

Lesson

Weekend

Intermediate JavaScript (/intermediate-javascript)

/ Team Week (/intermediate-javascript/team-week)

/ Hosting a Webpack Project with GH-Pages

Text


In this lesson we'll learn how to host our webpack project on **GitHub Pages**, also referred to as **gh-pages**. GitHub Pages is free, and lets us host our HTML, CSS, and Javascript right from the repository. It's a great way to quickly host a web app, and since it's integrated into GitHub, it's a nice option for developers. You may have hosted some projects this way by creating a `gh-pages` branch in your project, and while we'll end up with the same result here, the process for setting it up is slightly different. We'll make some edits to our `package.json`, and run some new commands from our terminal. In this lesson we'll also cover how to deploy a React app built with webpack for students in the React course.

Before we start, one important note about GitHub Pages is that it is **not meant** to host sensitive information, like API keys. The GitHub docs explain more:

GitHub Pages is not intended for or allowed to be used as a free web-hosting service to run your online business, e-commerce site, or any other website that is primarily directed at either facilitating commercial transactions or providing commercial software as a service (SaaS). GitHub Pages sites shouldn't be used for sensitive transactions like sending passwords or credit card numbers.

In other words, any information we put in our code will be available to someone visiting our site. If we want to host a site with sensitive information we would need some kind of backend, or server-side code, to store our sensitive information. For now, that is beyond the scope of what we'll do.

To read more about prohibited uses for gh-pages, visit the following documentation:

-  **Link to GitHub Pages Docs**
(<https://docs.github.com/en/pages/getting-started-with-github-pages/about-github-pages#prohibited-uses>)

Update the `package.json`

First, we'll add a `repository` key to our `package.json`. We're adding it under the `version` key but the order doesn't matter as long as it is in the top-most level of our package object, the same level as `name` and `version`. After we add that, let's set it equal to an object with a `url` key. Use the following code snippet as an example, replacing `{REPONAME}` with your repo and `{USERNAME}` with your GitHub username respectively.

```
./package.json
```

```
{
  "name": "project name",
  "version": "1.0.0",
  "repository": {
    "url": "git+https://github.com/{USERNAME}/{REPONAME}.git"
  },
  ...
}
```

Next we'll add two new commands to our `scripts` object:

- `predeploy` builds our site and bundles it in the `dist` folder.
- `deploy` pushes the contents of that folder to a new commit on the `gh-pages` branch, creating that branch if it doesn't already exist.

./package.json

```
{
  ...
  "scripts": {
    "build": "webpack --mode=development",
    ...
    "predeploy": "npm run build",
    "deploy": "gh-pages -d dist"
  },
  ...
}
```

Install gh-pages Package

Install the `gh-pages` package via the terminal by running the following command in the root of your project's folder:

```
$ npm install --save-dev gh-pages
```

You should see `gh-pages` listed in your `package.json` under `devDependencies`.

Deploy the site

At this point we can run the following command to deploy our site. Make sure to run this command in the root of your project's folder.

```
$ npm run deploy
```

This will build our project and then publish it to GitHub Pages. You may be curious where the `predeploy` script gets used if we aren't using it ourselves. Well, `gh-pages` runs the `predeploy` script automatically before deploying the site to ensure there's an up-to-date build of our project. We can look in our terminal to see exactly which scripts are executed and in what order. Our terminal should show us that the `predeploy` script is run, which itself calls the `build` script, all before the `deploy` script executes. Check out the code snippet below that shows an example of the terminal output for deploying. Note that 'PROJECTNAME' will be the name of your project as listed in the `package.json` file.

```
$ npm run deploy

>PROJECTNAME@1.0.0 predeploy
> npm run build

> PROJECTNAME@1.0.0 build
> webpack --mode=development

Hash: e6e0a675ea138d2edcd8
Version: webpack 4.46.0
Time: 2553ms
...

> PROJECTNAME@1.0.0 deploy
> gh-pages -d dist
```

After we've successfully deployed our site, it will be hosted at a URL like this, where {USERNAME} is your github username and {REPONAME} is the name of your repository :

```
https://{USERNAME}.github.io/{REPONAME}
```

If we want to update our live site with some changes, we'll need to `git checkout` to the branch that has the most up to date code, then run our deploy script. Typically we deploy from the `main` branch but it's possible to deploy from any branch. Each time we deploy, another commit gets made on our `gh-pages` branch. The commit message will be "Updates" by default, if you want a custom commit message you can specify it by using the `-m` option.

```
$ npm run deploy -- -m "Deploy site with new colors"
```

Finally, note that `deploy` can take a minute or two to update, so be mindful of that when checking the live site.

For React Projects

The process is almost identical for deploying a React site to GitHub Pages. We'll still update our `package.json`, install `gh-pages`, and add our new scripts. The difference between this and basic webpack hosting is in the `deploy` script. We need to configure GitHub Pages to deploy the contents of the folder that contains our bundled code, and in a React project that folder is called `build`, as opposed to `dist` in our basic webpack projects. Check out the code snippet below that shows how we've updated the value of the `deploy` script to point to the `build` folder.

./package.json

```
{
  "scripts": {
    ...
    "predeploy": "npm run build",
    "deploy": "gh-pages -d build"
  },
  ...
}
```

Outside of that the steps are exactly the same. Optionally, see this repo for a walkthrough of deploying a React project.
(<https://github.com/gitname/react-gh-pages>)

Further Exploration

Configure a Publishing Source

It's possible to configure our GitHub Pages site to publish when changes are pushed to a specific branch, or we can write a GitHub Actions workflow to publish our site. So for instance we could set our `main` branch as the publishing source and have GitHub

automatically deploy any time changes are pushed to the `main` branch. See this page in the GitHub docs for how to set a custom publishing source. (<https://docs.github.com/en/pages/getting-started-with-github-pages/configuring-a-publishing-source-for-your-github-pages-site>)

Enforce HTTPS

By default GitHub enforces HTTPS protocol (not HTTP), so you may get errors if your links use HTTP. For the deployed site you'll need to remove any HTTP content. See this page on enforcing HTTPS in GitHub (<https://docs.github.com/en/pages/getting-started-with-github-pages/securing-your-github-pages-site-with-https#resolving-problems-with-mixed-content>) for more information.

Un-Publish the Website

If you want to remove a site from GitHub Pages so it is no longer hosted, you can find instructions for un-publishing here. (<https://docs.github.com/en/pages/getting-started-with-github-pages/unpublishing-a-github-pages-site>)

And that's it! Our site should be live on GitHub Pages for the whole world to visit.

[Previous \(/intermediate-javascript/team-week/practicing-the-git-workflow\)](#)

[Next \(/intermediate-javascript/team-week/backend-course-preparation-software-installation\)](#)

Lesson 7 of 13

Last updated more than 3 months ago.

[disable dark mode](#)



© 2023 Epicodus (<http://www.epicodus.com/>), Inc.