Google Summer of Code 2023

Greener coding: Wagtail's climate impact

Name and Contact Information

Name: Paarth Agarwal

Github username: PaarthAgarwal

Email: paarthagarwal551@gmail.com

Alternative Mail: paarth.agarwal.mec20@itbhu.ac.in

University Name: Indian Institute of Technology (BHU) Varanasi

Current year: 3rd year

Twitter handle: Paarth Agarwal (@AgarwalPaarth) / Twitter

LinkedIn: Paarth Agarwal | LinkedIn

Country: India

Time Zone: IST (UTC+05:30)

Link to proposal doc:

https://docs.google.com/document/d/15xnHV2Dvuq_D4FY9u3RC8zcm244vyH0noS9QZsuQ9kA/edit?usp=sharing

Table of Contents

Overview

Abstract

Climate change is threatening the planet because human activity emits too much greenhouse gas. When I use a digital device, I consume electricity. Producing this electricity emits CO2. Digital activity currently accounts for 4% of carbon emissions worldwide, **growing by 9% annually**. That's *fast*: according to **the rule of 70**, it will double in the next 7.7 years. At 8%, digital will then emit as much as cars.

Companies are more and more data-driven. The Wagtail CMS is a popular open-source content management system many websites use worldwide. This project aims to address sustainability issues related to the Wagtail CMS by implementing energy-efficient features and reducing its overall carbon footprint.

"If you can't measure it, you can't improve it."

Before moving ahead, we must find a reliable way to measure Wagtail's CO2 emission and impact. Many tools are available for it, and based on the discussion, the GreenFrame CLI and Green Metrics Tool appear to be the best at this time. We have to reduce the energy consumption of the CMS while maintaining its functionality and user experience. By making the Wagtail CMS more sustainable, this project aims to contribute to reducing the environmental impact of web applications.

Project Goals and Benefits

1. Developing a suite of performance tests to measure the energy efficiency of Wagtail.
2. Produce a report on the climate impact of Wagtail sites at this point in time.
3. Implementing more sustainable web development practices in Wagtail and improving energy efficiency.
4. Encouraging the adoption of sustainable web development practices by raising awareness among Wagtail developers.

By the end of the project, I want to make Wagtail much more sustainable and greener, setting an example for other platforms.

How are other CMSs doing it?

I researched Joomla, Plone, Drupal and WordPress. Of all four of them, Drupal has done considerable work in this direction and has a sustainability channel on Slack. In their work with the Climate Group, they claimed,

"*Overall we cut down page weights by 50% which, in turn, halved the average page load time, increasing the website's overall SEO and security rankings.*" - ***A sustainable web build for The Climate Group | Drupal.org***

They have been implementing **image-optimizing techniques**, **serving images in the next-gen format** and **speeding up tests**. **Here** is a list of tasks they plan to accomplish to reach their sustainability goals.

I haven't found many publicly available reports for other CMS, but I found that WordPress is being supported by the Green Web Foundation through Green Web Fellowship. One of the **Green Web Fellows**, **Nahuai Badiola**, a freelance WordPress developer and web sustainability advocate, is part of the team behind the new

WordPress sustainability initiative. According to him, his project is centred on bringing **more awareness of digital sustainability** and climate justice to the WordPress community and its users. The project's result could be a **workshop on WP Events**.

Approach

Implementation and Rationale

The foremost priority of the project is to set up the codebase well enough to be tested with ease by the GreenFrame CLI and Green Metrics tool and customise it according to our use case to measure the energy impact of different parts of the CMS. Doing so will guide us and show how successful we remain in reducing emissions. The results will help inform future changes to the CMS.

