Lab 1: Git and GitHub

This lab is designed to give you practice setting-up, editing, developing and maintain a GitHub repository. Only a small subset of GitHub features are covered in this lab. For this lab you will be required to setup a repository from scratch, and add various files for displaying a web page (e.g., html, style sheets and scripts), in addition to folders and comments (markdown). The initial exercises in the lab sheet give you examples that you can work through initially to get you started (e.g., you are invited to type in and experiment with using GitHub before you complete the assessed tasks). You can practice this lab in your own time before the scheduled assessed lab sessions. During the lab sessions, once you have completed the exercise on this sheet, show it to a lab helper who will mark it as completed. **This is worth 10% of the marks for this course**. There are two scheduled lab sessions for this topic. If you complete the exercises you don't need to attend the second lab session.

Course Material

- F28WP GitHub Page (https://f28wp.github.io)
- CANVAS (https://canvas.hw.ac.uk/)

Git, GitHub, and GitHub Pages

- Git Documentation
- Learn Git and GitHub in 15 minutes
- GitHub Pages Help
- GitHub Help
- GitHub Cheat Sheet
- GitHub Glossary
- GitHub For Academics

Jekyll

- Sites Using Jekyll
- Blog Migrations to Jekyll

Markdown

- Official Markdown Spec
- Printable Markdown Cheatsheet
- Markdown Cheatsheet
- GitHub Flavored Markdown

IMPORTANT: Each lab must be demonstrated in one of the allocated lab sessions (either virtually or face-to-face). After you have demonstrated your lab work you should submit a copy of your work on CANVAS as evidence (your mark is allocated/given for the demonstration session). The submitted copy, should be a short report and include, your name, date, which lab, who you demonstrated to, and any screenshots/code/links.

Overview

Part 1 (5%)

- Create GitHub Repository
- Add various file types (.html, .css, folders, images, …)
- Add a ‘ReadMe.md’ file and some comments/details about the repository (use the ‘markdown’ syntax)
- Create a ‘pages’ repository (i.e., read the pages.github.com details). A repository that has the same name as the username is able to host a static webpage (e.g., f28wp.github.io)
- Your GitHub repository/working website (i.e., github.pages.io)
- GitHub repository with multiple commits

Part 2, 3 (5%)

For this part, you need to be sure you’re able to demonstrate the following items:

- ReadMe.md file (detailing of your repository, headings, sections, …)
- Additional webpage content (includes .html and .css files/tags)
- Folders (style and scripts should be in sub folders)
- Jekyll (automatically generating content) on GitHub Pages
- Blog/Markdown posts

Part 4 (No Marks) - Advanced/Extra

This part gives no extra marks and are extra optional tasks to help you develop your skills.

- Multiple people work on the same repository/file (try to grab/push changes from other people)
- Managing ‘conflicts’ – multiple people change the same line (resolve the conflict and choose which lines to use for the final commit)
- Create a branch (separate version of the build to work on and then merge it back into the main build)

This lab provides a step-by-step introduction to help you get up and running with Git, GitHub, Pages and Jekyll in a short lab session. It assumes you know very little about version control, Git, and GitHub. It is helpful if you know the basics of HTML and CSS since you'll be working directly with these languages (however, it's not essential). You'll also be using a little bit of Markdown, but by no means do you need to be an expert with any of these languages for this lab. The idea is to learn by doing, so the code you'll be implementing in this lab is available in this guide. Recommended that you type in the examples and work through them in your own time so you understand what is happening (i.e., vs just copying and pasting).

Also to complement this lab, please read through the recommended resources/additional material/links at the start of document.

What is Git, GitHub, and GitHub Pages?

Git, GitHub, and GitHub Pages are all very closely related. Imagine Git as the workflow to get things done and GitHub and GitHub Pages as places to store the work you finish. Projects that use Git are stored publicly in GitHub and GitHub Pages, so in a very generalized way, Git is what you do locally on your own computer and GitHub is the place where all this gets stored publicly on a server.

1. Git

Git is a version control system that tracks changes to files in a project over time. It typically records what the changes were (what was added? what was removed from the file?), who made the changes, notes and comments about the changes by the changer, and at what time the changes were made. It is primarily used for software development projects which are often collaborative, so in this sense, it is a tool to help enable and improve collaboration. However, its collaborative nature has led it to gain interest in the publishing community as a tool to help in both authoring and editorial workflows (essential industry tool).

2. GitHub

GitHub is a web hosting service for the source code of software and web development projects (or other text based projects) that use Git. In many cases, most of the code is publicly available, enabling developers to easily investigate, collaborate, download, use, improve, and remix that code. The container for the code of a specific project is called a repository.

NOTE. Git is not GitHub!

