GitHub Pages

A non-exhaustive introduction by

@KirbyPaint

Introduction to Programming

- In the beginning, the universe was created, blah blah
- Eventually, the abacus convinced humans to teach rocks to think, so it may finally retire
- This ultimately led to the mistake that is JavaScript and now we have websites and shit

Actual Introduction

- GitHub Pages is free site hosting (within limits)
- Comes with a domain (`username.github.io/repository`)
  - Username: your GitHub Username (`KirbyPaint`)
  - Repository: where the code lives
- GitHub Pages is one way to statically host a site
  - Static site: pre-loaded information (personal blog, wiki)
  - Dynamic site: user-submitted info (SocMed, YouTube)

What You Need

- GitHub account (free!)
- Text Editor
  - Anything that edits text *can* be a code editor... but should it…? no
  - Recommended: anything with syntax highlighting
    - VS Code, Notepad++, there are others probably
  - Not Recommended: **Microsoft Word** or any rich text editor, Vim*
    - *Vim is actually very good if you already know what you’re doing.
      If you get stuck in Vim, just
      - `esc esc :q!`
        see that makes total sense, right?????
    - A terminal text editor will be much faster than keyboard & mouse, if you take the time to learn it <3 we do not have that time today

First steps

- Make a new repository on GitHub
  - Needs a unique (to you) repo name, description of the project
  - MUST BE PUBLIC for a GitHub Pages site
  - Optional
    - `README.md` file, will only show on the repository home page, recommended for easier initial setup
    - A `.gitignore` can be added later
    - Licenses are more often used for code that will be re-used, not a static site

Using VS Code for Web

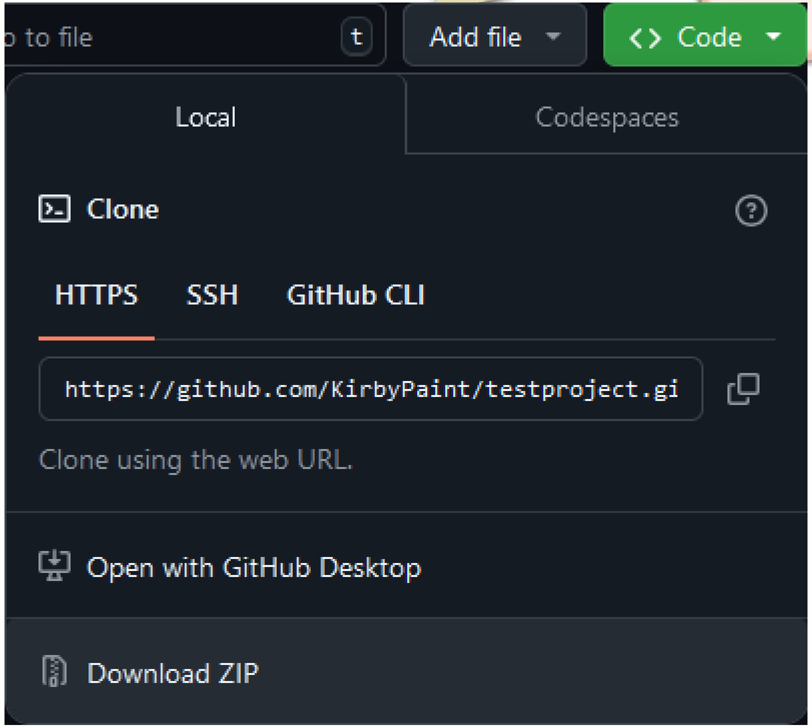
- Press “.” in browser when viewing your newly-created repo
- You can edit existing repository code in the browser
- I do not know how to make it save new files idk wasn’t working
- Useful when there is already code there

Cloning Your New Repo

- What are all those science words???
- “Cloning” a “repo” aka:
  - Cloning: making a copy that knows where the original is
  - Repo: short for “repository”

Cloning Repo pt. 2

- On repo home page look for big green button
- Click it and find “Download ZIP”
- Extract that
- Go into it all the way until you see your README



```
https://github.com/KirbyPaint/testproject.gi
```

Clone using the web URL.

