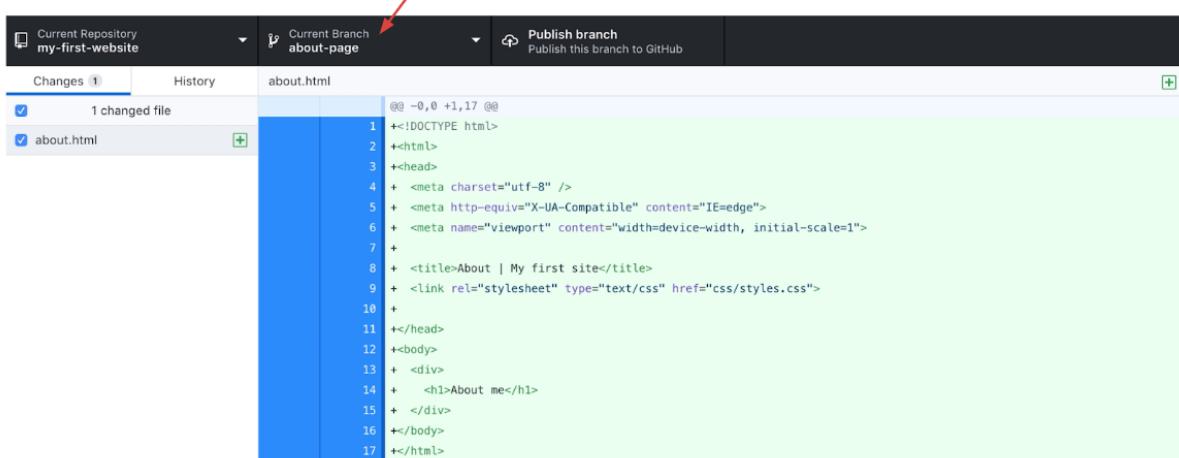
Let's create a branch on our repository.

Click on the branch tab to move onto a branch, or create (and then move onto) a new one



1. First remember to pull or push any changes up before you add a branch
2. Click on the 'branch' tab, you can type in a branch name to search for it. If there is no branch found, as in this case, you can simply create a new one.

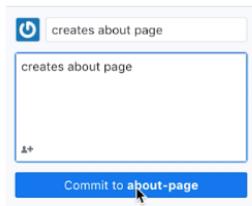
Remember to check out what branch you are on here



A screenshot of a GitHub commit interface. At the top, there are three tabs: 'Current Repository my-first-website', 'Current Branch about-page' (which has a red arrow pointing to it), and 'Publish branch'. Below these tabs, there are two sections: 'Changes' (with 1 changed file) and 'History'. Under 'Changes', a file named 'about.html' is listed. The main area shows the diff of the 'about.html' file:

```
@@ -0,0 +1,17 @@
1 +<!DOCTYPE html>
2 +<html>
3 +<head>
4 +<meta charset="utf-8" />
5 +<meta http-equiv="X-UA-Compatible" content="IE=edge" />
6 +<meta name="viewport" content="width=device-width, initial-scale=1" />
7 +
8 +<title>About | My first site</title>
9 +<link rel="stylesheet" type="text/css" href="css/styles.css" />
10 +
11 +</head>
12 +<body>
13 +<div>
14 +<h1>About me</h1>
15 +</div>
16 +</body>
17 +</html>
```

Make sure you are committing to the correct branch



© CodeFirst:Girls 2017

43

Here we have added a new page, about.html.

You can check what branch you are in on the branch tab, and when committing new changes you will see that the button tells you what branch you are committing to.

Let's commit this new page onto the about page branch.

```

about.html
@@ -0,0 +1,17 @@
1 +<!DOCTYPE html>
2 +<html>
3 +<head>
4 + <meta charset="utf-8" />
5 + <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 + <meta name="viewport" content="width=device-width, initial-scale=1">
7 +
8 + <title>About | My first site</title>
9 + <link rel="stylesheet" type="text/css" href="css/styles.css">
10 +
11 +</head>
12 +<body>
13 + <div>
14 +   <h1>About me</h1>
15 + </div>
16 +</body>
17 +</html>

```

The history on this branch is now different to the master branch. Press the **publish** button to get the new branch, and your changes, up on github.

