Introduction

Summary

This set of documents makes the case for updating the Django homepage. It is written almost entirely by Adam Hill, so if there are factual inaccuracies, they are his fault.

Please comment on this doc, continue the conversation in the original forum post, contact me directly on Mastodon, or via email. I look forward to your feedback, constructive criticism, and participation to help improve Django's homepage and overall marketing positioning.

Why Now?

The current website design has served the framework well. It is clean, readable, and flexible to handle different parts of the site – from the homepage to documentation, the same top navigation, optional secondary navigation, main content body, and sidebar gets used everywhere.

However, this design is now more than 10 years old – a lot has happened over the past 10 years! Not just changes in Django and Python, but in the broader web framework ecosystem. Previously, most web frameworks were open-source and their sites were primarily designed for a smaller audience of technical developers. Now, the audience for modern web frameworks has exploded as software continues to "eat the world" and the number of developers has dramatically risen.

Usage of Python itself has increased substantially over the past 10 years (from 3.8% to 25% based on the TIOBE index, with more mind-blowing stats here if you are curious). Django *should* be poised to ride that wave of enthusiasm, but the homepage would need to clearly articulate the features and benefits. Currently, a lot of Django's main features are hidden behind a few clicks and large amounts of technical documentation.

In addition, there are now frameworks that have "big tech" corporate-backing and/or are funded by venture capital. These frameworks take a much more "marketing-first" approach focused on selling their frameworks. Unsurprisingly, this approach works – developers, CTOs, and CIOs are customers just like everyone else. If Django wants to thrive in this new environment, it also needs to sell itself.

Beyond marketplace adoption, there are the very necessary fundraising needs of the framework. The Django Fellows have been instrumental in ensuring stability, security, and innovation. Currently, sponsoring companies are not on the homepage or GitHub repository readme at all. Improving where and how Django displays sponsors – especially in the places which already get substantial traffic – will result in a virtuous circle for Django and the sponsors. I believe that being more deliberate about where to show company logos will have a material effect on sponsorships and fundraising going forward.

Personally, I do not think Django will suddenly die if the homepage is not updated tomorrow. However, I do think that its market penetration will continue to erode if Django does not change its approach to marketing itself. If we want Django to continue to expand and for more people to adopt it, updating the homepage is required.

Who Am I?

I have been using Django professionally for over 10 years at The Motley Fool. 10 years ago, I (along with 3 other developers) led the technological transformation from a proprietary Microsoft platform (C#, ASP.NET, MS SQL Server, IIS) to an open source platform (Python, Django, PostgreSQL/MySQL, NGINX/gunicorn). That initiative was based on an evaluation of multiple web frameworks and is still a decision I am very proud of today.

At my day job, I have built, launched, and maintained highly trafficked content sites, multi-million dollar-generating order pages, portfolio tracking with ML-assisted recommendations, and onboarding workflows for $2B+ AUM portfolios – all with Python and Django.

Outside of work, I have been involved in the Django community in numerous ways. I have released numerous Python and Django open source libraries with the most popular being Django Unicorn with 2.5k+ stars on GitHub. I co-host a (semi-infrequently) updated podcast about Django called Django Brew, co-created stickers for Django, have participated in Djangonaut Space as a navigator, been to multiple DjangoCon US conferences, gave 2 lightning talks at DjangoCons, and recorded a talk for DjangoCon EU when it was online-only.

Outside of Django, I am also a keen follower of other web frameworks. Mainly Ruby on Rails, Laravel, and Phoenix – server-side-focused frameworks which solve similar use cases to Django. I have also built and supported applications built with ASP.NET, FastAPI, Bottle, Flask, Express, Koa, and NextJS over the years.

All that to say, while a lot of information in this packet includes my opinion and could be considered subjective, it is based on a long history of using and working with multiple web frameworks, participating in the community, and online discourse about Django.

Other Data Points

Other research and prior art was consulted in the creation of this document.

- 20tab UX research (12/2023)
  - djangoproject.com - 20tab deliverables
  - https://thib.me/django-website-user-research-report-by-20tab summary
- Annual Impact Report (2024): https://www.djangoproject.com/foundation/reports/2024/
- Current style guide (last updated 2014): https://www.djangoproject.com/styleguide/

Other Discussions

- Forum post: https://forum.djangoproject.com/t/want-to-work-on-a-homepage-site-redesign/42909
- Mastodon thread: https://indieweb.social/@adamghill/115238415537915047
- Bluesky thread: https://bsky.app/profile/adamghill.com/post/3lzefuxalj22r

Homepage Goals

Summary

Ideally a redesign would combat some of the prevailing (incorrect) sentiments about Django in the marketplace and allow it to compete with other web frameworks.