Baby's First HTML Document

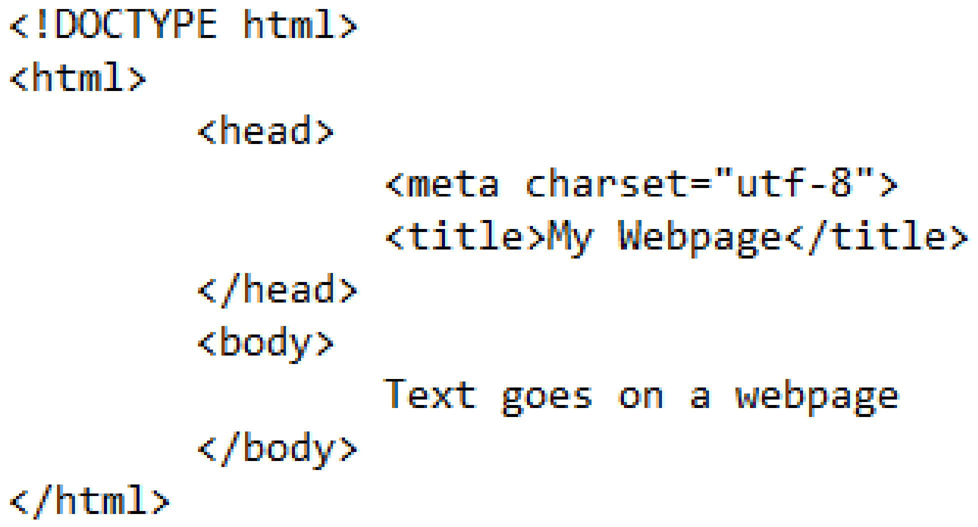
- Create a new file named `index.html` somehow:
  - Windows
    - GUI: open notepad, save file name as "index.html" AND save type as "All Files"
    - Command Prompt: pray that it's opened to `C:\Users\your_user>`
      - `cd Downloads` and `Enter` and then `cd repo_name` and `Enter` and then:
      - `type nul > index.html`
      - `Enter`

Baby's First HTML Document

- Create a new file named `index.html` somehow:
  - Non-Windows
    - Mac GUI: open TextEdit, save file as index, file format .html
    - Mac Terminal: ctrl+click the repo folder, Services > "New Terminal at Folder"
      - `touch index.html` and then hit `Enter`
    - Linux: wyd you probably already know all this, nerd, go install Arch
      - `cd /path/to/file && touch index.html`

Coding!

- Barebones HTML needs the following tags:
- <!DOCTYPE html>
- <html>
- <head>
- <meta charset=”utf-8”>
- <title>
- <body>



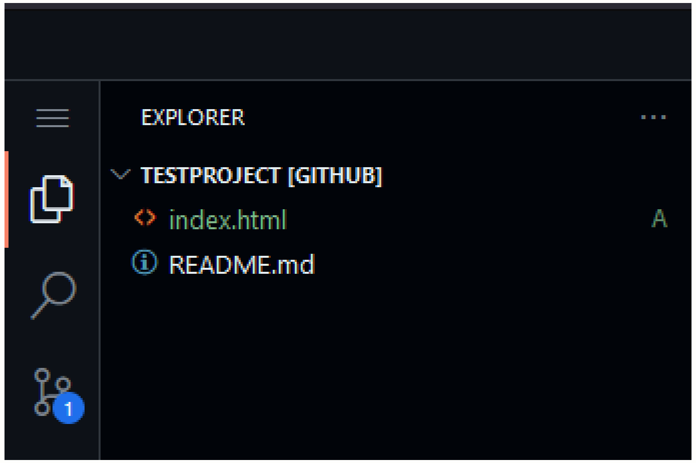
```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>My Webpage</title>
        </head>
        <body>
                Text goes on a webpage
        </body>
</html>
```