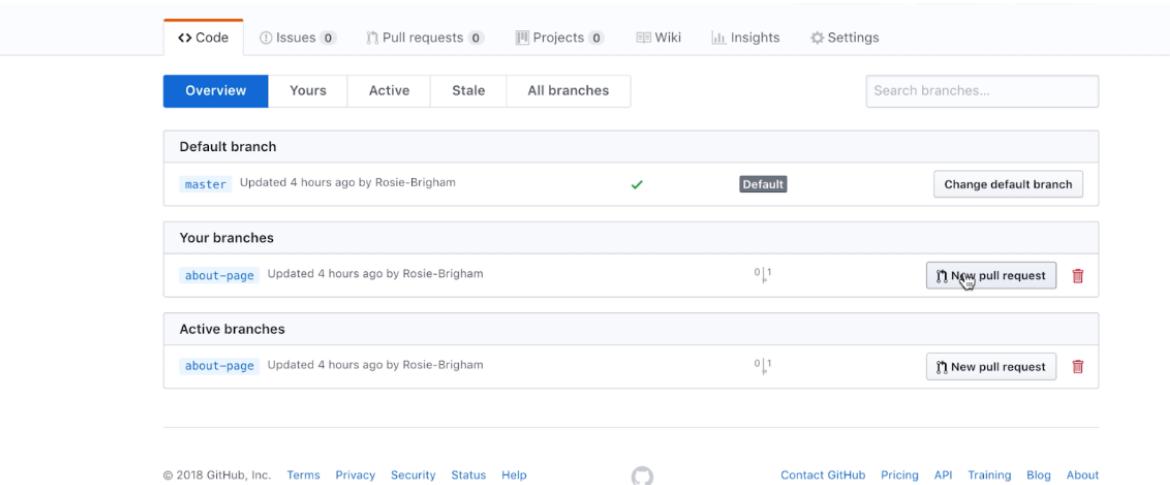
© CodeFirst:Girls 2017

44

Here we can see the commit made onto the about page branch. The history on the about-page branch is now different to that on master. Let's push the **publish** button to push our changes, and the new branch, onto github.
But how do we get these changes back onto the master branch? We do this using **pull requests**.

Pull requests

When you have pushed your new branch with your changes up to github, you can merge them into master using a pull request. On the ‘branches’ tab in your repo, select ‘new pull request’ on the branch you want to merge changes in from



The screenshot shows the GitHub branches page. At the top, there are navigation links: Code, Issues 0, Pull requests 0, Projects 0, Wiki, Insights, and Settings. Below these are tabs: Overview (which is selected), Yours, Active, Stale, and All branches. A search bar labeled 'Search branches...' is also present.

The main content area is divided into sections:

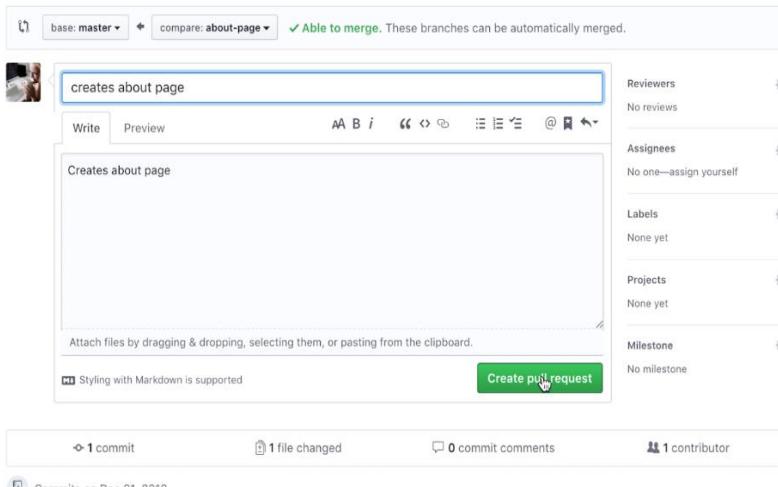
- Default branch:** Shows the 'master' branch, updated 4 hours ago by Rosie-Brigham. It has a green checkmark icon and a 'Default' button.
- Your branches:** Shows the 'about-page' branch, updated 4 hours ago by Rosie-Brigham. It has a 'New pull request' button and a trash bin icon.
- Active branches:** Shows the 'about-page' branch again, updated 4 hours ago by Rosie-Brigham. It has a 'New pull request' button and a trash bin icon.

© CodeFirst:Girls 2017

47

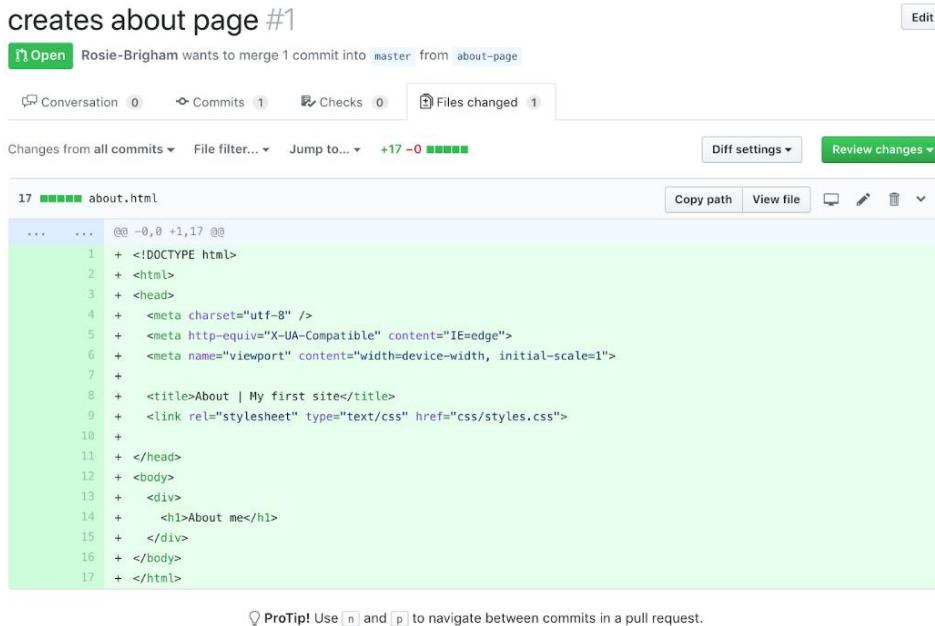
Go to your repo on github and navigate to the ‘branches’ tab. Here you will be able to see a list of all the branches that you have pushed to github. If any of the branches histories are different to that of master, there will be a ‘new pull request’ button. Click on this for our about page to create a new pull request.