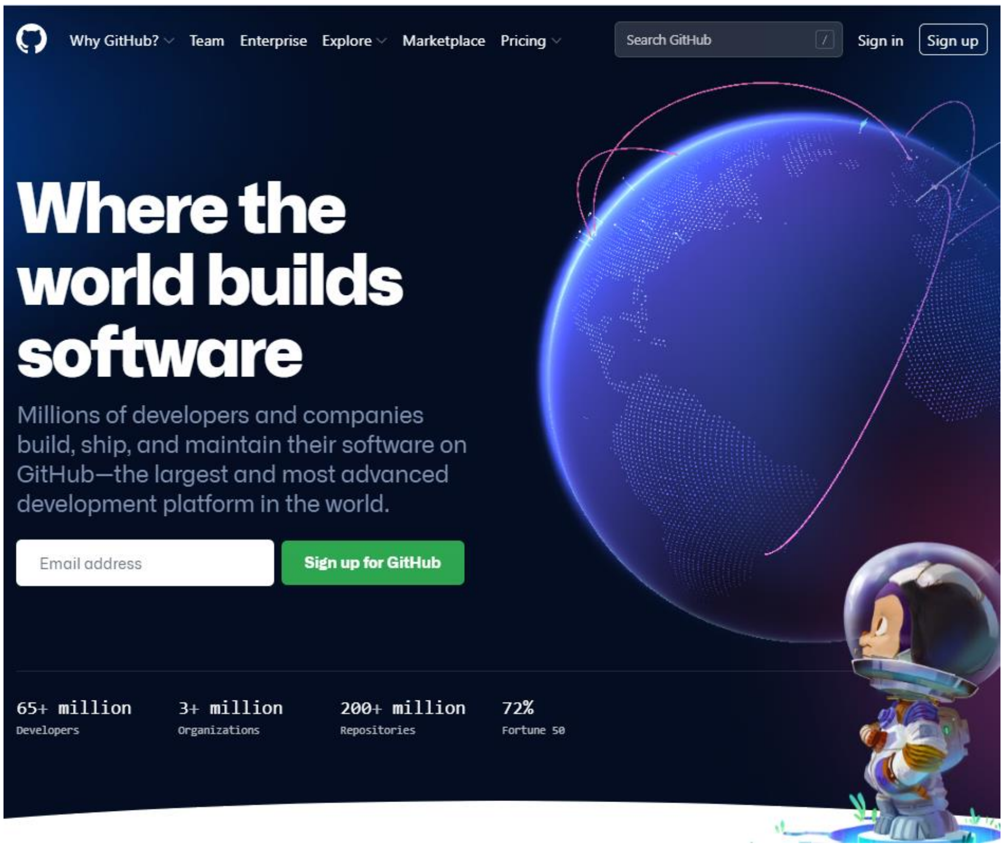
3. GitHub Pages

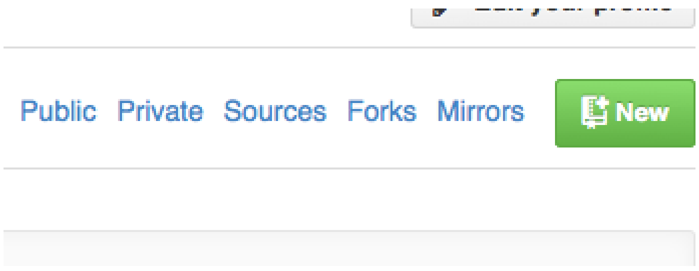
GitHub Pages are public webpages hosted for free through GitHub. GitHub users can create and host both personal websites (one allowed per user) and websites related to specific GitHub projects. Pages lets you do the same things as GitHub, but if the repository is named a certain way and files inside it are HTML or Markdown, you can view the file like any other website. GitHub Pages is the self-aware version of GitHub. Pages also comes with a powerful static site generator called Jekyll, which we'll learn more about later on.

Part 1 GitHub Pages

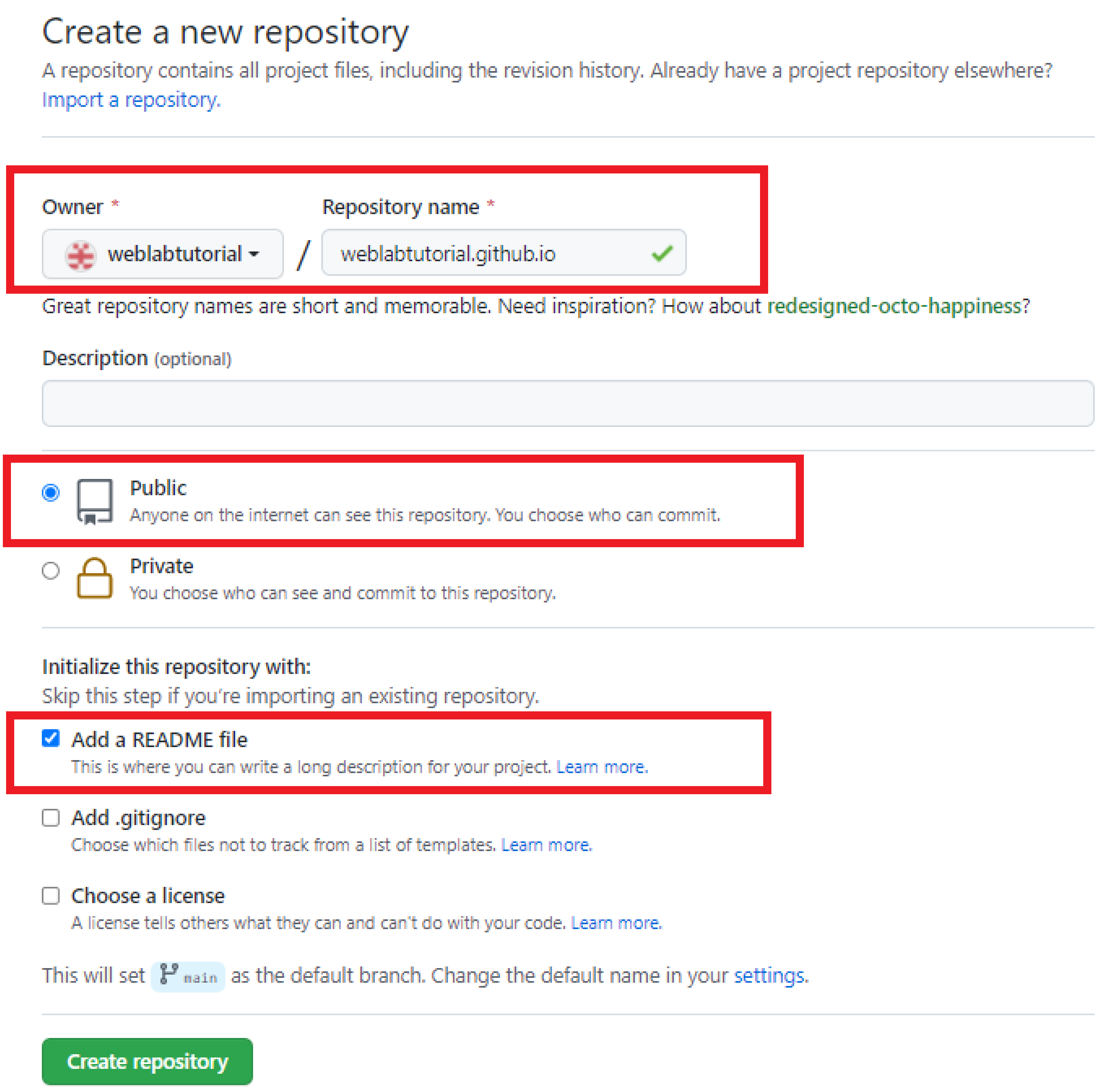
Don't worry if some of the concepts are a little unclear to you. The best way to learn this stuff is to just start doing the work, so let's not waste any more time and dive right in.