Once we achieve that, we can do weekly or bi-weekly runs to check our impact on a regular basis.
Now that we can measure the impact, we move towards reducing it. A few sustainability-related changes that I plan to bring to Wagtail:

1. **Serving responsive images**

   As mobile devices have improved, we've all become used to seeing crystal-clear images on our retina screens. But as developers, we can serve different images to all screen sizes and devices. A full-screen hero image on a large monitor does not need to be served up at the same size on a mobile device. It's straightforward to generate images in different sizes and allow the browser to select the optimal image to display.

   a. Resolution switching: Different sizes:
      Depending on the device, we want to display identical image content, larger or smaller. The standard <img>

element traditionally only lets us point the browser to a single source file:

```
<img src="elva-fairy-800w.jpg" alt="Elva dressed as a fairy" />
```

We can, however, use two attributes — **srcset** and **sizes** — to provide several additional source images along with hints to help the browser pick the right one.

```
<img
  srcset="elva-fairy-480w.jpg 480w, elva-fairy-800w.jpg 800w"
  sizes="(max-width: 600px) 480px,
         800px"
  src="elva-fairy-800w.jpg"
  alt="Elva dressed as a fairy" />
```

b. Resolution switching: Same size, different resolutions
If we're supporting multiple display resolutions, but everyone sees our image at the same real-world size on the screen, we can allow the browser to choose an appropriate resolution image by using srcset with x-descriptors and without sizes — a somewhat easier syntax.

```
<img
  srcset="elva-fairy-320w.jpg, elva-fairy-480w.jpg 1.5x, elva-fairy-640w.jpg 2x"
  src="elva-fairy-640w.jpg"
  alt="Elva dressed as a fairy" />
```

There is an RFC for the same - **rfcs/071-responsive-images-support.md at main · wagtail/rfcs (github.com)**, and I'd like to implement it.

2. Serving images in next-gen formats

Image compression technology has come on in leaps and bounds over the past few years, and for browsers that can

support them, next-gen formats such as WebP are much more efficient than JPEGs and PNGs. In order to make it work on all Wagtail-supported browsers, we need to have a fallback strategy.
One solution is to use the **pillow-avif-plugin**, which adds support for AVIF files and can serve as a fallback for browsers that do not support WebP. By generating images in the appropriate format for each browser and making them available on the generated markup, we can significantly **reduce data transferred with each page request**, leading to a more sustainable web.
Another solution can be to create a template tag that outputs both a WebP and a fallback version of an image.
The **formats** that Google Lighthouse indicates are JPEG 2000, JPEG XR, and WebP, which still have limited browser support.
JPEG XR: **IE and Edge compatible**
JPEG 2000: **Safari compatible**
Webp: **Chrome compatible** (and Edge also, but not IE).
We need to also generate images in that format from the image style and then make them available on the generated markup.
This gives an immediate **win in terms of data transferred with each page request** and, as a result, reduces carbon emissions.

We can measure the impact of these changes by using tools like **Google PageSpeed Insights** or **Lighthouse** to compare the **file size** and **loading times of images** on your site before and after implementing responsive images and next-gen formats.

3. Template caching for sites

Template caching is a technique used to improve website performance by reducing the time it takes to render web pages. When a page is requested, the server runs the necessary code and renders the HTML page, which is then sent to the user's browser. This process can be slow, especially for complex pages requiring many database queries or server-side processing.

The server can save a copy of the rendered HTML page in memory or on disk by caching templates. The next time the same page is requested, the server can serve the cached copy instead of re-rendering the entire page. This can significantly **improve page load times and reduce server load**, resulting in a more efficient and faster website.

In the context of Wagtail, template caching can be implemented using various tools, such as Django's built-in caching framework or third-party packages like Django Compressor. By enabling template caching, Wagtail sites can benefit from **faster page load times and improved user experience** while also reducing the carbon **emissions associated with server-side processing**.
We can use Django's built-in template caching mechanisms to implement template caching in Wagtail. We should update the default project template to have caching enabled and reference the same in docs. Here are the general steps:

- **Enable Django's cache backend:** We'll need to choose and enable a cache backend in your Django settings. [Django supports a variety of cache backends](), including

in-memory caching, file-based caching, and database caching.