Add a title and description for your pull request to describe the changes you've made



Here you will be able to review all the changes you have made, alongside adding a title and description to your pull request. For large complicated ones, these are important so that someone who is reviewing your code will know exactly what changes you are making and why.

In the pull request you can then review the changes you have made, and add comments to things that maybe need to change



Once created, you, or a colleague, can review the changes you have made. At this point, if there are any things you think that needs to be changed before it's ready to be merged in, you can comment and request for changes.

If everyone looks good in the pull request, simply press the merge button and then confirm!

creates about page #1

A screenshot of a GitHub pull request page for a repository named "Rosie-Brigham/my-first-website". The pull request is titled "Creates about page" and has one commit. The status bar at the top shows "Conversation 0", "Commits 1", "Checks 0", "Files changed 1", and a green progress bar with "+17 -0". On the right side, there are sections for "Reviewers" (No reviews), "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Notifications" (Unsubscribe). A red arrow points to the "Merge pull request" button, which is highlighted with a green border.

If everything looks rosy, simply press merge to merge your changes in!

All merged! Make sure you delete the branch after you have merged the pull request to keep your repo tidy

creates about page #1

A screenshot of a GitHub pull request page for the same repository. The status bar at the top now shows "Merged". The pull request details remain the same. On the right side, there is a "Delete branch" button next to the "Pull request successfully merged and closed" message. The "Leave a comment" section at the bottom is visible.

And voala! All merged in. It's best practice to delete your branch to keep your repo nice and tidy.

You can see the new commits from the merged branch, and the commit created by the merge, on the history of the master branch

The screenshot shows a GitHub repository page for 'Rosie-Brigham / my-first-website'. The 'Code' tab is selected. The 'Branch: master' dropdown is set to 'master'. The commit history for December 21, 2018, is displayed:

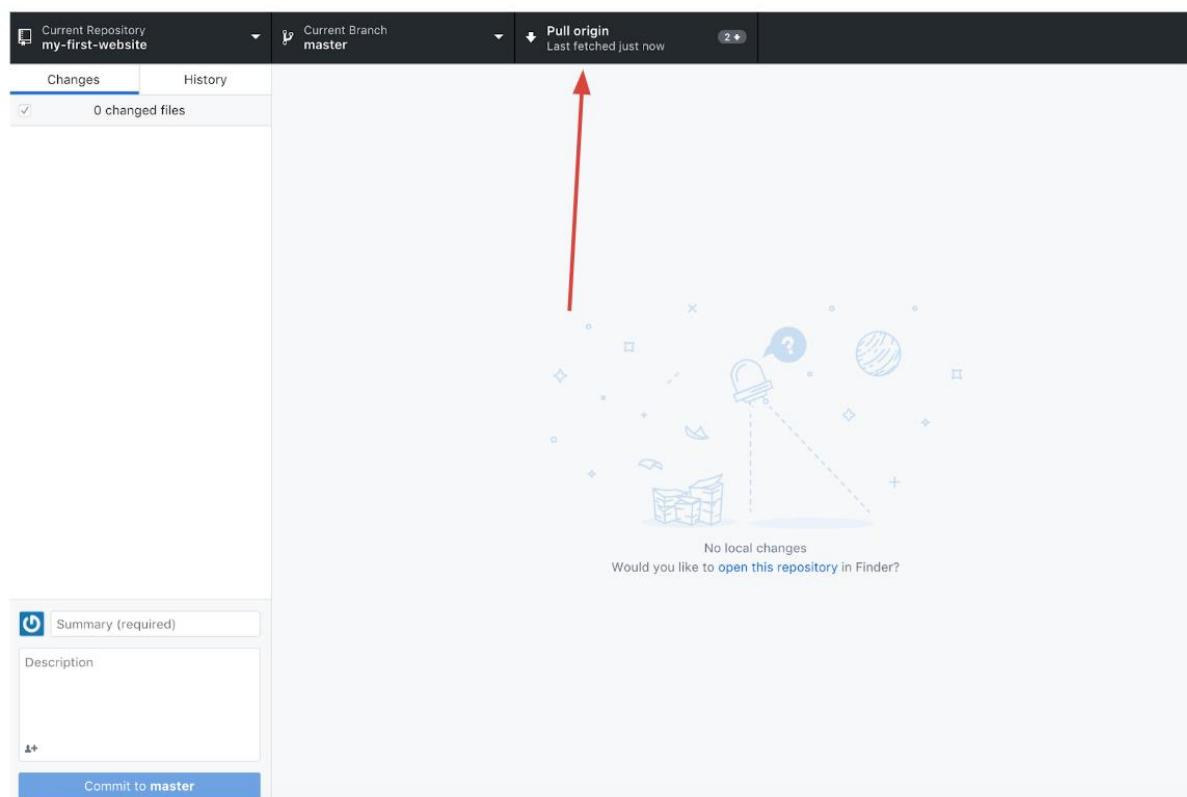
- Merge pull request #1 from Rosie-Brigham/about-page ...
Rosie-Brigham committed 13 seconds ago ✓ Verified f874583
- creates about page ...
Rosie-Brigham committed 4 hours ago 0001989
- Revert "updated hello world" ...
Rosie-Brigham committed 4 hours ago ✓ 33ba101
- updated hello world ...
Rosie-Brigham committed 4 hours ago ✓ 9a2e9af
- Initial commit
Rosie-Brigham committed 6 hours ago b3740bf