[Part 1- 1] Create your project's repository. Login to your GitHub account and go to https://github.com/new or click the **New** repository icon from your account homepage.


Why GitHub?
Team
Enterprise
Explore
Marketplace
Pricing
Search GitHub
Sign in
Sign up
Where the world builds software
Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.
Email address
Sign up for GitHub
65+ million Developers
3+ million Organizations
200+ million Repositories
72% Fortune 50



[Part 1 - 2] Name your repository username.github.io, replacing username with your GitHub username. Be sure it is public and go ahead and tell GitHub to create a README file upon generating the repo.


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * weblabtutorial / Repository name * weblabtutorial.github.io ✓

Great repository names are short and memorable. Need inspiration? How about redesigned-octo-happiness?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

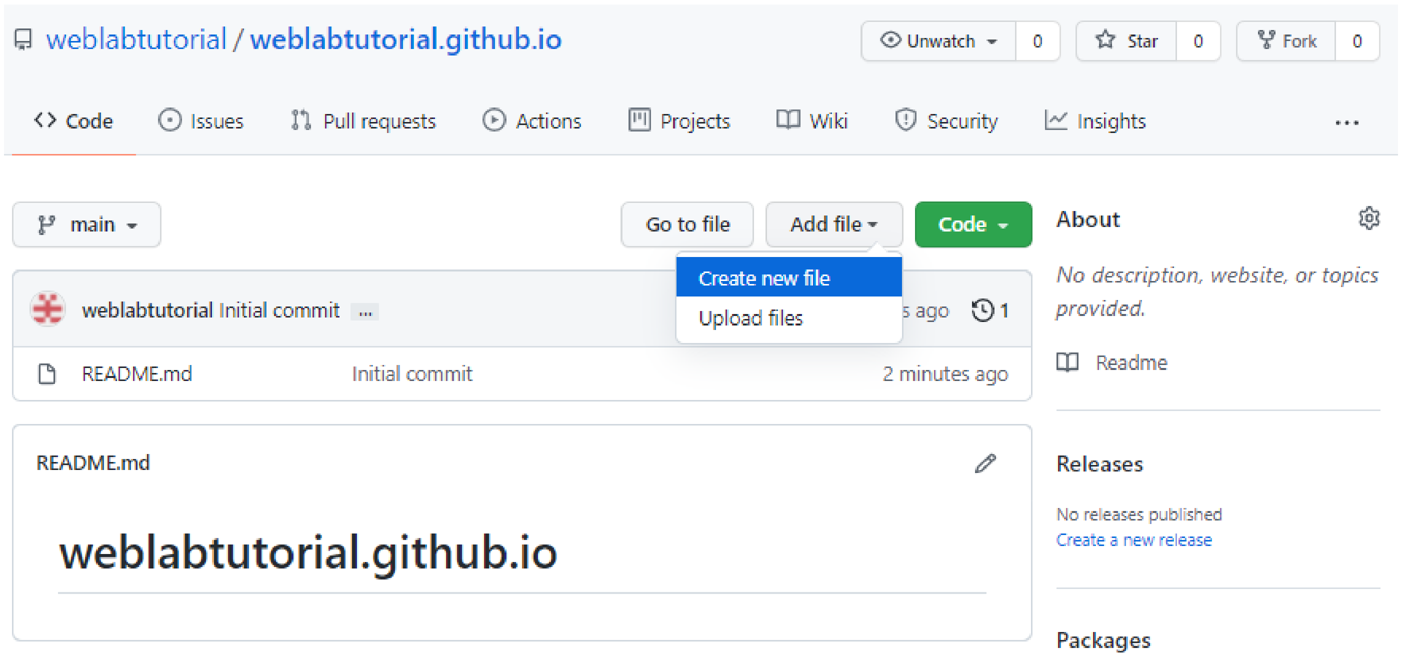
[✓ Add a README file] This is where you can write a long description for your project. Learn more.