- **Set up template caching:** Using Django's cache template tag, we can set up template caching in our Wagtail templates. This tag will allow us to cache the output of a template for a specified amount of time. Sample implementation -

```
{% load cache %}
{% cache 300 my_cache_key %}
    <p>This HTML will be cached for 5 minutes.</p>
{% endcache %}
```

  In this example, my_cache_key is a unique identifier for the cached content, and 300 is the number of seconds that the content will be cached.
- **Configure cache settings:** In the Django settings file, we can configure the cache settings for a Wagtail site using the CACHE_MIDDLEWARE_SECONDS setting. This setting determines the default length of time that responses will be cached. One can also set per-view cache settings using **Django's cache middleware**.

The most straightforward way to measure the impact of template caching is by comparing the **page load time** before and after implementing caching. Additionally, we can also measure the impact of template caching on server load and bandwidth usage. By reducing the **number of requests** to the server, we can decrease the energy required to serve the website, thus reducing its carbon footprint.

To monitor server load and bandwidth usage, we can use server monitoring tools such as **New Relic** or **Datadog.** These tools can give you **real-time insights** into your **server's**

performance and help you identify any issues that may be affecting your website's speed and efficiency.

4. YouTube lite embeds

For a single YouTube video embed, the browser must download ~900 kB of data to render the YouTube video player alone. And these files are downloaded even before the visitor has clicked the play button. The embedded YouTube video increases the byte size of web pages, and the browser has to **make multiple HTTP requests** to render the video player. This increases the overall **loading time** of the page, thus affecting the **page speed** and the **core vitals score** of the website.

YouTube lite embeds for Wagtail is a feature that provides a lightweight and more efficient way to embed YouTube videos in Wagtail pages. With this feature, only the video thumbnail and a play button are displayed on the page, and the player is only loaded when the user clicks the play button.

The YouTube lite embeds for Wagtail can be implemented by creating a custom Wagtail embed handler that replaces the default '**embed**' tag with a custom HTML template that includes the thumbnail and play button. The handler can then use JavaScript to load the YouTube player only when the play button is clicked.
Sample code to create a new template adapter in a

templatetags module, such as youtube.py:

```python
from django import template
from django.template.base import FilterExpression
from django.utils.safestring import mark_safe
from wagtail.core.models import Page

register = template.Library()

@register.filter
def youtube_embed(value, size='480x270'):
    try:
        video_id = value.split('/')[-1]
    except Exception:
        return value
    return mark_safe(f'<div class="video-wrapper"><iframe width="{size.split("x")[0]}" h
```

In the template:

```
{% load youtube %}

{{ page.youtube_video_url|youtube_embed:"640x360" }}
```

One way to implement it can be using [paulirish/lite-youtube-embed: A faster youtube embed. (github.com)](). According to them

"*Provide videos with a supercharged focus on visual performance. This custom element renders just like the real thing but approximately 224× faster.*"

Implementing YouTube lite embeds for Wagtail can contribute to sustainability by resulting in **faster page load times**, **reduced data transfer**, **a better user experience** and reduced emissions.

One way to measure the impact is by using a tool like **Google PageSpeed Insights** or **GTmetrix**. These tools can show you the current performance metrics of your website, including page load time, data usage, and various other optimization opportunities.

You can use **browser developer tools**, such as the **Network tab in Google Chrome or Firefox**, to monitor the amount of data downloaded when a page with YouTube video loads. You can compare the data usage of standard YouTube embeds versus YouTube Lite embeds to see the impact of the change.

5. Carbon Aware CMS

The concept of a carbon-aware website is one where the content/user experience of a site changes depending on how the grid intensity of the electricity grid. Taking inspiration from Fershad Irani's **blog post**, it'll be awesome if we can bring this to Wagtail sites.
When the grid intensity is equal to or greater than the standard we decide, the following modifications can be made:

- Image quality can be reduced.
    - Visitors can click to download better-quality versions of any image if they need to.
- Remove AVIF images since decoding them can be more CPU intensive.
- Non-critical JavaScript can be removed from the site.