At the bottom, there are 'Newer' and 'Older' buttons.

Here we can see the history of the master branch, including the commit made on the about-page branch that has now been merged in, alongside a merge commit.

A little bit of good advice

Whenever you have things on your remote repo that is not on your local one (such as merged branches) it is crucial that you pull them down. You can do this by pressing the 'pull origin' button.



It is really easy to make a right mess of things when working with repositories. In a moment we'll do this deliberately so that you can learn how to resolve conflicts. But the best way of working with git is to avoid conflicts all together.

So the first rule of git is as follows:

Always pull down new changes

The screenshot shows the GitHub interface for a repository named 'my-first-website'. The 'Changes' tab is selected, showing a merge commit from 'Rosie-Brigham/about-page' into 'master'. The commit message is 'Merge pull request #1 from Rosie-Brigham/about-page' and it was committed by 'Rosie-Brigham' a minute ago. The commit creates an 'about.html' file. The diff view shows the content of the 'about.html' file:

```
@@ -0,0 +1,17 @@
1 <!DOCTYPE html>
2 <html>
3 <head>
4 + <meta charset="utf-8" />
5 + <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 + <meta name="viewport" content="width=device-width, initial-scale=1">
7 +
8 + <title>About | My first site</title>
9 + <link rel="stylesheet" type="text/css" href="css/styles.css">
10 +
11 </head>
12 <body>
13 + <div>
14 +   <h1>About me</h1>
15 + </div>
16 </body>
17 </html>
```

Now you have the commit created on github by merging the about branch into master on your local machine

Now we can see the merge commit, created on your remote repository, pulled down onto your local machine.

Publishing to github pages

PUBLISHING ON GITHUB PAGES

- Free web hosting for git repositories on GitHub
- Make sure `index.html` is all lowercase
- Your website url will be
`[your-github-username].github.io/[repository-name]`

Publishing on GitHub Pages

Git and GitHub isn't only about version control. Another great feature that GitHub has is free hosting for your website projects. The hosting can only be used for repositories stored on GitHub accounts, and is only suitable for static websites where there is no requirement for a database. Despite that, it's a good way of having a website online at no extra cost.

How Github Pages works

In order to publish your website through GitHub pages you will need to create a branch in your repository called `gh-pages`. GitHub will look for this branch and use the contents of this repository to create a website from .

Any code you want to show on your website will need to be added to this branch. If it isn't on this branch you will not see it on your online website. In other words, if you for example added a page to your website on the master branch, and you haven't merged your master branch into your `gh-pages` branch, then your online website will not have this extra page.

Once you have published your website to GitHub Pages your site will show up at **[your-github-username].github.io/[repository-name]**

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

The screenshot shows the 'Source' section of a GitHub repository settings page. A modal window titled 'Select source' is open, showing three options:

- master branch** (selected): "Use the master branch for GitHub Pages." A note below says "You'll theme using the master branch. [Learn more.](#)"
- master branch /docs folder**: "Use only the /docs folder for GitHub Pages."
- None** (unchecked): "Disable GitHub Pages."

Buttons for 'None' (disabled), 'Save', and a close button are visible.

On the settings page of your repo, scroll down to Github Pages and enable it.

Select it to build from your master branch

On the settings page of your repo on github, select the 'settings' tab and scroll down to the 'github pages' section and select 'enable'. Select the master branch to be the branch that it is built from.

GitHub Pages

This will be the URL of the project

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is ready to be published at <https://rosie-brigham.github.io/my-first-website/>.

Source

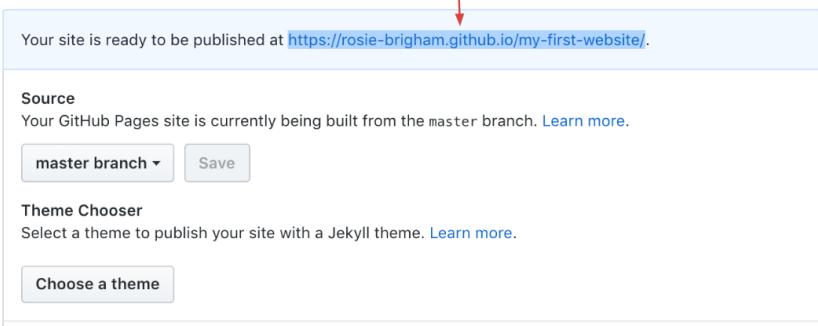
Your GitHub Pages site is currently being built from the master branch. [Learn more](#).

master branch ▾ Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme. [Learn more](#).