Add .gitignore Choose which files not to track from a list of templates. Learn more.

Choose a license A license tells others what they can and can't do with your code. Learn more.

This will set main as the default branch. Change the default name in your settings.

[Part 1 - 3] Create an index.html page by clicking the plus icon next to your repository name and typing the file name directly in the input box that appears.



weblabtutorial / weblabtutorial.github.io

Unwatch 0 | Star 0 | Fork 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

main | Go to file | Add file | Code

Create new file
Upload files

weblabtutorial Initial commit ... ago 1

README.md Initial commit 2 minutes ago

README.md

weblabtutorial.github.io

About

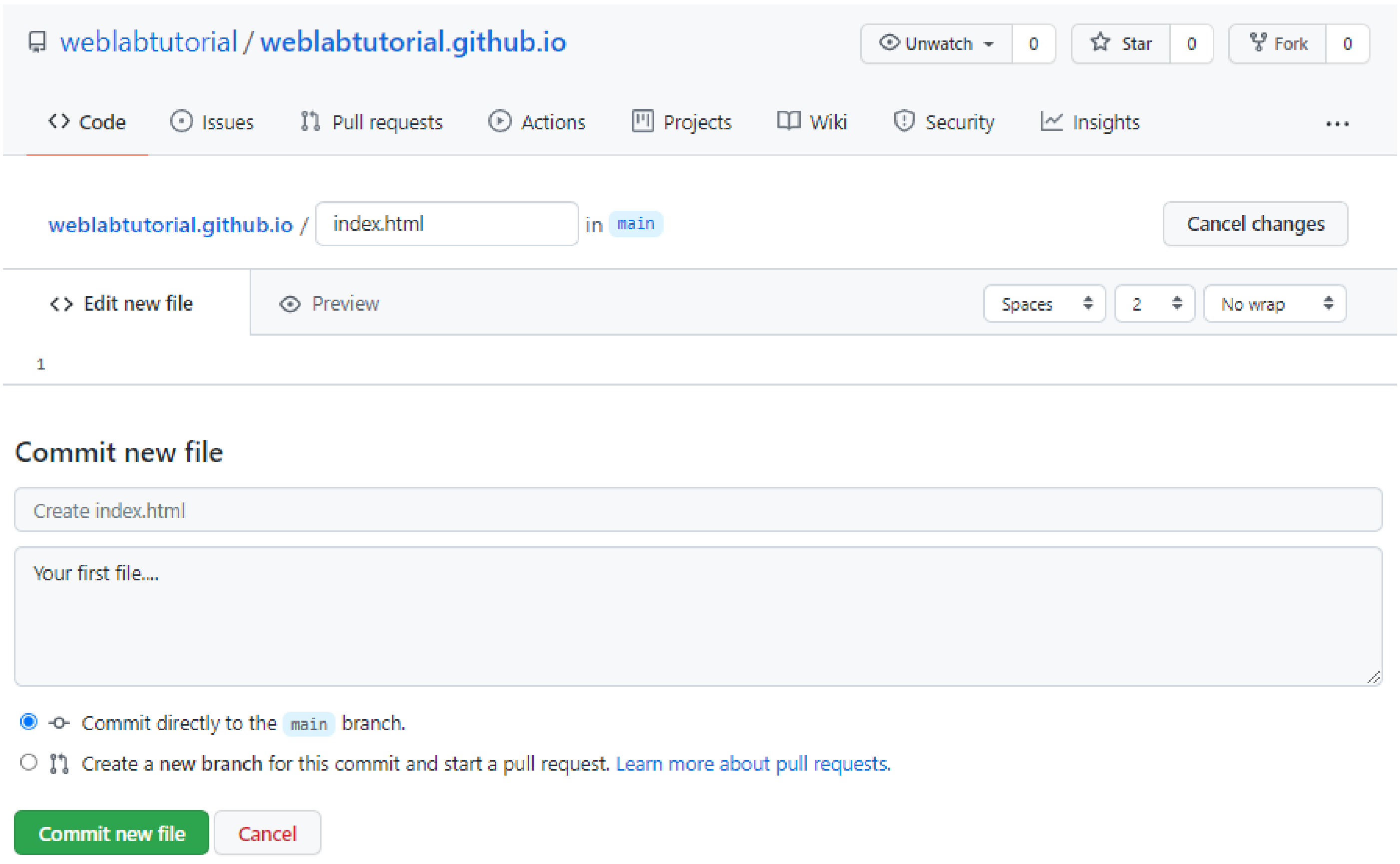
No description, website, or topics provided.

Readme

Releases

No releases published
Create a new release

Packages



weblabtutorial / weblabtutorial.github.io

Unwatch 0 | Star 0 | Fork 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

weblabtutorial.github.io / index.html in `main` | Cancel changes

Edit new file | Preview | Spaces | 2 | No wrap

Commit new file

Create index.html

Your first file....

Commit directly to the `main` branch.