(Incorrect) Django Perceptions

- Old and/or not modern
- It's dead and/or not updated
- Slow performance
- Boring (in a bad way)
- Only useful for backend REST APIs
- Conversely, only used for templated sites, bad for APIs
- The documentation is overwhelming or complicated
- Search doesn't find what I'm looking for, i.e. use Google to search the docs
- Doesn't scale
- Hard to find jobs that use Django

Homepage Goals

The most important goal of the homepage is that there needs to be continuous, iterative improvements. Staying static for 10 years at a time is not acceptable.

- Django is vibrant, active, and modern
    - Last release
    - Last PRs that got merged?
    - Metrics
        - Number of downloads from PyPI
        - Forum posts count
        - Commit count
        - Number of third-party libraries
        - Etc, etc.
- Major (especially unique!) features should be highlighted
    - Database ORM
    - Admin
    - Management commands
    - Robust third-party ecosystem
    - Url routing
    - Caching
    - Multiple supported template languages
    - Both sync and async functionality

    - Instead of having to piece a bunch of different libraries together, Django is “batteries included” and everything is designed to work well with each other (and if needed, the pieces can be swapped out)
- Major use cases
    - Server-side websites
    - REST/GraphQL API backend for SPA or mobile app
    - Interactive websites with “your choice” of frontend framework
    - Background worker task processing
    - CMS (either headless or not), either completely custom or with the help of Django CMS, Wagtail, CodeRed, etc.
- Prominently feature current sponsors and how other companies can sponsor Django
- Showcase impressive companies that currently use Django (or have used it in the past) for social proof, e.g. Eventbrite, Instagram, Dropbox, Mozilla, NASA, The Guardian, etc.
- Django is secure and has a robust security posture
- Django is “mature” which actually means stability, i.e. not that it has stopped innovating
- Top navigation should only include top tasks
    - Maybe mega menus or dropdowns would allow more links without cluttering top nav? Ala ExpressJS or Laravel
    - Add search to the top navigation so that people can quickly look for information
        - Might require improvements to search so that it’s not a frustrating experience
- Perhaps a video for people to catch people’s attention
    - Per Thibaud: Make room for a “Django 6.0 features highlights” marketing video
- Metrics and analytics to understand funnels, what pages are getting traffic, what gets clicked, what people are looking for, etc.
- Highlight community events, DjangoCons, etc
    - Maybe this is just mixed in with news?
- Djangonaut Space, Django Girls, etc.
- Governance, open source, foundation

Non-negotiables

- Needs to be as accessible as the current site

Open Questions

- What is the best way to combat the perception of slow performance?
    - Benchmarks?
- What is the accessibility standard? WCAG 2.2 AA?

Competitor Analysis

Summary

This list of frameworks and usage percentage is from this article however I only included server side frameworks.

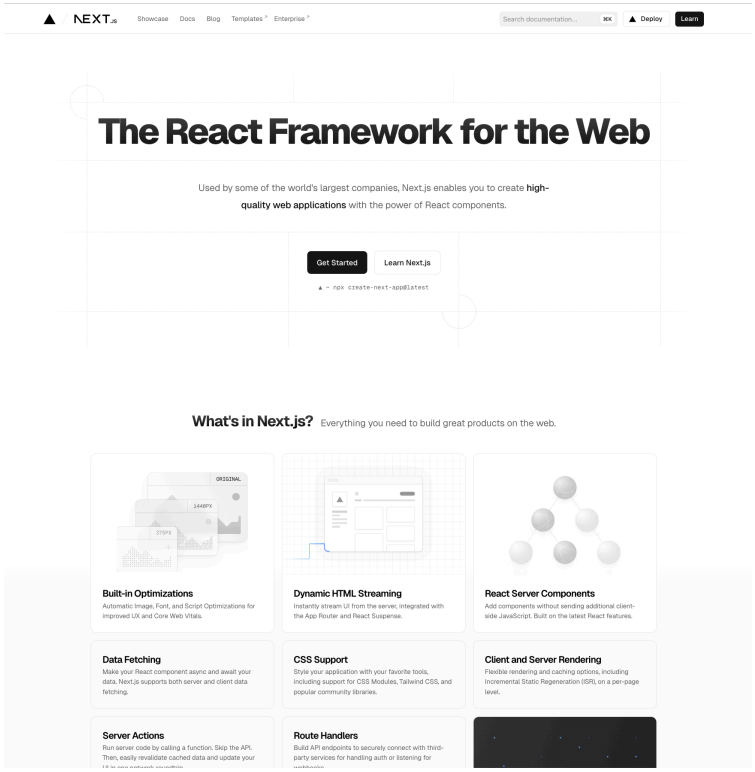
NOTE: The usage numbers here might not be completely accurate, but they are included for some measure of relevance. GitHub Stars are also not a perfect proxy, but are included as another proxy for popularity.