Fershad checks the location of visitors and uses it to fetch gird intensity data from **CO2 signal API**. However, CO2 signal API has a limit of 30 requests per hour which might be okay for a personal website but not suitable for public websites. If we go with this approach, there is a **research element**, but it'll be a great solution in the long run.

6. Speeding up tests

In agreement with LB and Chris's comments, our tests run 100s of times a day on GitHub servers. Increasing the performance there could make a huge difference.
One way would be to try running the tests with the --parallel flag. Running tests in parallel can significantly reduce the total time it takes to run all the tests in a project **by distributing the test suite across multiple processes or threads**.
Eco CI is a tool to give an estimate of the energy used for continuous integration runs:
https://www.green-coding.berlin/projects/eco-ci/
This is the same group that worked on the green code metrics tools. By integrating their **custom Github Actions** and **Github Apps** into our testing workflow, we can get an automated estimation of the energy cost of the workflow run.

7. Wagtail-aware Django debug toolbar

The **Django Debug Toolbar** is a popular third-party package for debugging Django applications during development. It is a **performance optimization tool** for Django web applications. It provides developers with insights into the performance of their applications and can help **identify performance bottlenecks** that can be optimized. Optimizing the performance of a web application can reduce the number of server resources required to serve the same number of requests, which can translate to energy savings and reduced carbon emissions. It also provides a set of panels displaying various debug information, such as request/response data, SQL queries, and cache usage.

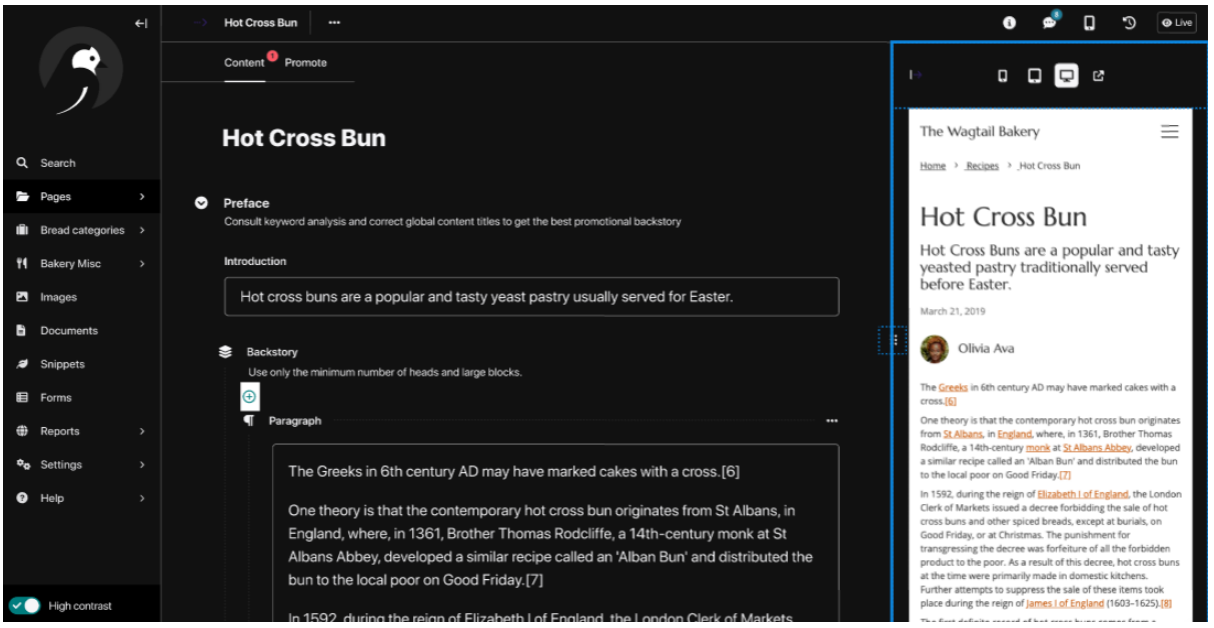
To make the Django Debug Toolbar Wagtail-aware, we can install a wagtail-debug-panel package, which adds a new panel to the toolbar showing Wagtail-specific debug information. The panel can display the following information:

- The current page is being viewed in the Wagtail admin.
- The page tree, with indicators for unpublished and locked pages.
- The current language being viewed (if using Wagtail's i18n support)

8. Improve accessibility features

Ensure that Wagtail's accessibility features are up-to-date and comprehensive to ensure that people with disabilities can use the platform effectively. This will reduce their need to rely on additional software or hardware to access the site, reducing their energy consumption. One of them can be dark mode.
To measure the impact of this change, we can use tools like **Google's Accessibility Scanner** or **Lighthouse** to check for accessibility issues on our site before and after implementing accessibility checks.

Dark mode: This can reduce energy consumption for users who prefer dark mode on their devices, as darker colours require less energy to display on most devices. Dark default is already being introduced, but Dark high-contrast variants can be introduced as the UI team plans. We can also add dark mode to Bakerydemo and Wagtail.org as part of this project Dark High Contrast:



(This image is just Proof of Concept. Actual designs may differ).

Some ways to quantify the improvement that dark mode brings:

- Power consumption: We can use power profiling tools like the **Firefox Profiler** to measure the power consumption of a web page in both light and dark modes and compare the results.
- Page load time: In some cases, dark mode can result in faster **page load times**, especially for pages with lots of white space.
- User engagement: Some users may prefer dark mode because it is **easier on the eyes** or provides a **better user experience**. Can be measured by tracking metrics like time on site, bounce rate, and conversion rate for users who use dark mode compared to those who don't.
- Tools: Tools such as the **JouleMeter** or **GreenFrame** can be used to compare a **website's power consumption** in light mode versus dark mode. We can also use performance testing tools such as **Google Lighthouse** or **WebPageTest** to compare **the loading time** and **network usage** of a website in both modes.

9. Optimising styles and scripts

Over time, it's often the case that sites accumulate a bit of tech debt, which can sometimes take the form of supporting CSS and JS code that is no longer used on the site. Plenty of tools can automatically detect stale files and code; keeping these clean and removing anything unnecessary will reduce the amount of data we are transferring with each request.
We can also shift our sites to use **system fonts** to eliminate the need for downloading additional font files. Wagtail CMS already uses system fonts, but we can do the same with bakerydemo and wagtail.org.

10. Spread awareness

We should also encourage Wagtail developers to use renewable energy sources when developing and testing the platform and provide them with some guidelines to make their site more efficient. For this, we can add a dedicated section in the documentation and start a sustainability channel on Slack. Perhaps a talk on sustainable development in the next Wagtail Space!

Project Timeline

Community Bonding and Onboarding(May 4, 2023 - May 28, 2023)	
`May 4, 2023 - May 28, 2023	• Discuss my project with my mentors and the community to see if any more improvements can

	be made to the plan of action. • Define specific sustainability goals and metrics to measure success. • Liaise with Wagtail's Performance and accessibility teams. • I will start a blog documenting my journey with Wagtail. • Prioritise the approaches proposed.
Project Period(May 29, 2023 - August 29, 2023)	
`May 29, 2023 - Jun 11, 2023	• Set up codebase to test with GreenFrame CLI and Green Metrics Tool. • Evaluate the current state of Wagtail and identify areas for improvement.
Jun 12, 2023 - Jun 25, 2023	• Add support for serving images in next-gen formats. • Test the results.
Jun 26, 2023 - Jul 9, 2023	• Add support for serving responsive images.

	• Continue making improvements and retesting until sustainability goals are met.
Jul 10, 2023 - Jul 23, 2023	• Enable Template caching for sites. • Implement a Wagtail-aware Django debug toolbar.
Jul 24, 2023 - Aug 6, 2023	• Add a plugin for YouTube lite embeds. • Improve accessibility features.
Aug 7, 2023 - Aug 20, 2023	• Make CMS carbon aware • Speed up tests. • Discuss stretch goals.
Aug 21, 2023 - Aug 28, 2023	• Complete stretch goals. • Finish up the documentation works. • Complete any remaining work and prepare my final report.