Choose a theme



After you have done that, and if there are no bugs in your code, it will give you the url that your site will be published on. **Note:** it will not automatically deploy the site, but will do after you push up your next commit. So you will need to create a new commit in order to see your site on the given URL.

Part 2 - Collaborating

Note - by this point, every student should have published their project on github, if they have not, allow some time for them to go through the slides at their own pace and do that before we move onto collaborating

COLLABORATING

So far you have initialised your first site with git, made couple of commits on two different branches, pushed those commits to github and even published your site on github pages!

Now lets step things up a notch, and use github to collaborate with the team projects you started last week

You can use a version control system on your own to track the changes you make to your projects. And in that case you will be the only person contributing to the repository. But version control is at its most powerful when a project is being worked on by more than one person. And that's what we are going to do now. We are going to add a collaborator to your repository.

Find the people in your team in order to create your group project repo on github. Only one of you needs to do this and this person will become the owner of the repository. The other person will be the collaborator.

Task:

1. Find your teammates and sit with them for the rest of the session. Choose one person from your team to lead the first section.
2. If you have already started coding your project, initialise this in a new repository on github desktop.
3. If you haven't started, create a new folder in your coding_course folder called 'group_project' and create a file called 'index.html'. Initialise this as a new repository on github desktop following the instructions earlier
4. Commit your work and publish it to github

ADDING COLLABORATORS

- You should now have your group project pushed up to github (don't worry if it's only an empty file for now, you can add to it later on)
- We are now going to add your other team-mates as collaborators

Everyone should now have a group project on someone's laptop, and pushed up to github.

Now we are going to add everyone else in the group as collaborators so that you can all add to the project

The screenshot shows the GitHub repository settings page for 'CodeFirstGirls / group_project'. The 'Settings' tab is selected. On the left, a sidebar lists various repository options. A red arrow points from the text 'Go to Collaborators' to the 'Collaborators & teams' button in the sidebar. Another red arrow points from the text 'Select the Settings tab for your repository' to the 'Settings' tab in the top navigation bar. A third red arrow points from the text 'Type the username of your teammate and select their account' to the search input field where 'moopandpoom' is typed.

CodeFirstGirls / group_project

Watch 6 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Insights Settings

Options

Collaborators & teams

Branches

Webhooks

Integrations & services

Deploy keys

Alerts

Moderation

Interaction limits

Teams

No teams have been given access to this repository yet. Use the form below to add a team.

Add a team + Create new team

Collaborators

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address
You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

moopandpoom

Add collaborator

Select the Settings tab for your repository

Go to Collaborators

Type the username of your teammate and select their account

© CodeFirstGirls 2017

64

1. In the 'settings' tab of the new project on github, select the 'collaborators and teams' button and add each team mate.
2. GitHub might ask you to confirm your password at this point.
3. You will need to search for them by email or github username. Once added, this will send them an invitation to their email account that they will then need to accept.

The invitation has been sent

1. Your Collaborators page will now show you your teammates account and telling you that it is awaiting a response.

Accept the invitation → Accept invitation Decline

Owners of group_project will be able to see:

- Your public profile information
- Certain activity within this repository
- Country of request origin
- Your access level for this repository
- Your IP address

Is this user sending spam or malicious content?
[Block Rosie-Brigham](#)

2. Meanwhile GitHub will have sent an email to your teammate asking them to view the invitation.
3. If you are the collaborator teammate, click the **View invitation** button. This will open up your GitHub account and you will see a page asking you to accept the invitation. Click the **Accept invitation** button.
4. You will now be taken to the repository page. Beware that even though you are now a collaborator on this project, the repository does not appear on your

GitHub account online. Repositories remain attached to the account of the creator.

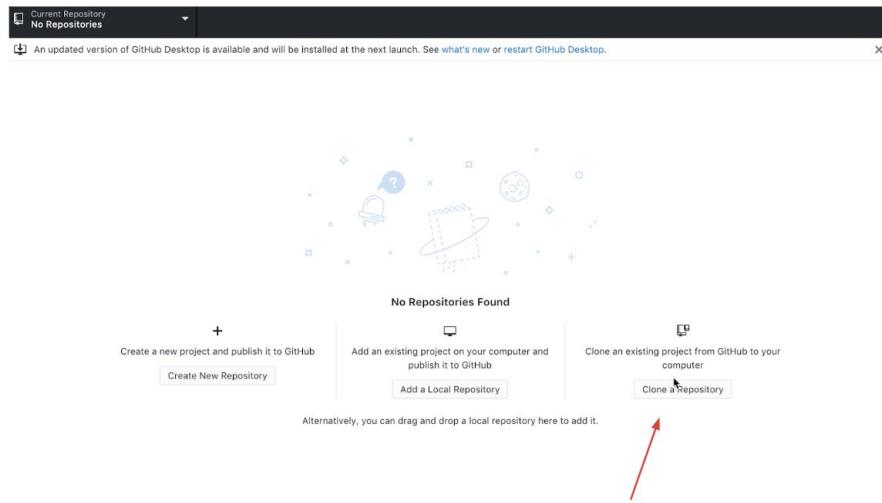
You now have push access!