There are more detailed notes and my take about each homepage for the highlighted frameworks below this table. My commentary is in *italics*.

	Usage	**GitHub Stars**
NextJS	17.9%	134k
Express	17.8%	67k
ASP.NET Core	16.9%	37k
ASP.NET	12.9%	
Flask	12.9%	70k
Spring Boot	12.7	78k
Django	12%	85k
Wordpress	11.8%	
FastAPI	9.9%	89k
Laravel	7.9%	82k
NestJS	5.8%	72k
Ruby on Rails	4.7%	57k
Nuxt	3.6%	58k
Symfony	3.2%	30k
Astro	3%	53k
Phoenix	1.9%	22k
Drupal	1.9%	

	Usage	GitHub Stars
Remix	1.6%	31k

NextJS



H1: The React Framework for the Web
H2: Used by some of the world's largest companies, Next.js enables you to create high-quality web applications with the power of React components.

The current 800 pound gorilla in the room when it comes to server side web frameworks. Lots of marketing weight by Vercel which is huge in the startup space and gets a ton of traffic and interest because of its close tie to ReactJS.

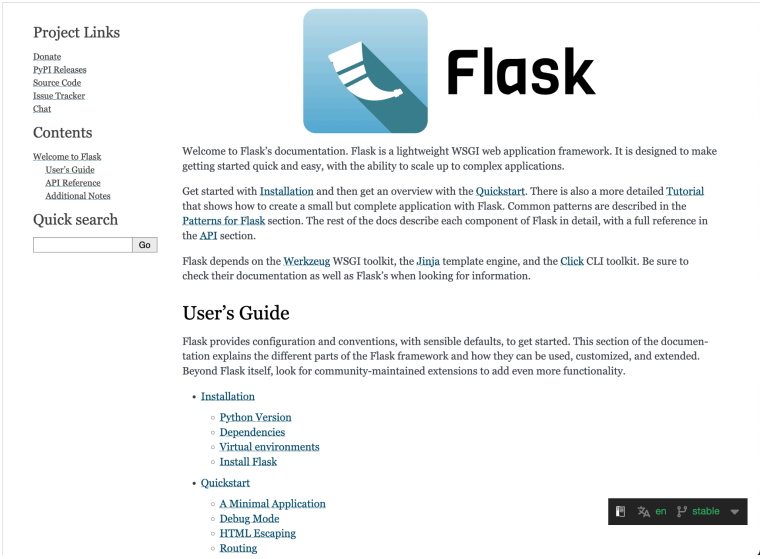
Top navigation

- *Showcase (link)*
- *Docs (link)*
- *Blog (link)*
- *Templates (link)*
- *Enterprise (link)*
- *Search (input)*
- *Deploy (button)*

- *Learn (button)*

Feels light and modern with nice pops of color. Includes some animations for visual interest (although it's probably a little too flash for my taste). They have a clear list of features and clear CTAs for their cloud hosting business. Explicitly have a "Showcase" (almost like white papers?) for social proof and also testimonials from known tech companies for more social proof.

Flask



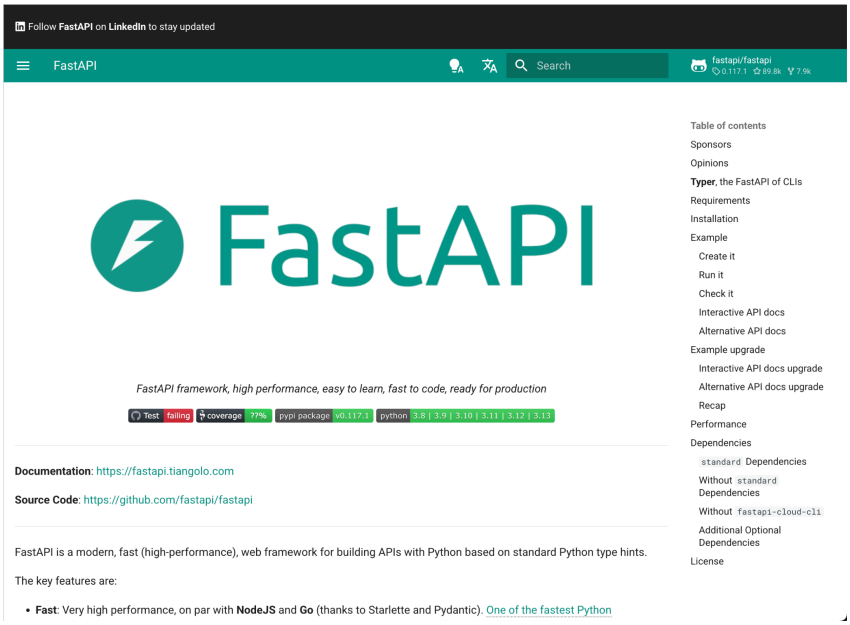
Standard readthedocs-style site.