Extended Goals

1. Blog my entire journey
2. Deliver a talk at Wagtail space
3. At the moment, I'm the 21st highest contributor in terms of commits. My goal is to get in the top 10.
4. Keep contributing and fixing bugs.

About me

I am **Paarth Agarwal**, an undergraduate student at the Indian Institute of Technology (BHU) Varanasi, India, pursuing a Bachelor of Technology in my third year. I started programming in Python and C++ and learned essential web technologies in high school. Since my first year of college, I have been very enthusiastic about using Web-based technologies, including React and Django frameworks, for two years. I have experience in building REST APIs, JAMstack / full-stack web applications, and test-driven development. I am a **Full-Stack developer** with proficiency in **HTML, CSS, Javascript, Reactjs, Nodejs, Python, Django, Wagtail, Typescript, Docker and Github**. I love exploring new tech and have worked with various technologies and languages. I love contributing to the open-source community and building something other developers can use.

UX Unification of Wagtail

Last year, I spent my summers bringing Ben Enright's designs to life. During the project, I applied shared **Django templates** across the whole CMS, worked with the Python backend, **modified React components** and functions, created new ones, and added

components to the Wagtail **pattern library** for smoother testing. I also did some **heavy unit testing with Jest, Typescript and Python Testing**. As a result, we ended up making Wagtail **more maintainable** and **accessible** than ever before. The details of the documentation and the project can be found in my blog post.
Link to my blog -
https://wagtail.org/blog/open-source-internship-ux-unification/
Ever since then, I have remained involved with Wagtail and helped in guiding and mentoring new contributors to Wagtail for **Outreachy**.

Contributions to Wagtail

I reflected my familiarity with the codebase and the tools required for this project through my contributions. I have made several pull requests to Wagtail and am still working.

In **wagtail/wagtail** repository:

PR Number	Description	Status
#9191	Fixed hover behaviour of Delete button	Merged
#9168	Applied new designs to disabled buttons	Merged
#9166	Applied new designs to secondary buttons	Merged
#9165	Removed yes button styles	Merged
#9153	Aplied new design for delete buttons	Merged

#9104	Refactored button styles	Merged
#9085	Centered spinner icon and signing in text for login page	Merged
#9080	Added buttons to pattern library	Merged
#9064	Fixed padding and close button bugs for chooser modal	Merged
#9044	Removed legacy unbutton styles	Merged
#9043	Added role equal to status to messages	Merged
#8993	Fixed console error for breadcrumbs	Merged
#8972	Keyboard support for sort menu order	Merged
#8944	Fixed empty variable issue in header template	Merged
#8925	Adopted new designs for the authentication pages(log in, password reset, etc)	Merged
#8922	Updated summary panels in Dashboard	Merged
#8920	Updated page listings header to new designs	Merged
#8904	Adopted non-collapsible breadcrumbs for root page	Merged

#8896	Updated help block colors	Merged
#8895	Added translate button to header actions dropdown along with some cleanup	Merged
#8870	Slim Header - redundant styles cleanup	Merged
#8862	Adopt shared header in Form submissions listing - 2nd approach	Merged
#8857	Adopted shared header for form submission page - 1st approach	Closed
#8854	Adopted shared header for menu settings editing view	Merged
#8850	Adopted shared header template in workflows	Merged
#8849	Adopted shared header for redirect page	Merged
#8839	Adopted shared header in reports view	Merged
#8831	Fixed typo in classname	Merged
#8799	Added prefix w- before header classes	Merged
#9792	Adopted slim header in page listing views	Merged
#8781	Adopted shared header template on the dashboard view	Merged

