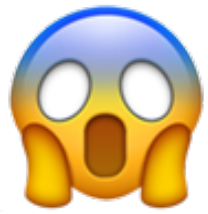
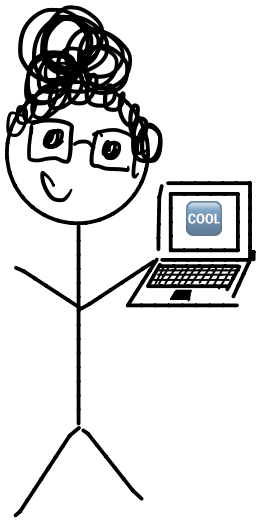
How to DEPLOY your personal project Witnout wanting to



by Galen E. Corey

So you've worked hard and built a cool project to showcase your skills!

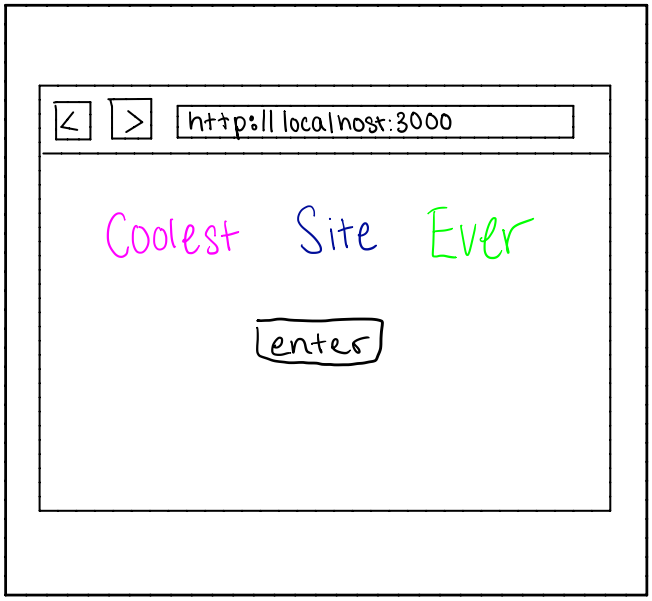


Amazing!

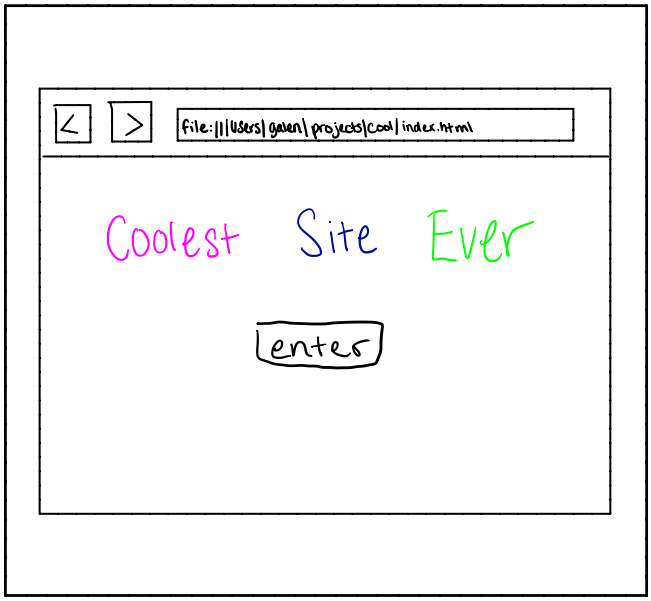
now you're ready to

* show your friends
* tell your grandma
* tweet it out for the world
* update your resume
* Get your work OUT THERE!

But... how?