Sidebar navigation

- *Donate (link)*
- *PyPI Releases (link)*
- *Source Code (link)*
- *Issue Tracker (link)*
- *Chat (link)*
- *Welcome to Flask (link)*
    - *User's Guide (link)*
    - *API Reference (link)*
    - *Additional Notes (link)*
- *Search (input)*

Seems more tailored more toward developers and a lot of words. Search is available in the sidebar for quick access which is a nice touch.

FastAPI



Standard mkdocs-style site with an attention-grabbing alert bar at the top that changes between different messages to follow FastAPI on different social media platforms.

H1: FastAPI framework, high performance, easy to learn, fast to code, ready for production

Top navigation

- *Light/dark mode (toggle)*
- *Language (dropdown)*
- *Search (input)*
- *GitHub link with stats*

Sub navigation

- *FastAPI (link)*
- *Features (link)*
- *Learn (link)*
- *Reference (link)*
- *FastAPI People (link)*
- *Resources (link)*
- *About (link)*
- *Release Notes (link)*

The homepage has these sections:

- *Key Features*
  - *Fast*
  - *Fast to code*

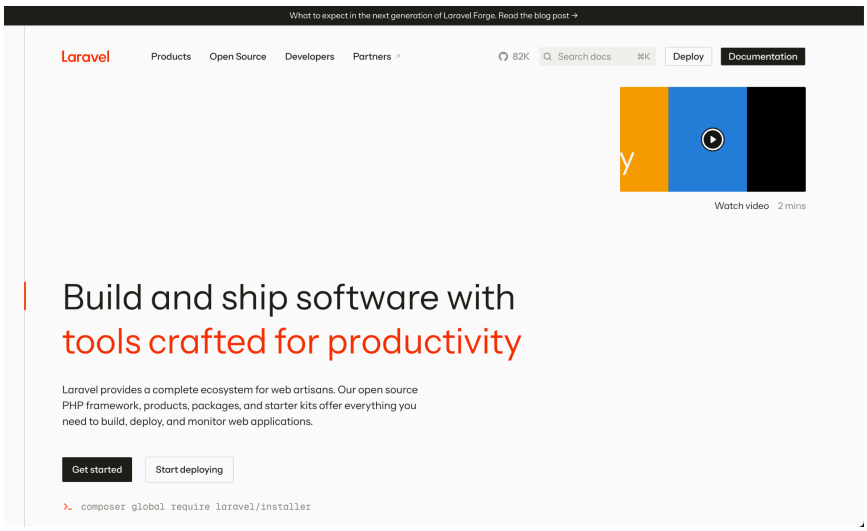
    - *Fewer bugs*
    - *Intuitive*
    - *Easy*
    - *Short*
    - *Robust*
    - *Standards-based*
- *Sponsors (currently empty?)*
- *Opinions (these are testimonials from people at big tech companies and OSS people)*
- *Typer (this seems out of place)*
- *Requirements*
    - *Starlette*
    - *Pydantic*
- *Installation*
- *Example upgrade (adding routes)*
- *Performance*
- *Dependencies*
- *License*

FastAPI hits a lot of the key parts that I think Django's homepage should strive for: call out key features first, show sponsors, and have testimonials. They interestingly also include screenshots of installing the library, example code to start up, the Open API interface, and instructions for making updates to the code.

Personally there seems to be too much on this homepage, but I'm guessing it works ok for them. Key differentiators: it's fast, look how easy it is to start with, look how easy it is to add new functionality, and look at this UI you get out of the box.

It's also one of the few sites which has stats from GitHub (current tag, star count, forks) which I thought would be more prevalent since putting that in the top nav would be a form of social proof.