#8755	Remove old legacy breadcrumbs completely	Merged
#8743	Removed breadcrumbs from ModelAdmin	Merged
#8741	Clean up move page breadcumbs	Merged
#8705	Updated move breadcrumbs to use new design	Closed
#8702	Adopt new breadcrumbs within the page chooser modal	Merged
#8679	Added breadcrumbs next to explorer	Merged
#8764	Add new breadcrumbs to page explorer	Closed
#8666	Removed breadcrumbs next's reliance on data-slim-header	Merged
#8652	Add tab nav link and tabbed interface to pattern library(storybook)	Merged
#8638	Added new breadcrumbs to Pattern Library	Merged
#8626	Adopted new Tabs in Workflow History Detail page	Merged
#8616	Fixed empty tabs showing to non admin users	Merged

#8608	Some minor modifications to make integration tests' documentation more clear	Merged
#8606	Added unit test for breadcumb	Merged
#8530	Removed trailing slash	Merged
#10160	Migrated breadcrumbs to stimulus controller	Draft
#8111	Removed upper casing from SCSS files.	Merged
#8113	Replaced focus-outline-on utility.	Merged
#8129	Removed redundant text-transform instances	Merged
#8130	Updated to the latest version of stylelint-config-wagtail	Merged
#8151	Added the contrasting logo for the dark theme.	Merged
#8174	Removed font inconsistency.	Merged
#8112	Excluded tests and maps from published artifacts.	Under discussion
#8167	Added display data for Twitter embed blocks and tests.	Merged
#8178	Added pagination to docs search results page.	Merged

#8385	I have added the full_url field for image renditions and updated tests and documentation.	Merged

In **wagtail/stylelint-config-wagtail** repository:

PR Number	Fixes	Description	Status
#16	#15	Added linting rule to block text-transform.	Merged
#17	#14	Enforced mixins as first in declarations.	Merged

In **wagtail/sphinx_wagtail_theme** repository:

PR Number	Description	Status
#178	Main nav made scrollable	Merged
#175	Added changelog for #167	Merged
#173	Setup stylelint and configured it to use wagtail stylelint-cofig	Merged
#168	Separated dev and non-dev dependencies	Merged
#167	Page toc made sticky and scrollable	Merged

#165	Updated to lockfileVersion 2	Merged
#136	Added rst-content class to body.	Merged
#137	Updated color for warning, note, and important.	Merged
#143	Added hover state to links.	Merged via #152
#150	Removed build of api docs.	Merged
#158	Deemphasised toc text.	Superseded by #159

LFX'23 Mentee at CNCF-Harbor for Spring term

LFX mentorship is an open-source internship program offered by The Linux Foundation. I have been accepted as a mentee to the **CNCF - Harbor: Harbor Robot accounts with full Harbor API access** mentorship. This project aims to give Robot accounts access to the full Harbor API. Primarily it's a DevOps project. Technologies I'm using in this project - **DevOps, Docker, Go, Javascript, Angular, Clarity and UX**.

Contributions to other open source projects

I've contributed to various crypto projects like **Specter Desktop**, **Rust-Bitcoin**, and **Fedimint** involving technologies like Python,

Bitcoin API and Rust language. I've also contributed to **Purr Data** which is an audio web library based on Pure Data written in C++. The complete list of contributions can be found in my **Github profile** contribution overview chart.

Built the institute's R&D website

I along with a team of 4 members, rebuilt our Institute's new RnD website. Technologies used - **HTML, CSS, Next.js, Tailwind, Vercel.** Link to project - **https://iitbhu-rnd-one.vercel.app/**

Postman Student Expert

I've been using Postman for years now for API testing. So I took up the Postman API Fundamentals to learn more about it, and at the end of the course, I aced the assignment and was awarded a badge. My learnings:

- How to use Postman to work with APIs
- Making GET, POST, PATCH, and DELETE requests
- Query parameters, bodies, headers, response codes
- Basic scripting
- API Key Authorization
- How to use APIs in your applications

Motivation

Why do I wish to participate in the Google Summer of Code?