Committing Your Code

- To `commit` code is essentially to create a save state of code
- There’s a lot more to it than that but learning `git` is like ten more powerpoints ngl
- We will not be installing the `git cli` here, we will be using the web editor instead
- Since Git is meant to be interfaced via command line, this presentation will lack a lot of the power of the `cli`
- `cli` means command line interface, you say it c-l-i not “clih”

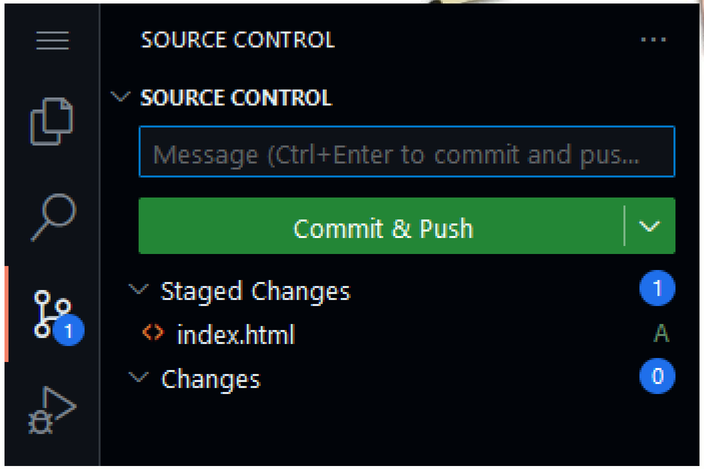
Committing Your Code

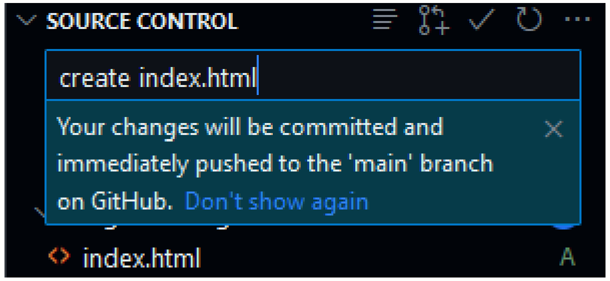
- Go back to the web editor for VS Code with your project
- Drag and drop index.html into your project root (top level)
- See the circle lines with a 1? That means you have Uncommitted Changes



Committing Your Code

- Click the + next to your file(s) you want to commit
- Anything under Staged Changes will be in one commit
- Enter a short message describing the commit
- Commits should be present-tense “add file” vs “added file”
- That’s not required by law or anything, just convention





Your changes will be committed and immediately pushed to the 'main' branch on GitHub. Don't show again

Sidebar: Branches

- Git is version control, which means that multiple versions of code (branches) can exist.
- By convention, the `master` branch has been renamed to `main`, to distance from master/slave terminology
- Also it is a 33% reduction in total typed characters, saving precious keystrokes :D
- Why this is important:

Sidebar: Branches

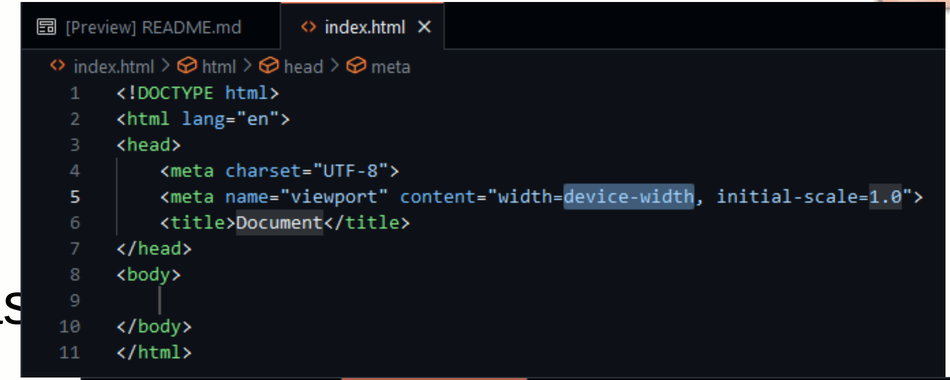
- GitHub Pages works off of the `gh-pages` branch
- Anything named `gh-pages` with a root index.html file is known by GitHub to deploy in a particular way
- Do not worry about changing branches right now, that happens soon