Laravel



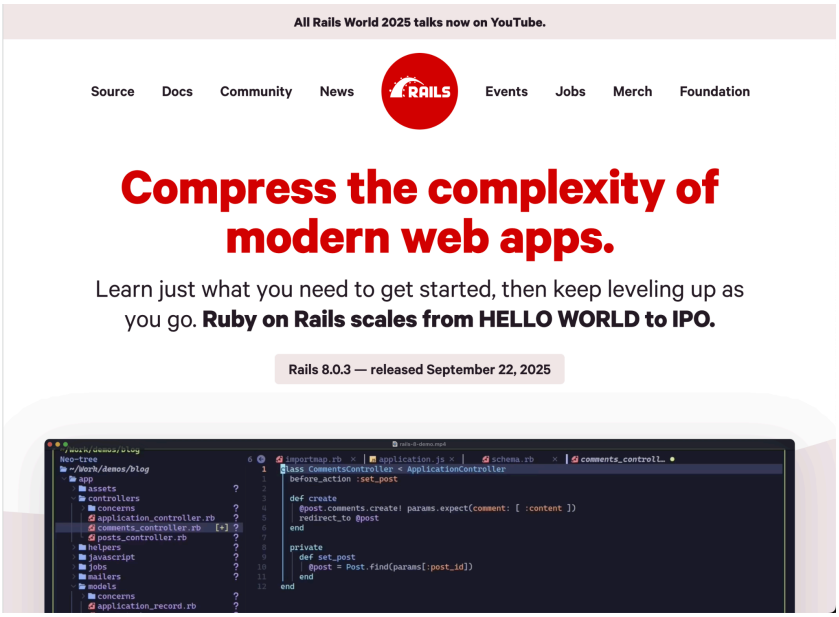
H1: Build and ship software with tools crafted for productivity
H2: Laravel provides a complete ecosystem for web artisans. Our open source PHP framework, products, packages, and starter kits offer everything you need to build, deploy, and monitor web applications.

Top navigation

- *Logo*
- *Products (mega menu with Cloud, Nightwatch, Forge, Nova)*
- *Open Source (mega menu with framework, starter kits, packages)*
- *Developers (mega menu with documentation, resources, events)*
- *Partners (link)*
- *GitHub logo and star count*
- *Search docs (input)*
- *Deploy (button)*
- *Documentation (button)*

Video that starts small and takes up the whole screen as you scroll down is a little distracting IMO. Lots of social proof of companies. Long lists of official packages, starter kits, and products. Interesting approach to show example code by splitting up features into backend and frontend sections with rollover tabs for features of both. Lots of testimonials with people's faces – more social proof.

Ruby on Rails

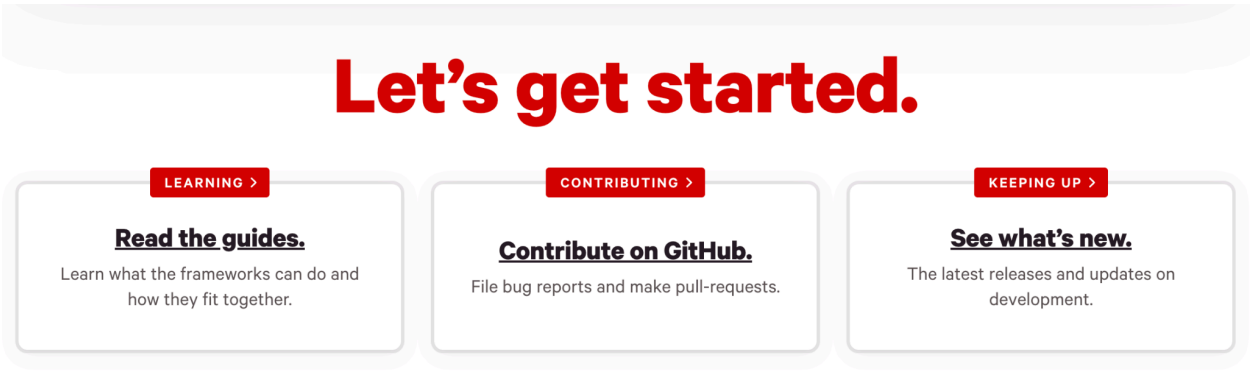


H1: Compress the complexity of modern web apps.
H2: Learn just what you need to get started, then keep leveling up as you go. Ruby on Rails scales from HELLO WORLD to IPO.