The screenshot shows a GitHub repository page for 'CodeFirstGirls / group_project'. A blue banner at the top says 'You now have push access to the CodeFirstGirls/group_project repository.' Below the banner, the repository name is displayed with a fork icon. To the right are buttons for 'Watch' (6), 'Star' (0), and 'Fork' (1). A navigation bar below the repository name includes tabs for 'Code' (selected), 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', and 'Insights'. The main content area is titled 'demo group project'. It shows summary statistics: 2 commits, 1 branch, 0 releases, and 0 contributors. A dropdown menu shows 'Branch: master'. Buttons for 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download' are available. A list of commits is shown, starting with 'Rosie-Brigham adds index' (latest commit, 8 minutes ago) and 'Initial commit' (10 minutes ago). A note at the bottom encourages adding a README, with a 'Add a README' button.

Once you have accepted, you will now have push access to the repo! You can now clone it down onto your desktop.

TASK: make sure everyone in the group is added as a collaborator

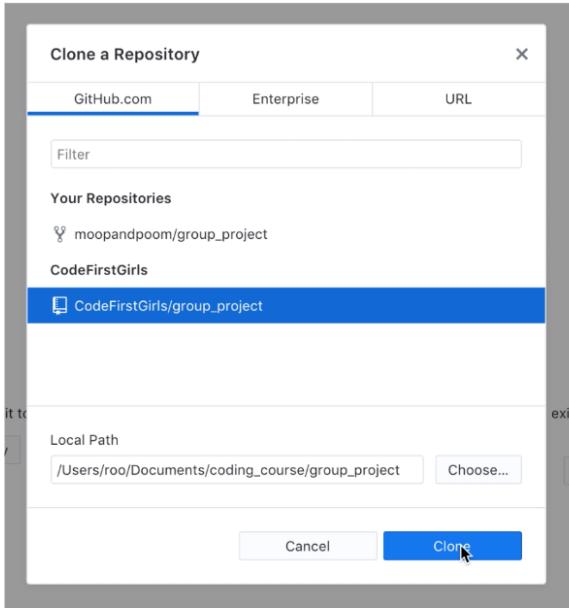
Cloning a repository



Select the 'Clone a Repository' to download a repository from github down to your machine

Now that everyone is a collaborator, you will now all have to clone the project onto your machine. This is slightly different to how we did it the first time as now we are simply downloading a current project onto your machine, instead of initializing a project that is already on your machine with git.

Select the 'clone a repository' option and select the repository you want to download



Select the repository you want to download

Make sure you put it in your coding_course folder. Once the cloning is complete you will now see your newly cloned repository in your GitHub Desktop client. As a collaborator you can now make some changes in this repository.

TASKS

Take it in turns, working together, to do the following:

1. Check out a new branch
2. Make a small change, commit it and push it up to github
3. Open a pull request on github
4. Get someone to merge in the pull request
5. Pull down the changes onto everyone's version

CONFLICT SCENARIO!

- Working on a repository with more than one collaborator can create conflicts
- Conflicts are a nuisance
- But they can be fixed!

Conflict scenario!

When you are working on repositories with more than one collaborator it is likely that at some point a conflict will arise. And I don't mean conflicts in terms of a disagreement between collaborators on an aspect of the project. But conflicts where one collaborator is working on an older version of the repository that has already been updated.

Remember the first rule of git? Always pull down changes first will avoid most conflicts, but sometimes they still slip through the net. The most likely time when conflicts will happen is when collaborators are working on the same repository at exactly the same time.

Let's set up a conflict scenario so that we can learn how to resolve it.

CONFLICT SCENARIO!

- In the following example, line 14 of index.html was changed in the **change-intro**, and in master *after* that branch was created. Github can't work out which one should have precedence so you have to tell it manually

In this scenario, a branch was created called 'change-intro' in which we have changed our intro paragraph. However, after the branch was created, that same line was changed on the master branch. As both branches have been changed on the same line, github cannot work out which one should have precedence, so it requires you to tell it manually. We can do this by opening a pull request.

You can still open a pull request with a merge conflict, but you have to manually fix the conflicting changes

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master ▾ compare: change-intro ▾ X Can't automatically merge. Don't worry, you can still create the pull request.

changes intro html tags

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Create pull request

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Here is the pull request to merge the 'change-intro' branch into the master. As you can see, we can still open the pull request even though it cannot be automatically merged.

changes intro html tags #2

Open Rosie-Brigham wants to merge 1 commit into master from change-intro

Conversation 0 Commits 1 Checks 0 Files changed 1 +2 -2

Rosie-Brigham commented just now

No description provided.

changes intro html tags 5ff5d5a

Add more commits by pushing to the **change-intro** branch on [Rosie-Brigham/my-first-website](#).

This branch has not been deployed

No deployments

This branch has conflicts that must be resolved

Resolve conflicts

Conflict files

index.html

Merge pull request You can also open this in [GitHub Desktop](#) or view [command line instructions](#).

