

Lesson

Tuesday

Intermediate JavaScript (/intermediate-javascript)

/ Test-Driven Development and Environments with JavaScript

(/intermediate-javascript/test-driven-development-and-environments-with-javascript)

/ Creating a Template Repository

Text

When we create projects using an environment (with multiple packages managed by npm, and webpack to bundle our source code), we'll often start with the exact same files. It can be tedious to build all of these files from scratch, especially when we're building new projects every day.

Fortunately, we can create template repositories in GitHub which we can reuse (and modify) for future projects. In this lesson, we'll learn how to create a template repository in GitHub as well as how to use the template for other repositories.

Using GitHub Template Repositories

First, we need to create a repository that includes all the files we want. We'll keep this example simple. It will include just a README. On your computer, go ahead and create a directory called `template-repo` which includes the following `README.md` file.

```
# (Application Name)

#### (Brief Description of Application)

#### By (Your Name Here)

## Technologies Used

* _List all_
* _the major technologies_
* _you used in your project_
* _here_

## Description

## Setup/Installation Requirements

* _This is a great place_
* _to list setup instructions_
* _in a simple_
* _easy-to-understand_
* _format_

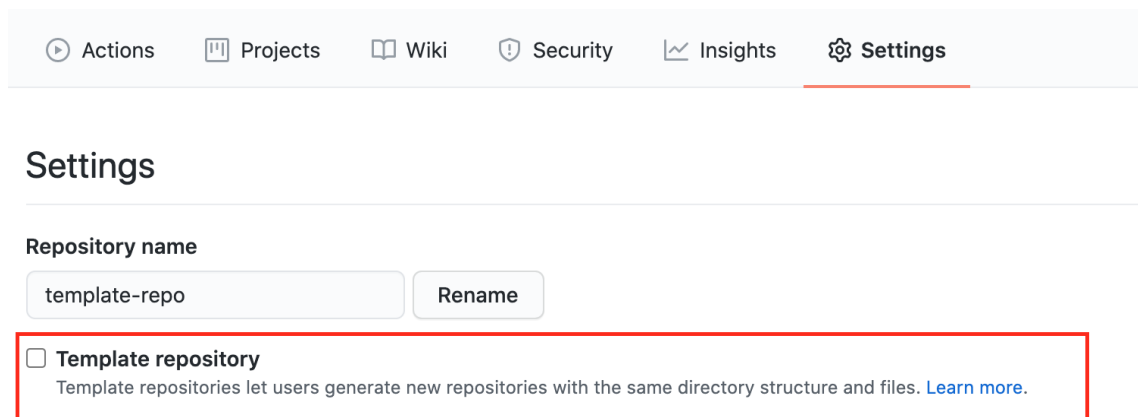
## Known Bugs

* _Any known issues_
* _should go here_

## License
```

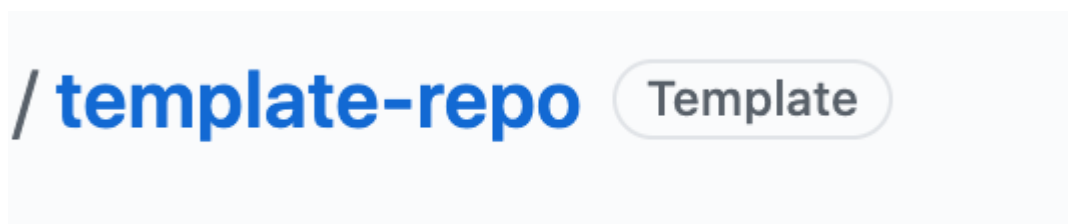
As we can see, this README contains headers for all of the items that are required in the READMEs for projects we create at Epicodus.

The next step is to commit the code, create a GitHub repo, and push your local code to the remote repo. Once that's finished, go to the repository in GitHub and click on *Settings*.



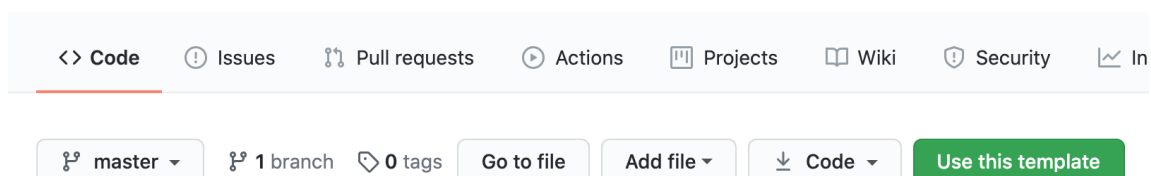
As the image above shows, once you click on *Settings*, you can click the checkbox marked *Template repository*. A little green checkmark will pop up by the box, which means that GitHub has auto-saved the update.