Top navigation

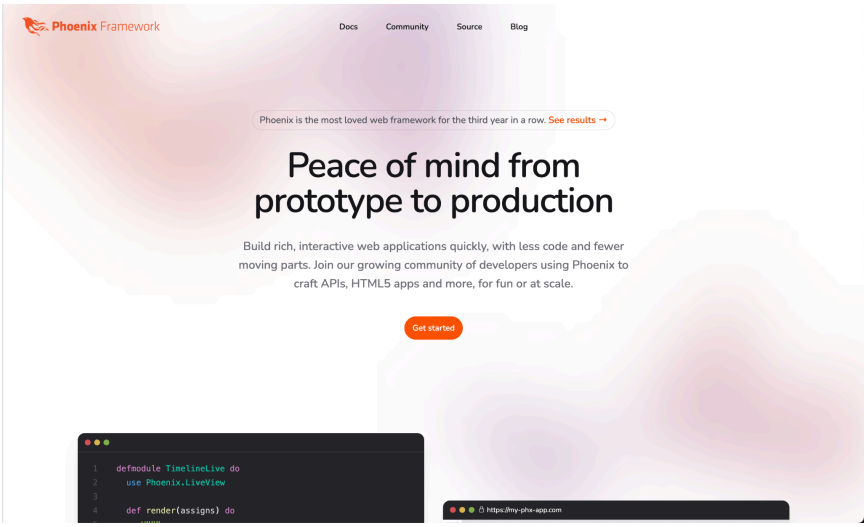
- *Source (link)*
- *Docs (link)*
- *Community (link)*
- *News (link)*
- *Rails (logo image)*
- *Events (link)*
- *Jobs (link)*
- *Merch (link)*
- *Foundation (link)*

Red color scheme is definitely attention-grabbing. Giant 30 minute video that builds a blog after the H2 – I'm curious how many people watch that, but it's probably more than I expect. Lots of social proof with company logos (maybe too many IMO?). Code examples to show how certain features look is definitely useful.



Bottom CTAs titled "Learning", “Contributing”, “Keeping Up” are nice and straight-forward.

Phoenix



H1: Peace of mind from prototype to production
H2: Build rich, interactive web applications quickly, with less code and fewer moving parts. Join our growing community of developers using Phoenix to craft APIs, HTML5 apps and more, for fun or at scale.

Top navigation

- *Phoenix Framework (logo image)*
- *Docs (link)*
- *Community (link)*
- *Source (link)*
- *Blog (link)*

They definitely lean into the interactive HTML and Liveview with large example code + gif and a YouTube video. CLots of call outs of unique features of the framework. Large grid of companies

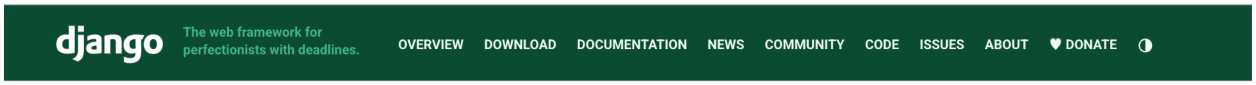
for social proof. Maybe the simplest homepage (not counting the “just docs” ones), but it still feels designed.

Homepage Analysis

Summary

An analysis of the current homepage with some commentary about each section. My commentary is in *italics*.

Navigation

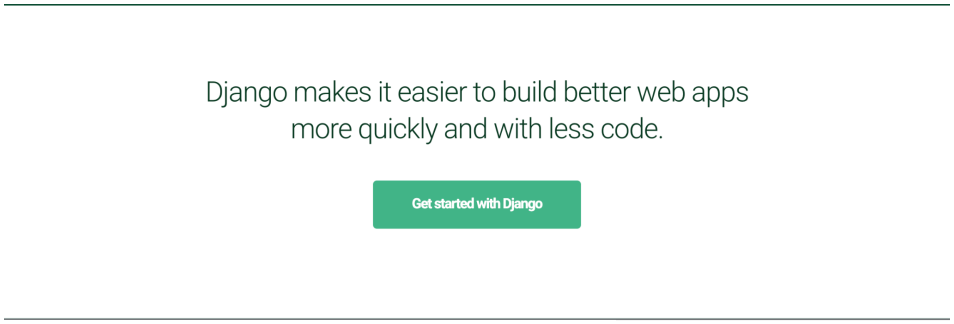


- "Django" green background
    - *I don't see a reason to move away from the "Django" green for bits of color here and there. I would tone down the different shades of green on other parts of the page and go to a more neutral color palette in general – something that feels more in line with modern website trends and professional, but not bland.*
- Logo (image)
- Tagline (image)
    - *Remove from the top navigation. It's a good tagline, however it takes up space on the navigation that would be useful for other things. Make it the H1 on the homepage, but it doesn't need to be in the top navigation.*
- Overview (link)
    - *Remove this link altogether. The information on the overview page is good, but it should all be on the homepage – devs should not have to click a link to see that it is secure and scalable, for example.*
- Download (link)
- Documentation (link)
- News (link)
- Community (link)
    - *I like the idea of this, but having a whole page for it seems like overkill. Maybe this could be a dropdown menu instead?*
- Code (link)
    - *Having a link to the code makes sense, but it could just be an icon for GitHub? Everyone will know what that means, saves some space, and brings some visual interest to the top navigation, so it isn't just a lot of words. Also, the only link that opens in a new window, so it feels a little out of place. I would expect a visual indicator icon for links that open in a new window.*
- Issues (link)
    - *This doesn't feel like the right place to me. It's probably not a top task for most of the people going to the homepage, although having it available somewhere seems important.*
- About (link)
    - *Not sure having this as a top-level navigation is required, however this information feels important. If it stays in the navigation, maybe change the label*