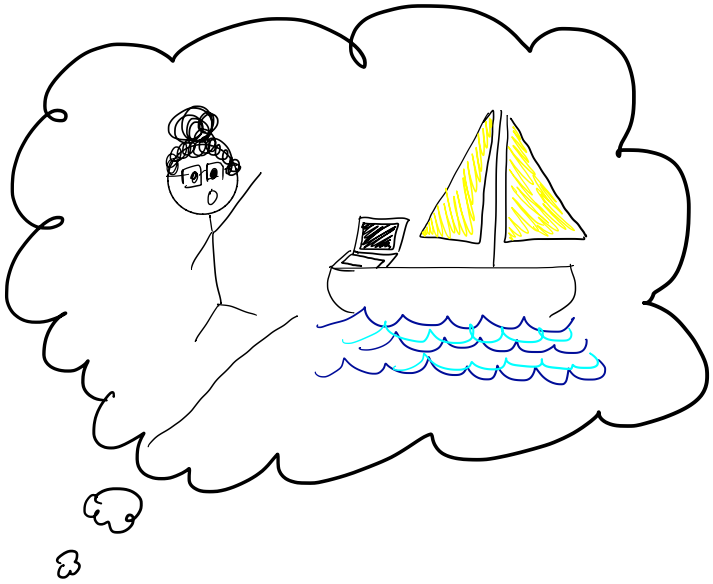


Maybe you're serving up your site locally?



... Or maybe you are viewing an html file on your machine?

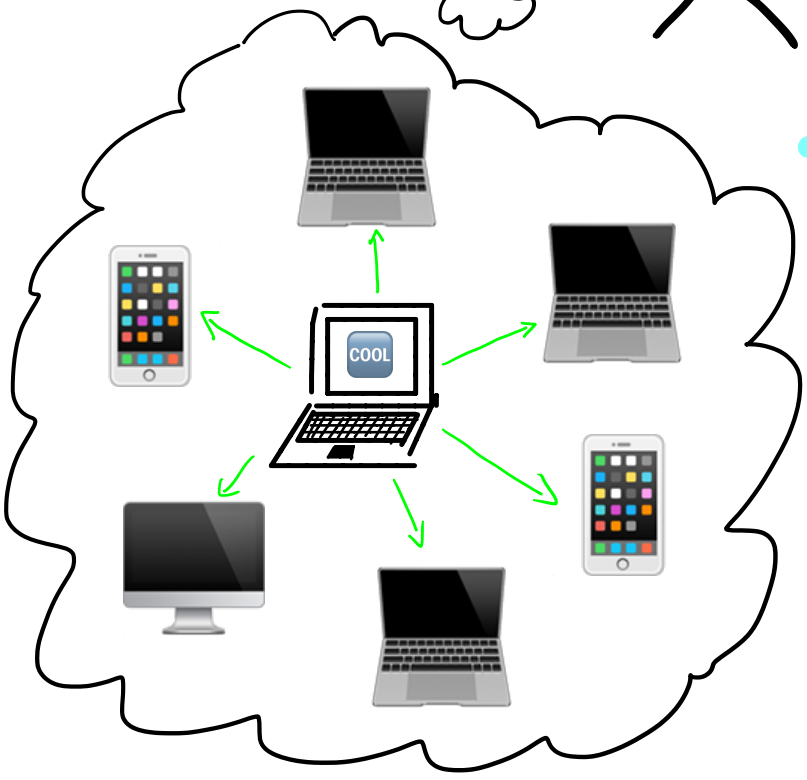
Now you need to...
Ship your work out there, into the Actual Internet.



OR

To put it a different way,

You need to let the Actual Internet have access to the machine your site is hosted on.



Luckily

There are plenty of services out there designed to do exactly this!

Option #1

Github will host a static site for you!

Github Pages

Great For:

- 👍 Simple sites like a portfolio page.
- 👍 Sites that are mostly vanilla HTML, CSS and JS.
- 👍 Documentation or project pages.

Not Great For:

- 👎 Sites that require server code.
- 👎 Sites larger than 1gb.
- 👎 Sites that need more than 100gb of bandwidth (you expect a lot of visitors or are streaming/hosting large files)

Every github account gets 1 user page and unlimited project pages.