I started contributing to OSS during Hacktoberfest 2021, and since then, I've contributed to many organizations. Working with so many people, getting reviews, improving my skills and knowledge base, and interacting with them was delightful, which wouldn't have been possible without open source. GSoC provides an excellent opportunity to work on amazing open-source projects impacting thousands of people and receive mentoring from senior developers.

Why do I wish to work with Wagtail in particular?

I've been with Wagtail for over a year now, contributing to it and being a part of this community. I love the uplifting and helping nature of the people here, and I'm grateful for everything I learned here, especially from LB and Thibaud. I'm always keen to work with the Wagtail team on some exciting projects, and GSoC is one such opportunity. Doing GSoC with Wagtail is an excellent opportunity to dive deep into the CMS and enhance my Django and web development skills.

Why this particular project?

I am passionate about projects that have a considerable impact, especially those concerning climate impact. As a developer, I want to develop energy-efficient apps to help reduce the overall digital footprint. The programs we write are everywhere, and everybody wants more software. A program designed by a software engineer is executed many times. Either because it has been used for several

years or because many people use it. For instance, Wagtail is used by thousands of developers, and the websites developed using Wagtail are being visited millions of times. Had we designed Wagtail to be just a bit more energy efficient in the first place, the effects would already be huge today. A single line of code can have a tremendous impact. That's why I want to play my role in making Wagtail greener and helping build a sustainable future for all.

Operating systems I work with?

I primarily work with **Ubuntu** in a virtual machine for development purposes. At the same time, I also have Windows 11 as the base operating system, which lets me tests programs on both OS.

Availability

I have my **official college summer break from the first of May to the end of July**, and during the official GSOC Period, I have no other commitments for the summer, so I will be able to devote a minimum of **35-40 hours per week**. I will also be accessible full-time during weekends after my semester starts in August and keep the community updated about my progress.

Why me?

I'm a versatile developer working with DevOps, blockchain and web2 technologies yet my most incredible skill is ***figuring things out along the way***. I know the required technologies, and as I shared some of my accomplishments above, I've hands-on working experience with all of those. I'm fun to be around and good at communicating and expressing my idea. My passion for open source and never give up attitude when it comes to software development sets me apart from others. I've always felt that working in open source and helping in its development is my way of doing good for society but through this project, I'll not only be able to give back to the Wagtail community that I love so much but to the planet as well. I am excited about the opportunity to work on this project and will work as hard as I have to make this project a grand success.

Research material and References

- Is There a Sustainable Internet? How "Green" is WordPress? (raidboxes.io)
- Lean WordPress: A guide to optimizing your CMS | Opensource.com
- Sustainable Drupal: 10 ways to save energy by speeding up your CMS | Opensource.com
- Sustainability | Drupal.org
- A sustainable web build for The Climate Group | Drupal.org
- Developers Can Save The Planet, Part I: The Problem (marmelab.com)
- Developers Can Save The Planet, Part II: The Solution (marmelab.com)

- Digital Carbon Footprint: The Current State of Measuring Tools (marmelab.com)
- GreenFrame.io: What is the carbon footprint of a web page? (marmelab.com)
- Web performance | MDN (mozilla.org)
- Responsive images - Learn web development | MDN (mozilla.org)
- Lite YouTube Embeds - A Better Method for Embedding YouTube Videos on your Website - Digital Inspiration (labnol.org)
- Sustainable UI - Carbon Hack 22 @ TAIKAI
- This website is carbon aware - Fershad Irani
- Making this website carbon aware - Fershad Irani
- Carbon-Aware vs. Carbon-Efficient Applications - Sustainable Software (microsoft.com)
- Installing Django Debug Toolbar - LearnWagtail.com
- Performance — Wagtail Documentation 5.0a0 documentation
- Performance and optimization | Django documentation | Django (djangoproject.com)
- How can we improve the sustainability of our digital products (torchbox.com)
- gsoc/project-ideas.md at main · wagtail/gsoc (github.com)
- Greener coding - Making a 'gold' reference configuration with the Wagtail bakery app · wagtail/wagtail · Discussion #8843 (github.com)