    - *to “Foundation” or “Governance” or something similar. “About” makes it seem like it’s about Django the framework, when it’s actually about the DSF.*
- Donate (link)
- Light/Dark mode (button toggle)

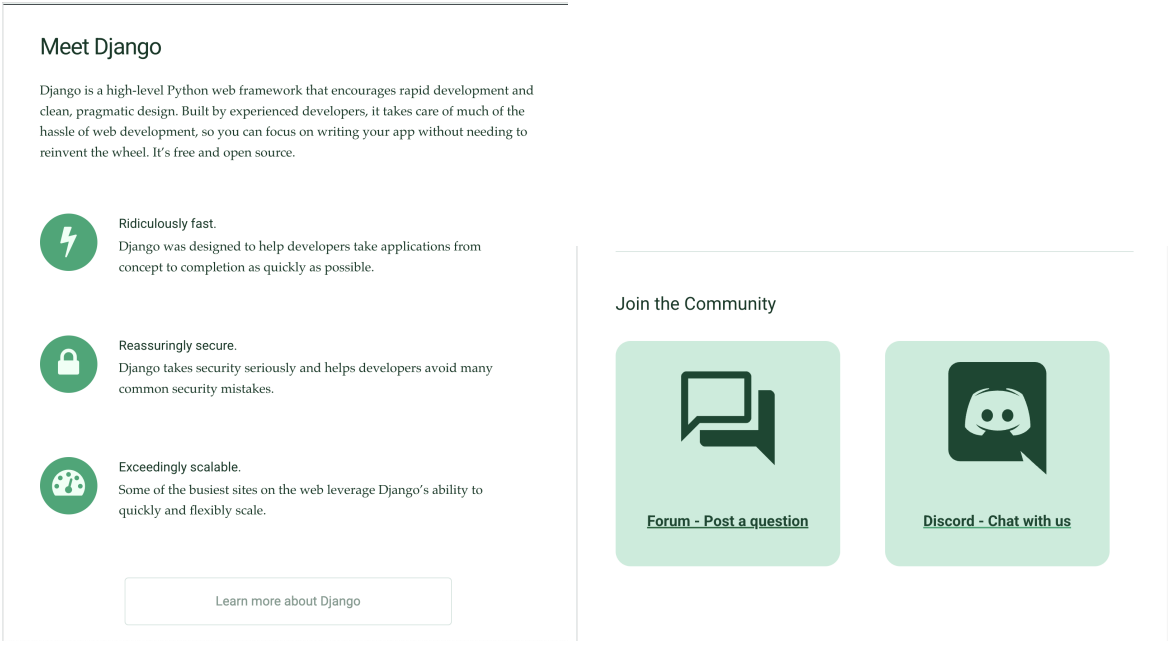
In general, the navigation has too many links and feels very busy. What are the top tasks for people? Maybe consolidating some links into dropdowns would clean up some of the clutter? Can we have a search bar in the top navigation?