Committing Your Code

- Once your changes have been staged and your commit message has been written, click “Commit & Push”
- “Pushing” the code commit to the main branch is like saying “This is the version of my code I want others to see”
- In a production environment, if you are pushing to main, you better be the senior developer or you work at a terrible place
- Commits are good ways to test code functionality – if something goes wrong, revert the commit (out of scope for this lesson)

Using the Editor

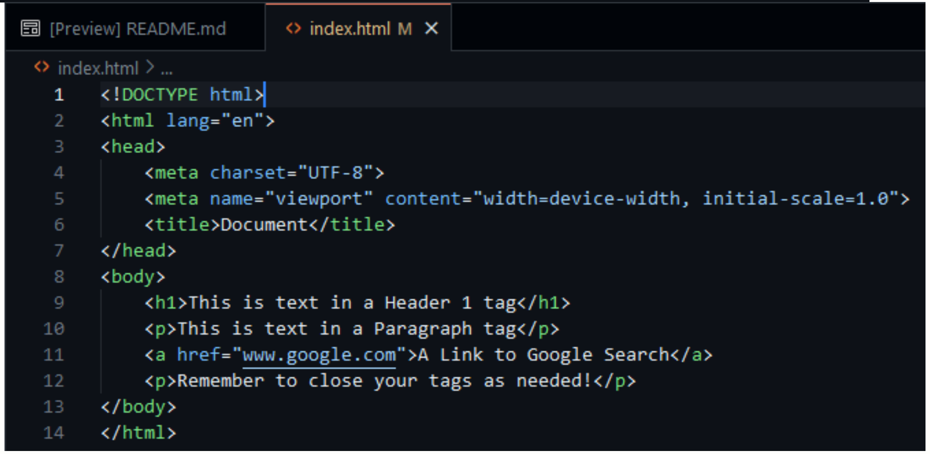
- VS Code has nice shortcuts
- On index.html, type "!" then press Tab
- Watch as it autofills all the barebones html with some extras
- Highlighted areas can be typed and tabbed through
- Make a few changes, maybe throw some text in the body, then stage the file, commit changes, and push to main



[Preview] README.md | index.html ×

index.html > html > head > meta

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    
</body>
</html>
```



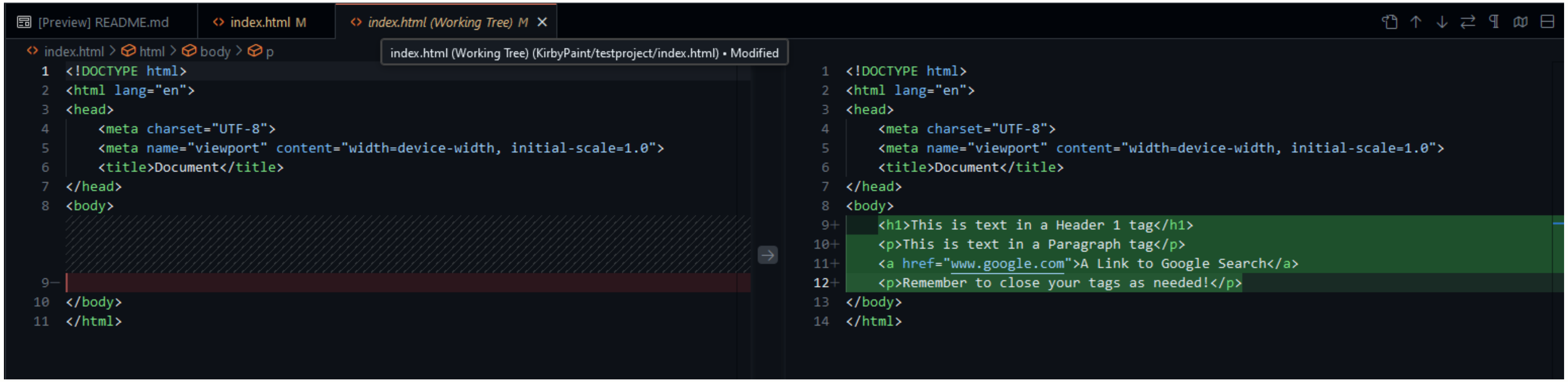
[Preview] README.md | index.html M ×

index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>This is text in a Header 1 tag</h1>
    <p>This is text in a Paragraph tag</p>
    <a href="www.google.com">A Link to Google Search</a>
    <p>Remember to close your tags as needed!</p>
</body>
</html>
```