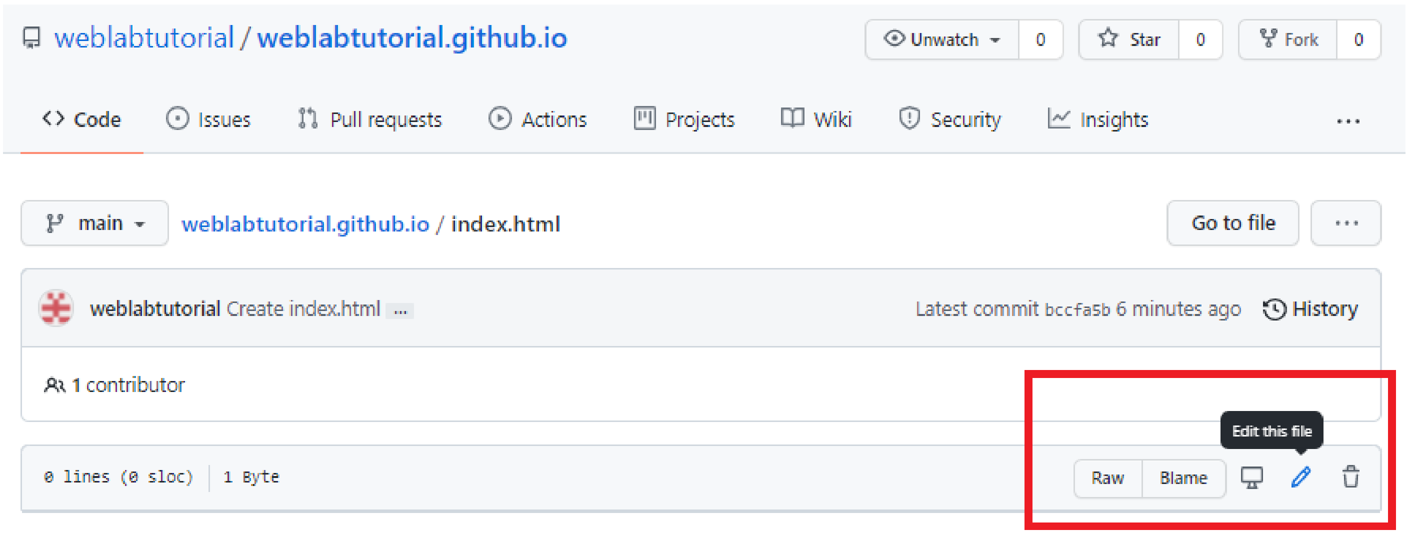
Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Commit new file | Cancel