Setting Up Your User Page

1. Go to github.com and create a new repository called myusername.github.io (using your gh username).

2. Add your content to the repository. Github will automatically serve whatever is in the index.html file in the root folder.

3. Once you have pushed your project up to github, you can view it at https://myusername.github.io.
   - ↳ any changes to your master branch will be updated automatically.

All Set!

Setting Up Project Pages

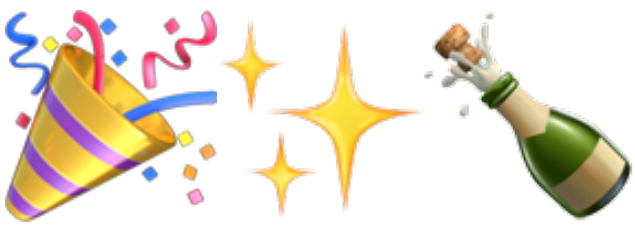
1. You can create as many project pages as you want. Start by making a repository. (it can have any name.

2. Create a new branch called `gh-pages` and push it to github. By default, github will serve up the `index.html` file on the branch with this name.

3. Once your code is on github, you can view it at:

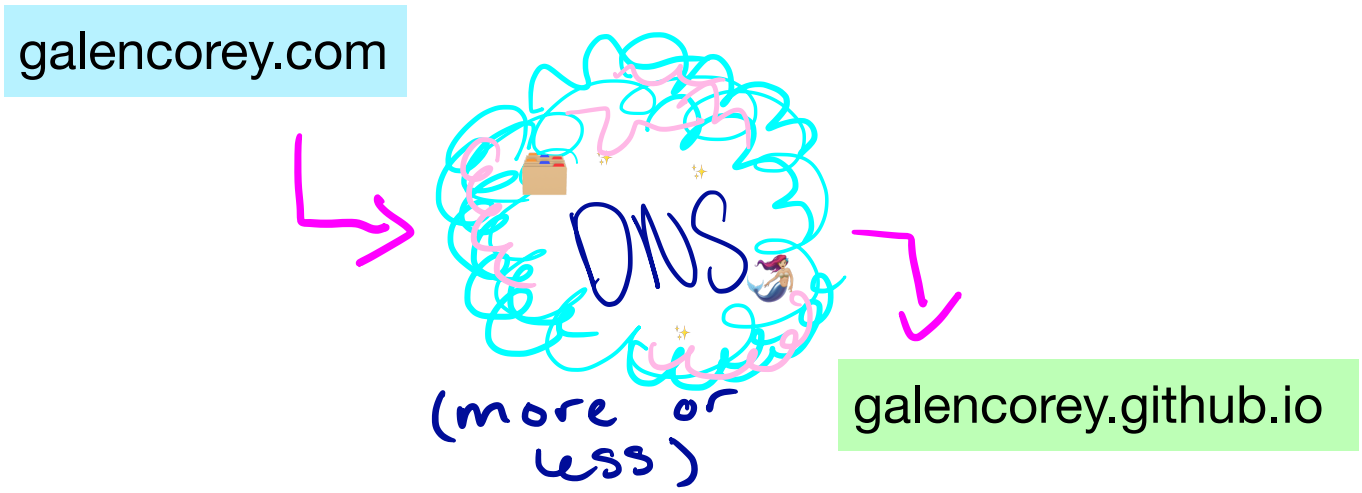
   https://myusername.github.io/myprojectname

(where the project name is the name of your repository).



a note about custom domains

You can buy a domain name and set it up to point to your github user page. For example, my user page is a portfolio site and you can find it at galencorey.com.



That means all my project pages live at galencorey.com/project-name. This is because github sets up project pages as subdirectories of the user page.