The Diff

- The diff page will visually show changes you've made, handy when trying to come up with a commit message if you forgot what you did already (this is why small, frequent commits rule)


[Preview] README.md
index.html M
index.html (Working Tree) M
index.html > html > body > p
index.html (Working Tree) (KirbyPaint/testproject/index.html) • Modified

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>This is text in a Header 1 tag</h1>
    <p>This is text in a Paragraph tag</p>
    <a href="www.google.com">A Link to Google Search</a>
    <p>Remember to close your tags as needed!</p>
</body>
</html>
```

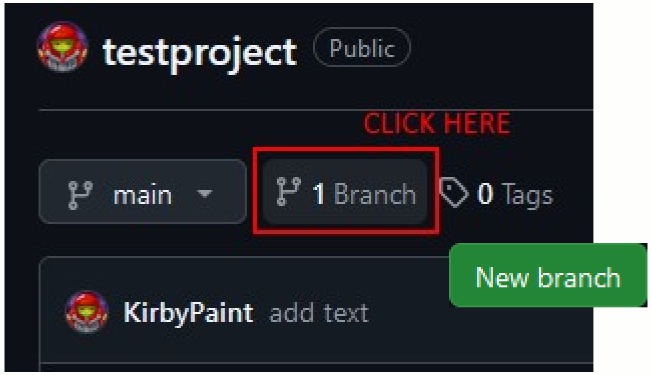
Viewing Your Page

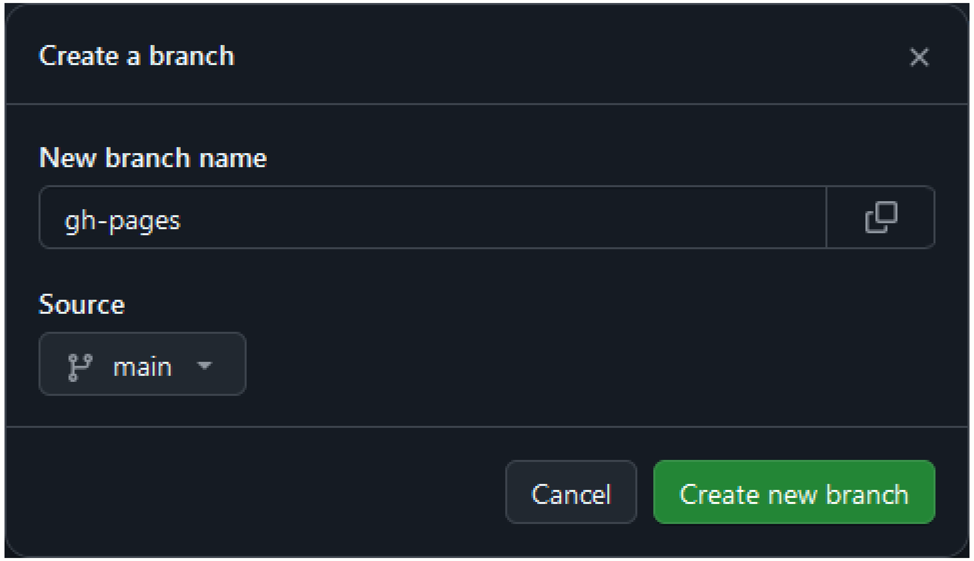
- VS Code when fully installed has ways to live preview your pages as you develop your site
- If you edit outside of the web, you can just open your local index.html in your browser and refresh after you’ve saved your page