After checking the box, navigate back to the main page of the repository. You'll see it has a *Template* label by the name of the repository:



If there's not a *Template* label, you'll need to go back to the *Settings* and make sure you checked the box.

You'll also see that the repository now has a green *Use This Template* button.




If we want to create a new repository using this template, we just need to click the *Use This Template* button. We'll be taken to a page for creating a new repository. It looks almost exactly the same as the new repository page we've been using so far. The only difference is that we are now creating the repository from a template. We can create the repository as usual, but this time once the repository is created, all of the files from the template will be included in the new repository. Finally, we can clone the repository to our desktop and modify it as needed, changing the README or any other files it already has such as configuration or code files.

Creating a Template from Shape Tracker

Using a template repository is especially helpful for projects that include boilerplate environments such as the ones we've been building in this section. We can easily create a template repository that includes a `package.json` file, a webpack configuration, and any other configuration files such as what we might need for Babel or ESLint.

Now that you've created a template repo for a README, the next step is to create a template for a project that uses webpack. We'll be using a very similar JavaScript environment in the next section so you'll be able to reuse this template then, too.

You can base your template off of the Shape Tracker repo (linked below) or a project you created in this course section. If you decide to use the Shape Tracker project repo, make sure to use the branch titled `3_multiple_business_logic_files` as your point of reference. As needed, review the lesson on accessing code from different branches (<https://www.learnhowtoprogram.com/intermediate-javascript/object-oriented-javascript/accessing-code-from-different-branches>).

 **Example GitHub Repo for Shape Tracker**
(https://github.com/epicodus-lessons/section-5-shape-tracker/tree/3_multiple_business_logic_files)

Your template repo should have the following files copied over from your reference project:

- `.gitignore`
- `.eslintrc`
- `package-lock.json`
- `package.json`
- `webpack.config.js`
- `.babelrc`
- `src/index.js`
- `src/css/styles.css`
- `src/index.html`
- `README.md`

You can also include directories such as `__tests__` and `src/js`, though you'll need a file in any directory for Git to be able to track it.

You can also optionally include boilerplate code such as the structure for Jest tests.

When you create your template repository, take the time to add thorough installation/setup instructions to your README. This should include information on how to run the project:

- Installing all packages with `$ npm install`.
- Building the project using webpack with `$ npm run build`
- Starting a development server with `$ npm run start`
- Linting JS files in the `src` folder with `$ npm run lint`
- Running tests with Jest using `$ npm run test`

Also, make sure you update the *Technologies Used* section of your README, too! There's been a host of new tech that we've used in our projects in this course section.

Template repositories can make it much easier to get started on a project. To really utilize a template repository effectively, you need to get it right. That's why it's worth taking a little extra time to put

together a great template README now. If the README in the template repository is mediocre, chances are it will lead to mediocre READMEs in other projects, too, and a lot of additional work to get those READMEs in shape later down the road.

You aren't required to use template repositories for your independent projects but you may do so if you wish. We recommend updating your template repositories regularly and also creating new ones for different kinds of projects. For instance, you'd have different templates for React projects than you would for a vanilla JavaScript project.

One final recommendation — while we do recommend getting specific about details such as installation steps, you don't want to make your templates so specific that they aren't reusable. If you find yourself regularly deleting content from repositories that have been built from a template, that's a sign that the content in the template isn't serving its purpose; you want to minimize the amount of boilerplate you need to add from scratch, but you also want to avoid having to delete a lot of boilerplate, too.

For more information on creating template repositories and reusing them, see the following GitHub documentation:

- Creating a template repository
(<https://docs.github.com/en/free-pro-team@latest/github/creating-cloning-and-archiving-repositories/creating-a-template-repository>)
- Creating a repository from a template
(<https://docs.github.com/en/free-pro-team@latest/github/creating-cloning-and-archiving-repositories/creating-a-repository-from-a-template>)

Previous (/intermediate-javascript/test-driven-development-and-environments-with-javascript/working-with-multiple-business-logic-files)

Next (/intermediate-javascript/test-driven-development-and-environments-with-javascript/haiku-creator-rpg-sudoku-solver-two-

day-project-part-1)

Lesson 40 of 49

Last updated more than 3 months ago.

disable dark mode



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