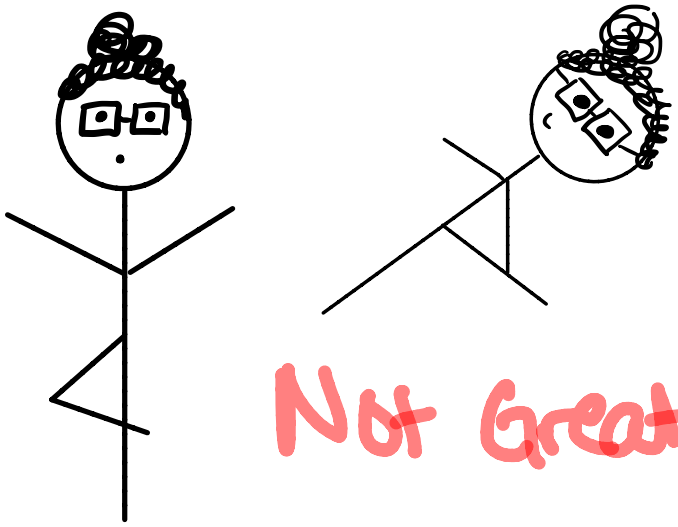
If you want to host projects at different domains, you might want to read on!

Option #2

Netlify

It's as easy as github pages...

But with more flexibility!



Great For:

- 👍🏽 Static sites **OR** sites that only require a basic form.
- 👍🏽 Sites that require a build step (i.e. gulp, webpack, etc.)
- 👍🏽 Easily implementation of features like A/B tests and user auth.

Not Great For:

- 👎🏽 Sites that require complex server code (although there is support for aws Lambda functions).

Not a lot of other cons, netlify is a pretty great hosting service for static sites that is pretty darn easy to use!

Hosting your static site with Netlify

1. Netlify works with github, so start by setting up a repo for your project (if you don't already have one).

2. Go to app.netlify.com and make an account/login (I like the "login with github" option).

3. Select "New Site From Git", and follow instructions to find the repo of the project you want to deploy.

4. You can specify the git branch you want to deploy, a build script, and other settings before you click deploy.

5. Grab a celebratory snack and wait for your site to go live!!! 🍫🍪🍦

Option #3

Heroku

Great For:

- Getting your full-stack app out there quickly
  - * they support node, ruby, python, java, and more!
- A personal project that isn't expecting a ton of traffic (they offer paid tiers of service for apps that need better performance.)

Not Great For:

- Running a very large application as inexpensively as possible.
- Having complete control over the build of your app.
- People who hate magic. 🧐

Hosting a basic full-stack app with Heroku

1. In the root of your project, create a file with the exact name Procfile. Add your app's start script to the Procfile. A Procfile for a rails app might look like:

```
web: bundle exec rails server -p $PORT
```

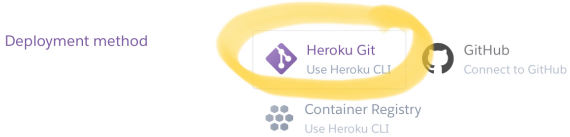
* If you are building a node app, you can skip this step as long as your package.json has a start script.

2. Make sure your server is listening on the port specified in the $PORT environment variable.
   * In node, you can access this from process.env.PORT. You can also add your own environment variables once you set up the app.

3. Create an account at id.heroku.com and select "new" to start a new app. From here, there are several ways to proceed. I am a fan of using the Heroku CLI, because it's great for debugging your build.

Heroku, cont.

4 Select "The heroku git CLI option.



5 Follow instructions on the page to deploy your app for the first time!

Pro Tips:

- You can easily deploy any future commits by running `git push heroku master` from the command line. That's it!
- View your server logs by running `heroku logs --tail`

There is so much more you can do - check out the docs!

You Did It!

To Summarize:

- For a portfolio or simple personal project, try Github Pages.
- For other serverless sites, try netlify.
- If your project needs a server, check out Heroku.