- Out of scope for this lesson, but immensely powerful in a real coding environment

Merge Changes

- Merging the branch is like putting two different code pages together. `git merge` will compare differences between branches and allow the developer to choose which code stays and goes
- Generally, `merging` is not desirable, as it can `squash` commits, and `rebasing` is often preferred
- This does not matter for this presentation, as it is too complex to explain the differences succinctly and in a way that matters
- For now, we are treating `gh-pages` as a copy of `main`

Merge Changes




Create a branch

New branch name

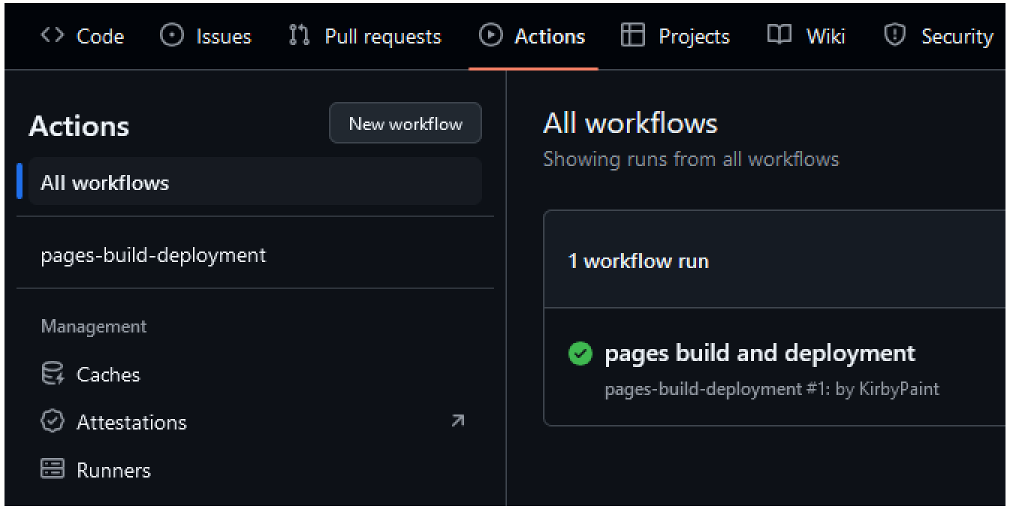
gh-pages

Source

main

GitHub Actions

- GitHub has automated `CI/CD` aka it will handle the site deployment
- Go to the Actions tab on the repo to see the workflow
- If everything is correct, the workflow will run and deploy your site