Hero



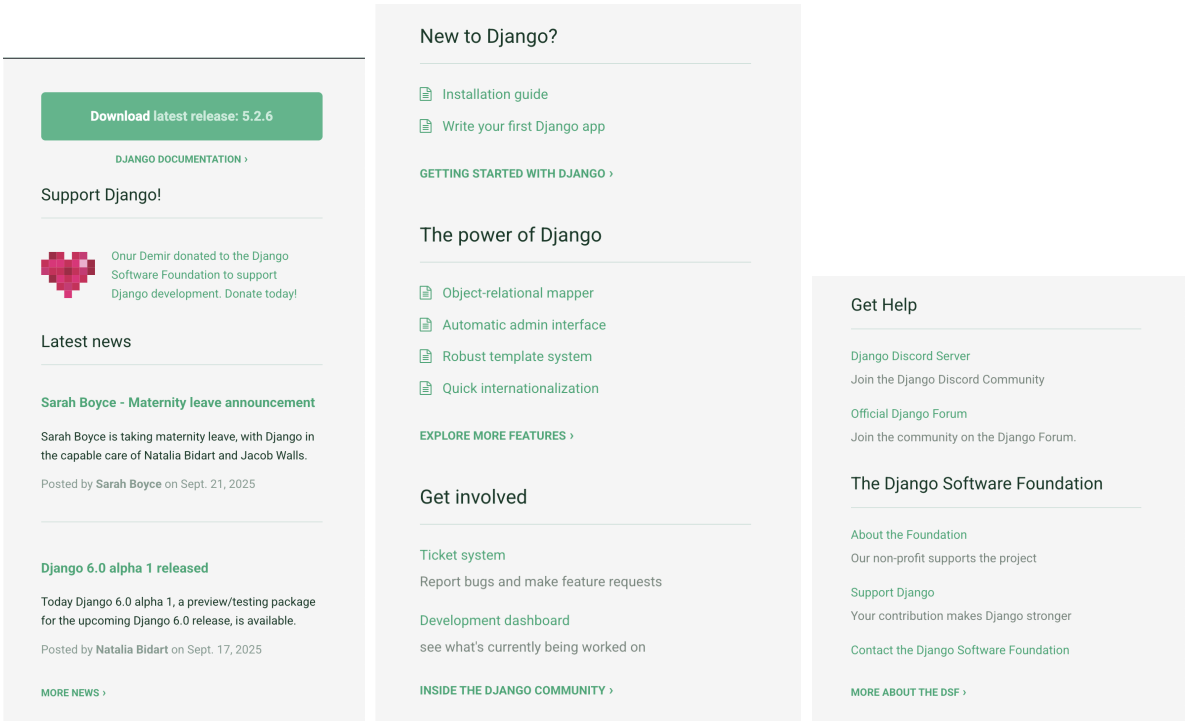
- “Django makes it easier to build better web apps more quickly and with less code.” H1 tagline
    - *I like the gist, however it’s wordy and clunky. Personally, I think “The Web Framework for perfectionists with deadlines” is catchier. If we wanted to stay in the same vein of this current H1, maybe something in the active voice and just give the benefits like* “Quickly build better web apps with less code” might be punchier
- Get started with Django (button)
    - *Shorten to “Get started” or “Get started today” to create urgency*

Body



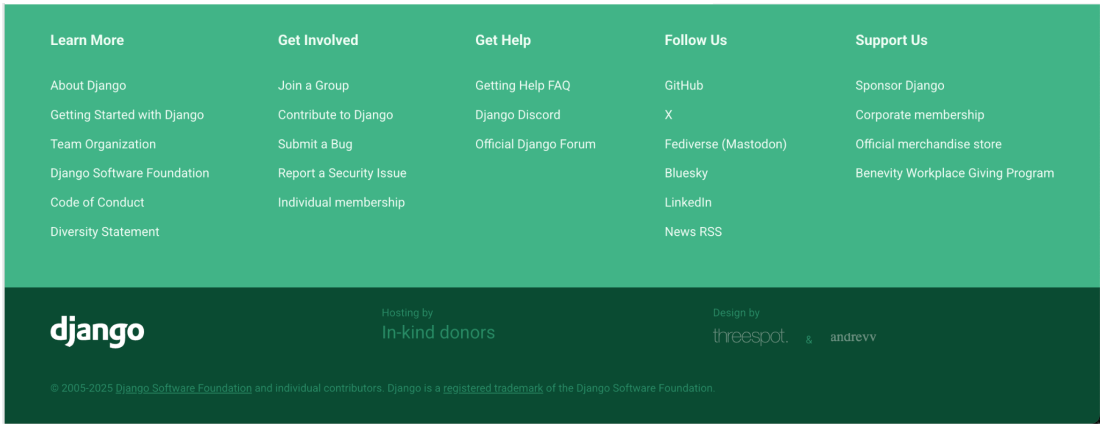
- *The gist here is fine, but it's wordy and does not say enough about Django's features or benefits. It should be punchier with more visuals. This is a technical audience, so show some code, but keep it short and sweet.*
- *Remove all of the green accents for the links to the community*

Sidebar



This is all good information, but it's a lot. I like the big download CTA, however it doesn't stand out from the "Get started with Django" above. The support Django heart is a nice splash of color on the page, though. I also think some of it "The power of Django" should be front and center on the homepage, but it's buried in this very long sidebar with a billion other links.

Footer



Looking at the footer, I realized that the Discord and Forum are linked 3 places on this page (body, sidebar and footer)? Also realized that social media profiles are all linked in the footer and nowhere else – if the hope is for developers to follow the socials, they should be more

prominent. Also missing is an email newsletter that devs could sign up for to get updates for new releases, security fixes, etc.