On the resulting page, put this markup inside of the GitHub text editor:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Web Programming</title>
        </head>
        <body>
                <nav>
                <ul>
                <li><a href="/">Home</a></li>
                <li><a href="/blog">Blog</a></li>
                </ul>
                </nav>
                <div class="container">
                <div class="blurb">
                <h1>Hello World</h1>
                <p>Your page ….</p>
                <footer>
                  <ul>
                    <li><a href="https://github.com/weblabtutorial">github.com/weblabtutorial</a></li>
                    <li><a href="https://weblabtutorial.github.io"> weblabtutorial.github.io </a></li>
                </ul>
                </footer>
        </body>
</html>
```

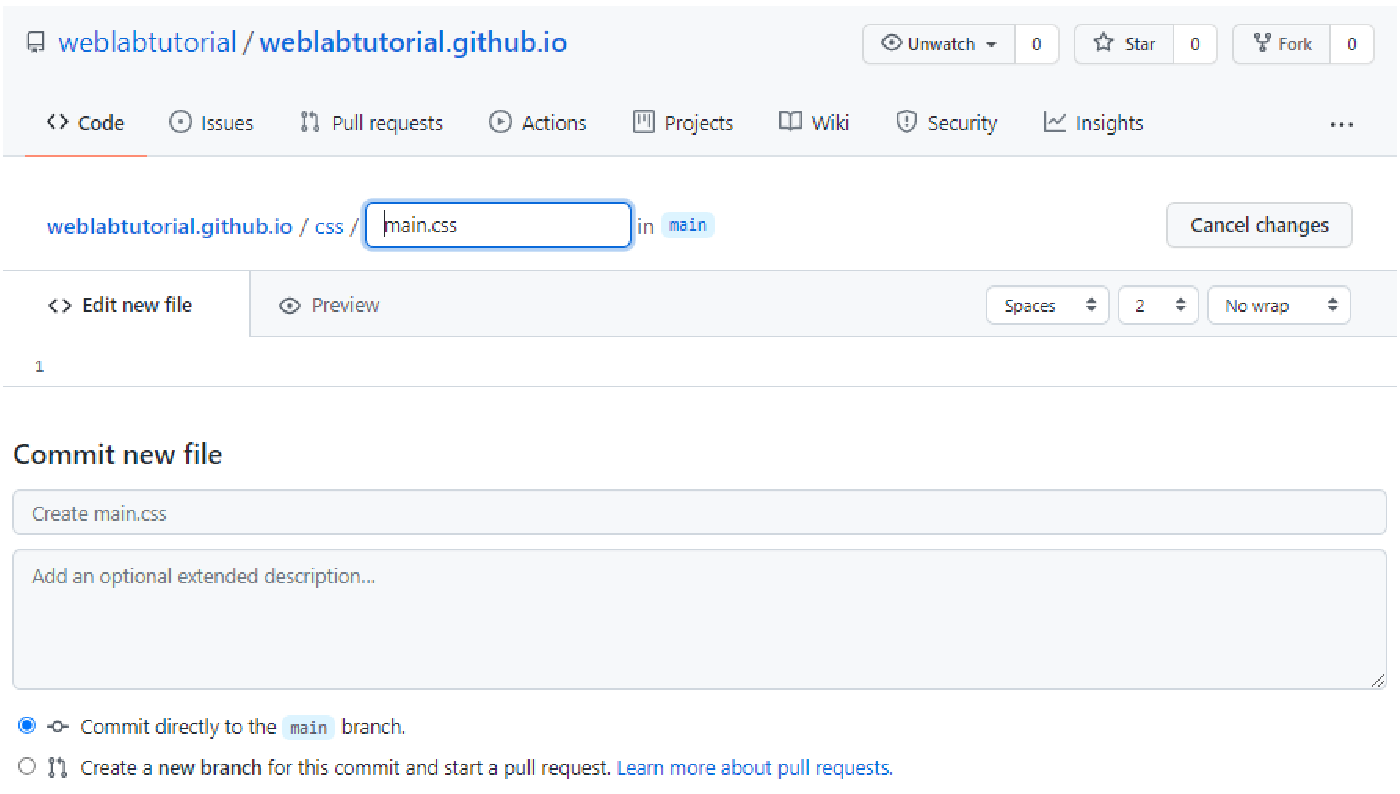
Note, if you need to edit a particular file, you can simply select it and then choose ‘edit this file’ (as shown below).


weblabtutorial / weblabtutorial.github.io
Unwatch 0
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
weblabtutorial.github.io / index.html
Go to file
weblabtutorial Create index.html
Latest commit bccfa5b 6 minutes ago
History
1 contributor
0 lines (0 sloc) | 1 Byte
Edit this file
Raw
Blame

[Part 1 - 4] Commit index.html. At the bottom of the page, there is a text input area to add a description of your changes and a button to commit the file.

Congrats! You just built your first GitHub Pages site. View it at http://username.github.io. Usually the first time your GitHub Pages site is created it takes 5-10 minutes to go live, so while we wait for that to happen, let's style your otherwise plain HTML site.

[Part 1 - 5] To style the content go back to your repository home and create a new file named css/main.css. The ‘css/’ before the filename will automatically create a subdirectory called css (saves you having to manually create folders).


weblabtutorial / weblabtutorial.github.io

weblabtutorial.github.io / css / main.css in main

Spaces 2 No wrap

Commit new file

Create main.css

Add an optional extended description...

Commit directly to the main branch. [✓]

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Place the following inside main.css:

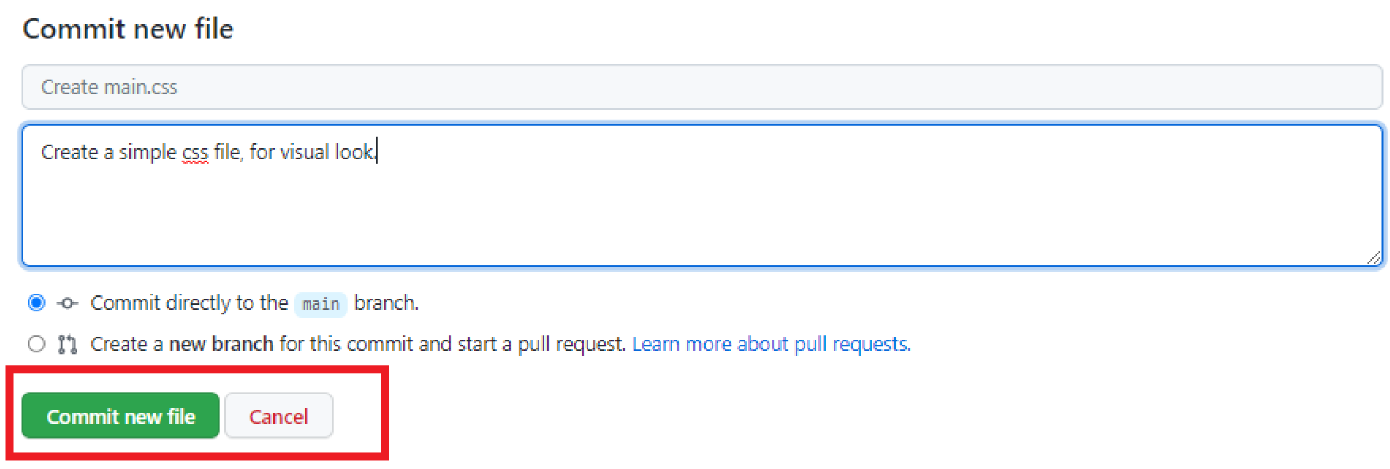
```
body {
   margin: 60px auto;
   width: 70%;
}
h1 {
   font-size: 3em;
   font-family:'Helvetica', 'Arial', 'Sans-Serif';
}
p {
   font-size: 1.5em;
   line-height: 1.4em;
   color: #333;
}
footer {
   border-top: 1px solid #d5d5d5;
   font-size: .8em;
}

nav ul, footer ul {
   font-family:'Helvetica', 'Arial', 'Sans-Serif';
   padding: 0px;
   list-style: none;
   font-weight: bold;
}
nav ul li, footer ul li {
   display: inline;
   margin-right: 20px;
}
a {
   text-decoration: none;
   color: #999;
}
a:hover {
   text-decoration: underline;
}

ul.posts {
   margin: 20px auto 40px;
   font-size: 1.5em;
}

ul.posts li {
   list-style: none;
}
```

Don't forget, when you add the new file, and its contents, that you need to ‘commit’ the file to your repository.


Commit new file

Create main.css

Create a simple css file, for visual look.

● Commit directly to the `main` branch.

○ Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

[Part 1 - 6] Link to your CSS file inside your HTML document's <head>. Go back to index.html and select the "Edit" button.

Add a link to main.css (new markup is in bold):