All workflows

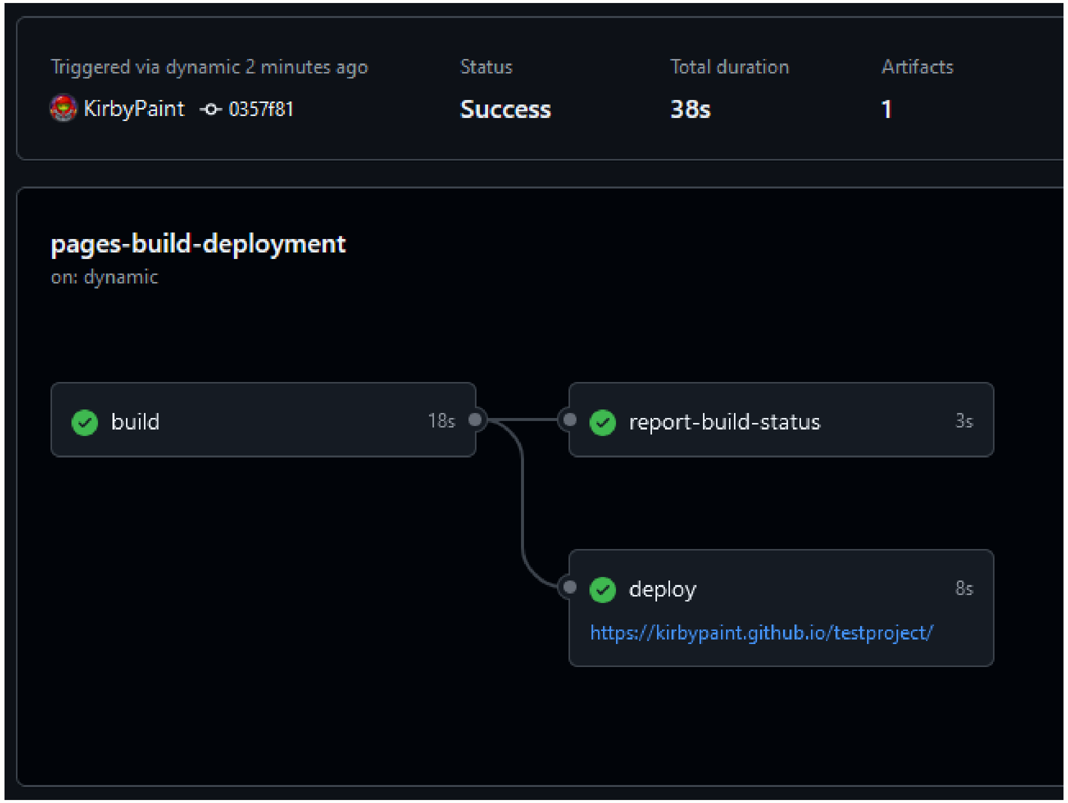
Showing runs from all workflows

1 workflow run

pages build and deployment

pages-build-deployment #1: by KirbyPaint

GitHub Actions



Triggered via dynamic 2 minutes ago
KirbyPaint 0357f81

Status
Success

Total duration
38s

Artifacts
1

pages-build-deployment
on: dynamic

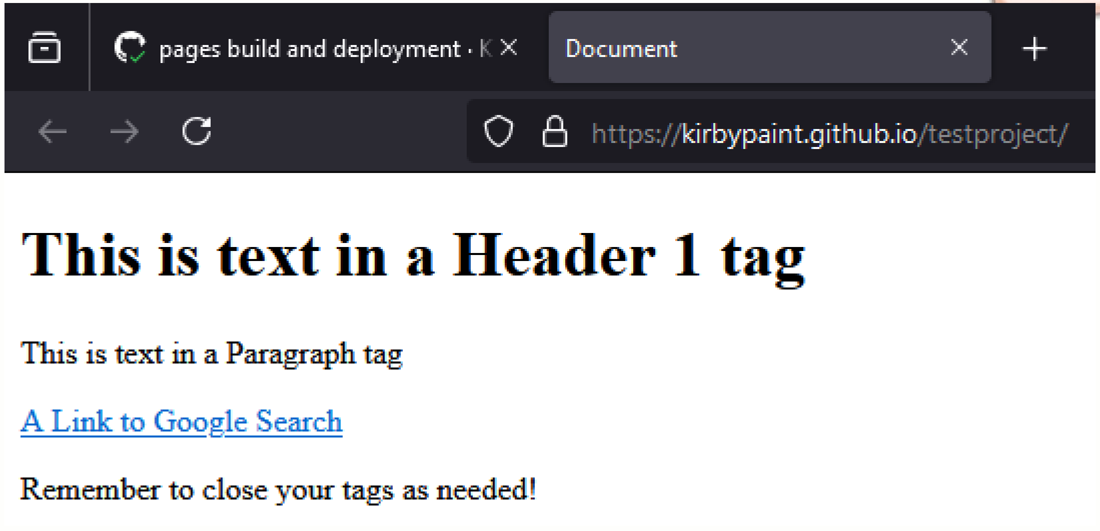
build 18s

report-build-status 3s

deploy 8s
https://kirbypaint.github.io/testproject/

View Your Muhfukin Website

- The Action should contain the deployment URL
- If not, the page will be hosted at
  - https://github_username.github.io/repository/


pages build and deployment · K
Document
https://kirbypaint.github.io/testproject/

This is text in a Header 1 tag

This is text in a Paragraph tag

A Link to Google Search

Remember to close your tags as needed!

THANK YOU FOR YOUR ATTENTION
ANY QUESTIONS?