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications

Unsubscribe

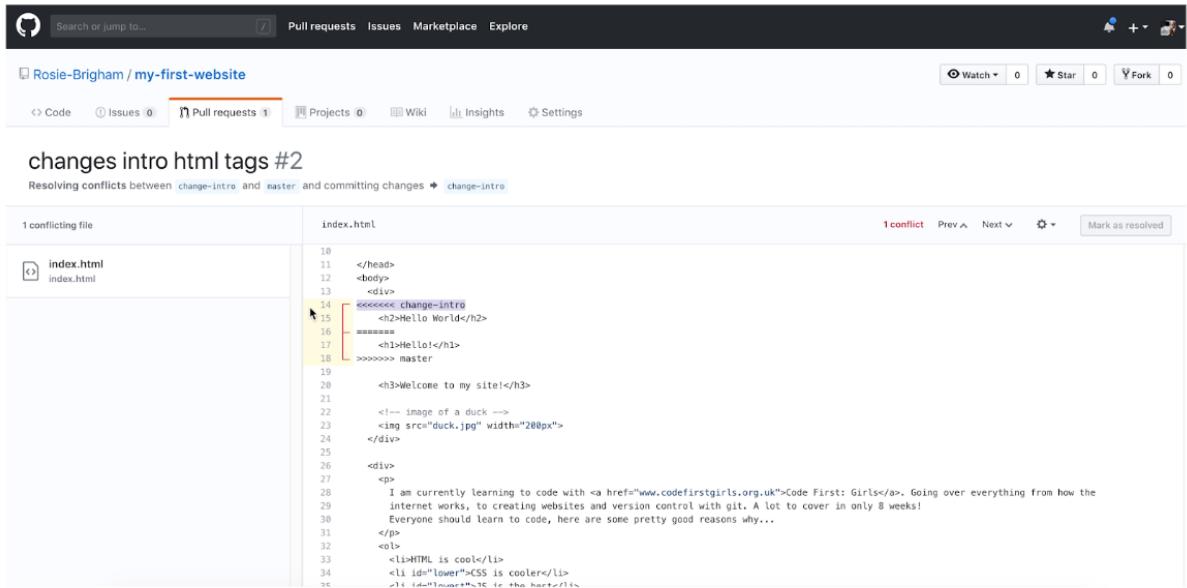
You're receiving notifications because you authored the thread.

1 participant

Click on the 'resolve conflicts' button

Handily, github gives us the option to 'resolve conflicts' manually and opens a web text editor. This is what we will do here.

This opens the web editor on github, simply delete the changes that you do not want and select ‘mark as resolved’



The screenshot shows the GitHub web editor interface for a file named 'index.html'. A conflict is indicated by yellow highlights and chevrons on line 14. The code on line 14 is:

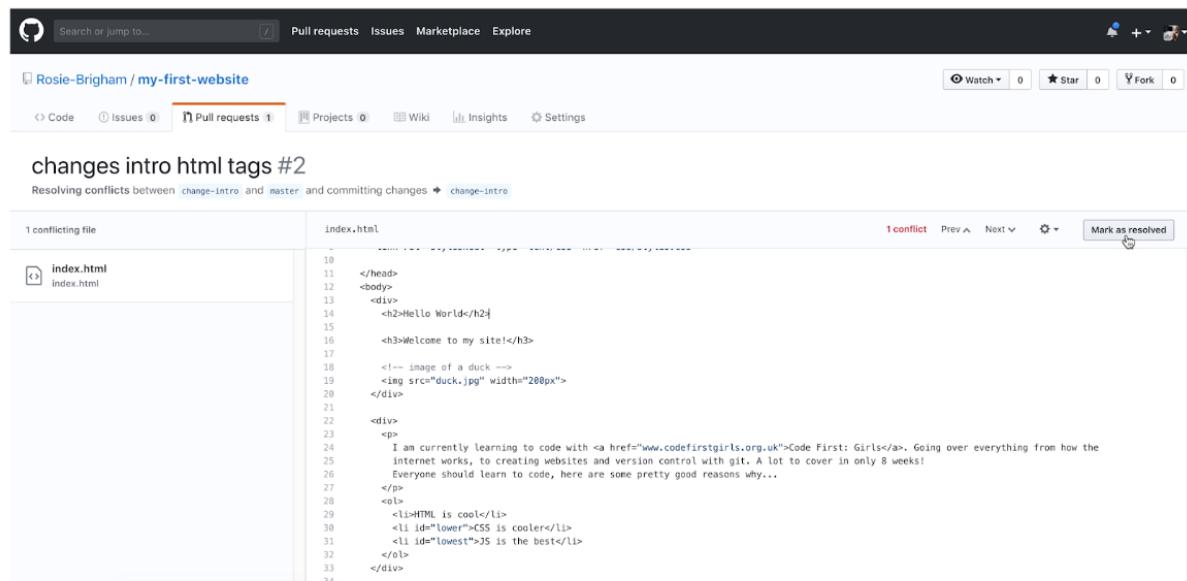
```
14 <><><> change-intro
15 <h2>Hello World</h2>
16 <=====>
17 <h1>Hello!</h1>
18 >>>>> master
```

The 'Mark as resolved' button is visible at the top right of the code editor.