```
<!DOCTYPE html>
<html>
        <head>
                <title>Web Programming</title>
                <!-- link to main stylesheet -->
                <link rel="stylesheet" type="text/css" href="/css/main.css">
        </head>
        <body>
....
```

Test your website out by opening it in the browser – if you visit http://username.github.io, you should see your styled website (i.e., swap username for your repository username).

Part 2 Jekyll and GitHub Pages

Like GitHub Pages, Jekyll is self-aware, so if you add folders and files following specific naming conventions, when you commit to GitHub, Jekyll will magically build your website.

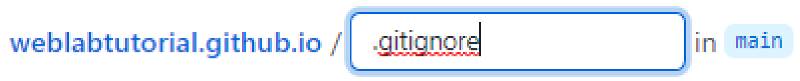
What is Jekyll?

Jekyll is a very powerful ***static site*** generator. In some senses, it is a throwback to the days of static HTML before databases were used to store website content. For simple sites without complex architectures, like a personal website, this is a huge plus. When used alongside GitHub, Jekyll will automatically re-generate all the HTML pages for your website each time you commit a file.

Setting Up Jekyll on github.com

In order for Jekyll to work with your site, you need to follow Jekyll's directory structure. To learn about this structure, we're going to build it right into our GitHub repo.

[Part 2 - 1] Create a '.gitignore' file. This file tells Git to ignore the '_site' directory that Jekyll automatically generates each time you commit. Because this directory and all the files inside are written each time you commit, you do not want this directory under version control.



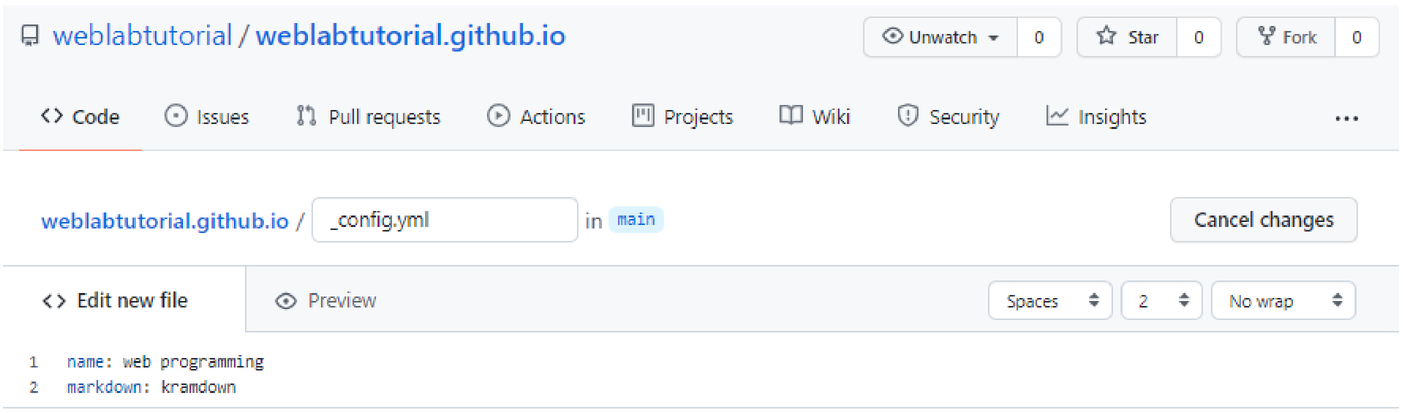
Add this simple line to the file:

```
_site/
```

[Part 2 - 2] Create a _config.yml file that tells Jekyll some basics about your project. In this example, we're telling Jekyll the name of our site and what version of Markdown we'd like to use:

```
name: web programming
markdown: kramdown
```

Look something like this



```
name: web programming
markdown: kramdown
```

At this point, hopefully you've got the hang of creating files and directories using the GitHub web interface, so we'll stop using screenshots to illustrate those actions.

[Part 2 - 3] Make a '**_layouts**' directory, and create file inside it called '**default.html**'. (Remember, you can make directories while making new files. See the main.css step if you forgot.)

This is your main layout that will contain repeated elements like our <head> and <footer>. Now you won't have to repeat that markup on every single page we create, making maintenance of your site much easier. So let's move those elements from index.html into default.html to get something that looks like this in the end:

```
<!DOCTYPE html>
<html>
	<head>
		<title>{{ page.title }}</title>
		<!-- link to main stylesheet -->
		<link rel="stylesheet" type="text/css" href="/css/main.css">
	</head>
	<body>
		<nav>
			<ul>
				<li><a href="/">Home</a></li>
				<li><a href="/blog">Blog</a></li>
			</ul>
		</nav>
		<div class="container">

		{{ content }}

		</div>
		<footer>
			<ul>
			 <li><a href="https://github.com/weblabtutorial">github.com/weblabtutorial</a></li>
			<li><a href="https://weblabtutorial.github.io"> weblabtutorial.github.io </a></li>
			</ul>
		</footer>
	</body>
</html>
```

Take note of the {{ page.title }} and {{ content }} tags in there. They're what Jekyll calls liquid tags, and these are used to inject content into the final web page. More on this in a bit.

[Part 2 - 4] Now update your index.html to use your default layout. Replace the entire contents of that file with this:

```
---
layout: default
title: Web Programming
---
<div class="blurb">
	<h1>Hello Sweet World</h1>
	<p>Once upon a time, ..</p>
</div>
```

Notice the plain text at the top of the file. Jekyll calls this the Front-matter. Any file on your site that contains this will be processed by Jekyll. Every time you commit a file that specifies layout: default at the top, Jekyll will magically generate the full HTML document by replacing {{ content }} in _layouts/default.html with the contents of the committed file.

Part 3 – Blog

A Jekyll-based blog uses the same conventions that you're familiar with from the previous steps, but takes things further by adding a few more details. Jekyll is very flexible allowing you to extend your site as you wish, but in this lab you're only going to cover the basics: creating a post, making a page to list our posts, creating a custom permalink for posts, and creating an RSS feed for the blog.

[Part 3 - 1] You'll want to create a new layout for your blog posts called **post.html** and a **folder** to store each individual post called **_posts/.**

Start by creating the layout. Create a file named **post.html** in your **_layouts** folder. Notice the post layout uses the default layout as it's base, and adds a couple new liquid tags to print the title of the post and date:

```
---
layout: default
---
<h1>{{ page.title }}</h1>
<p class="meta">{{ page.date | date_to_string }}</p>

<div class="post">
  {{ content }}
</div>
```

[Part 3 - 2] Make a **_posts/** directory where you'll store your blog posts. Inside that folder will be our first post. **<u>Jekyll is very strict with how these files are named</u>**, so pay attention. It must follow the convention YYYY-MM-DD-title-of-your-post.md. This file name gets translated into the permalink for the blog post. So in this example, you'll create a file named 2021-09-01-site-launched.md:

```
---
layout: post
title: "Site Launched "
date: 2021-09-01
---

Created a simple website using GitHub Pages with integrated Jekyll (note md files use Markdown for syntax
formatting).  **Bold** words.  As Markdown is actually is a lot easier to use for writing posts/notes than raw
HTML.
```

Note the file extension **.md** stands for **Markdown**, and the Markdown syntax used inside the file gets converted to HTML by Jekyll. Like Wikitext, Markdown is a markup language with a syntax closer to plain text. The idea of Markdown is to get out of the author's way so they can write their HTML content quickly, making Markdown very suitable as a blog authoring syntax. If you aren't already, you'll want to get familiar with Markdown syntax, and this printable cheatsheet (PDF) will be your best friend.

After committing the new post, navigate to http://username.github.io/YYYY/MM/DD/name-of-your-post to view it.

All this is great, but your readers won't always know the exact URLs of your posts. So next we need to create a page on our site that lists each post's title and hyperlink. You could create this list on your homepage or alternatively, create a blog subpage that collects all of your posts.

[Part 3 - 3] Create a blog directory and create a file named **index.html** inside it. To list each post, you'll use a foreach loop to create an unordered list of our blog posts:

```
---
layout: default
title: Web Programming Page
---
        <h1>{{ page.title }}</h1>
        <ul class="posts">

          {% for post in site.posts %}
            <li><span>{{ post.date | date_to_string }}</span> » <a href="{{ post.url }}" title="{{ post.title }}">{{
post.title }}</a></li>
          {% endfor %}
        </ul>
```

Now checkout http://username.github.io/blog/. You should see your first post listed and linked there. Good job!

[Part 3 - 4] Customizing Your Blog
You've only begun to scratch the surface with Jekyll's blog aware functionality. This lab is only going to cover a couple more steps you might want to take for your blog.

You may have noticed that the URL of your blog post does not include the blog directory in it. In Jekyll you can control the structure of our permalinks by editing the _config.yml file you created earlier. So let's change your permalink structure to include the blog directory.

Edit the _config.yml file. Add the following line at the end of the file:

```
permalink: /blog/:year/:month/:day/:title
```

Now your blog posts will live at http://username.github.io/blog/YYYY/MM/DD/name-of-your-post.

It is also very easy to setup an RSS feed for your blog. Every time you publish a new post, it will get added to this RSS file.

[Part 3 - 5] You're almost done! Don't forget to create and commit an about/index.html page. Since you should have got the hang of things now, and be able to add additional pages (i.e., about.html).

Extra Tutorials/GitHub Repositories to Explore

- Beautiful Jekyll
- Building Websites With Jekyll and GitHub
- Convert an HTML site to Jekyll

Part 4 Optional Extras (Not Assessed – 0 Marks)

Hopefully this lab has given you the confidence to do many other things with Git, GitHub, Jekyll, and your website or blog. You could go in many different directions at this point, you've probably already started thinking about possible future updates, but here are a few other things to try if you have time:

- Create _includes. They're a lot like _layouts, only smaller snippets of markup and can be injected into your _layouts and pages.
    - Try creating an _include file that inserts Google Analytics tracking code into your <head> so you can get stats on the visitors to your website (see example here)
    - Want commenting for your blog? Create a DISQUS _include and call it in your post.html layout.

- Don't want github.io in your URL? Set up a custom domain.
- Add blogging pagination
- Create a sitemap.xml file for better SEO. You can have one automatically generated by GitHub Pages.

- Become a coding pro and create a development branch of your site. Each time you want to fix a bug or add a new feature you can create a copy of your master branch, make your changes, and then merge that branch to the master branch. The idea is to keep the master branch clean.
- Need more inspiration? Check out how the sites from the developers of Jekyll are setup or browse this huge list of sites using Jekyll.

- Create a 'webhook' on the GitHub repository page, so each time a change is committed to your website repository, you're sent an email with the details (useful when teams are working on the same project)