© CodeFirst:Girls 2017

78

Simply delete the version of the line, or lines, that you do not want. Make sure you also delete the chevrons as well, otherwise they will appear in your code.



The screenshot shows the GitHub web editor interface for a file named 'index.html'. The conflict from the previous screenshot has been resolved. The code now looks like this:

```
10 </head>
11 <body>
12 <div>
13 <h2>Hello World</h2>
14 <h3>Welcome to my site!</h3>
15 <!-- image of a duck -->
16 
17 </div>
18 <div>
19 <p>
20 I am currently learning to code with <a href="www.codefirstgirls.org.uk">Code First: Girls</a>. Going over everything from how the
21 internet works, to creating websites and version control with git. A lot to cover in only 8 weeks!
22 Everyone should learn to code, here are some pretty good reasons why...
23 </p>
24 <ol>
25 <li>HTML is cool</li>
26 <li id="lower">CSS is cooler</li>
27 <li id="lowest">JS is the best</li>
28 </ol>
29 </div>
```

The 'Mark as resolved' button is visible at the top right of the code editor.

When you're done, hit the ‘mark as resolved’ button.

47

changes intro html tags #2

[Edit](#)

[Open](#) Rosie-Brigham wants to merge 2 commits into `master` from `change-intro`

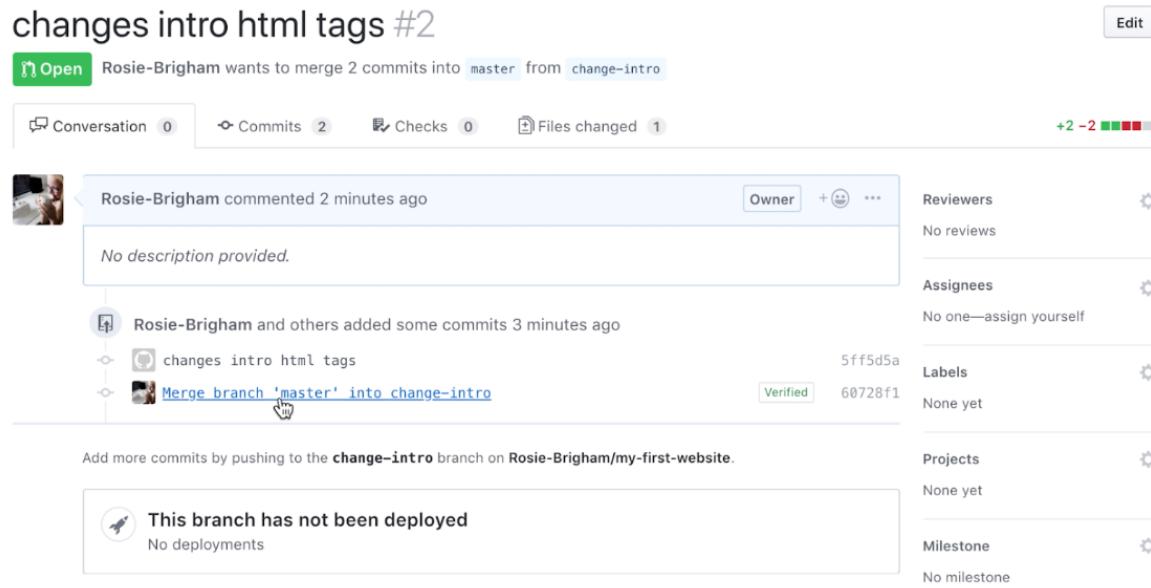
Conversation 0 Commits 2 Checks 0 Files changed 1 +2 -2 ■■■■■

Rosie-Brigham commented 2 minutes ago Owner +
No description provided.

Rosie-Brigham and others added some commits 3 minutes ago
changes intro html tags Merge branch 'master' into change-intro 5ff5d5a Verified 60728f1
Labels None yet

Add more commits by pushing to the `change-intro` branch on [Rosie-Brigham/my-first-website](#).

This branch has not been deployed Projects None yet
No deployments Milestone None yet



This adds a new merge commit - and the conflict is resolved!

This creates a new merge commit, and the conflict has been resolved!
Make sure you then pull this down onto your remote machine.

How to avoid and reduce merge conflicts

- Always Sync first
- Always end your work with Sync
- Plan with your collaborators who is going to work on what and when

How to avoid and reduce merge conflicts

Merge conflicts are not fun to deal with, but a few things can be done to avoid them.

1. Always sync first. Anytime you want to make a change to a shared repository make sure you sync first. This should also be the last thing you do so that you don't run into sync issues at a later date.
2. It's worth planning with your fellow collaborators which sections who is going to work on and when. This will avoid two people working on the same file at the same time.

HOMEWORK

1. Make sure both your group project and personal site is published on github, and your personal site is published on GH pages
2. Work on your group projects, using github to collaborate!

Homework for this week:

1. Make sure both your first website and group project is on github, and your first site is published on github pages
2. Work on your group projects, using git to collaborate!