iGEM Wiki Starter Pack

Release 0.0.0

Pranav Ballaney

Oct 26, 2020

CONTENTS

The iGEM Wiki Starter Pack is the easiest way to create your iGEM Wiki.

Please head over to the *Overview*, *Installation* instructions or *Usage Guide* to get started.

CHAPTER

ONE

FEATURES

1. Built-in theme that
    a) Looks great on all devices
    b) Comes with a dark mode
2. Markdown support
3. Automatic uploads with WikiSync
4. Extract citation information from DOI
5. Automatic table of contents on each page
6. Endless customization with Webpack
7. Included common web development libraries
    a) Bootstrap
    b) jQuery
    c) MathJax
    d) Font Awesome
8. Extensive templating using Pug
9. Reset styles on iGEM.org

CHAPTER

TWO

CONTRIBUTION AND COLLABORATION

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

Please visit our Contributing page to find out how you can help make this project better.

Using this software or submitting issues and pull requests can count towards a collaboration for our teams. Please give us a shoutout at @igem_bits on Instagram if this Starter Pack has made your iGEM experience easier! For contibuting to this software or discussing further collaborations, please reach out to us at igembitsgoa@gmail.com.

CHAPTER

THREE

CONTENTS

3.1 Overview

The iGEM Wiki Starter Package is a template that contains everything needed to build an iGEM wiki and then some. Instead of reinventing the wheel and starting from scratch every year, teams can expand on this base to kick-start their wiki building process.

The default out-of-the-box design of the package is clean and modern, scales to mobile devices, comes with a dark mode, and works great with screen readers and the like. All wiki content can be written in plain English (using Markdown) instead of pure HTML, which has poor readability and code to text ratio. Design is also completely separated from content while editing the starter pack; this lets teams customize pages and write content simultaneously, and prevents the workflow from being bottlenecked at either step.

In addition to simplifying content and design, the package includes common web development libraries like Bootstrap (layout), jQuery (JavaScript), MathJax (mathematical notation), and Font Awesome (icons) that add functionality to the wiki. It also features utilities that automate time-consuming tasks like adding citations to your wiki. The title, author, and other publishing information of an article can be retrieved directly from its DOI and included at the end of the page.

The package comes built in with WikiSync, a Python software that uploads the entire wiki to the iGEM servers at once with a single command. This eliminates the need to manually name and upload each file, replace each URL, and copy paste the source code for each page into their respective editors. WikiSync integrates effortlessly into automation workflows like GitHub Actions and Travis CI, so content added to the team's wiki repository on GitHub is automatically uploaded to the iGEM servers every time.

Please head over to the *Installation* instructions or *Usage Guide* to get started.

3.2 Installation

Table of Contents

Note: The iGEM Wiki Starter Pack requires **git, Nodejs, Python and pip** to be installed. Please make sure you have a working installation of all three before starting here.

3.2.1 Quick Start

Installation:

```
pip install copier
copier gh:igembitsgoa/igem-wiki-starter wiki
cd wiki
npm install
pip install -r requirements.txt
npm start
```

Git setup:

```
git init
git add --all
git commit -m "Initial commit"
git remote add origin <your Github repository URL>
git push --set-upstream origin master
```

3.2.2 Detailed Guide

Note: The iGEM Wiki Starter Pack requires **git, Nodejs, Python and pip** to be installed. Please make sure you have a working installation of all three before starting here.

The iGEM Wiki Starter Pack is a Copier template. To set up the starter pack, first install `copier` by executing the following at the command line:

```
pip install copier
```

Now, set up the starter pack by running:

```
copier gh:igembitsgoa/igem-wiki-starter <wiki-folder-name>
```

It now shows the following prompts:

- `iGEM_official_team_name`
- `iGEM_team_name`
- `year`
- `author`
- `email`

Next, go inside the directory that you created and execute the following to install Node dependencies:

```
npm install
```

Finally, to install Python dependencies, run this command:

```
pip install -r requirements.txt
```

Installation is complete now and you can start the webpack development server by running:

```
npm start
```

3.2.3 Updates

For you to recieve updates on the Starter Pack, it is necessary for your wiki to be a git repository. Please read the *Version Control* section to find out how to do this.

Before updating, make sure all your changes are either committed or stashed, leaving your working directory clean.

Then, execute:

```
copier update
```

in your wiki directory to update your Starter Pack installation.

Copier will ask you the same questions listed above, but your previous responses will be saved, so you can just press Enter at all the prompts.

It will then generate the updated files and ask you whether your existing files should be replaced by the new ones. If you're sure that you have not edited a file, you can safely overwrite it. If you answer *No* for any prompt, your file will be replaced with a `.rej` file with the same name, containing a `diff` of the old vs new version of the file. You can then decide which lines to keep.

3.2.4 Known Issues

Installation on Windows might require some additional effort.

1. **localhost**: `npm start` opens `0.0.0.0:8080` for the live server. This might not work on Windows. In this case, visit `localhost:8080` to see your live server.
2. **Emoji in Terminal:** Copier displays emoji in its prompts, which cannot be processed by Command Prompt and PowerShell. Please install the Windows Terminal from the Windows Store to overcome this issue.
3. **Python command issue:** in `package.json`, npm scripts have been defined. These are the commands that are executed when you run `npm <something>`. All Python commands are executed using the `python3` commands, which is the standard for Linux systems, and consequently, Github Actions or Travis CI. If you run Python using the `python` command, you might face some errors when running these commands. In this case, create additional tasks for your Windows system as shown below

```
"preprocess:win": "python utils/preprocess.py",
"nav:win": "python utils/nav.py",
"citations:win": "python utils/citations.py",
"custom_tests:win": "python utils/test.py",
"server:win": "webpack-dev-server --config webpack.development.js --open -
↪-host 0.0.0.0",
"start:win": "npm-run-all preprocess:win nav:win citations:win custom_
↪tests:win server:win"
```

Then, run `npm run start:win` instead of `npm start`.

3.2.5 Version Control

It is recommended that you set up version control for your wiki by creating a Github repository right away.

This will not only make development easier, but also allow you to try out your wiki on your github.io URL before uploading it to iGEM servers. Besides, you can set up Github Actions to automatically deploy your wiki directly from Github to iGEM servers. A detailed tutorial on this subject will soon be available here.

Setting up version control on your Wiki is necessary for you to recieve updates on the template. This will ensure that as we continue to add features to the Starter Pack, you will be able to integrate them into your wiki.

To set up Git for your wiki, create an empty repository on Github and set it up by executing the following in the folder you have created:

```
git init
git add --all
git commit -m "Initial commit"
git remote add origin <your Github repository URL>
git push --set-upstream origin master
```

Please send us an email at igembitsgoa@gmail.com if you need any help with installation.

3.3 Initial Setup

3.3.1 Getting Started

Code Editors

Unlike traditional text editors (notepad, Word, etc.), code editors contain interface features and functionality that will facilitate and streamline the process of writing your wiki code. For this starter pack we recommend using VSCode because it comes pre-packaged with version control, and console access.

Collaboration

In order to be able to work on the wiki together with your team, we suggest you set up Git and GitHub right from the start. If you're unfamiliar with it, take a look at the *Version Control* section.

3.3.2 Separation of Concerns

The Starter Pack tries to organize your code into sections, so that each section focuses on just one aspect of the wiki. This makes maintaining it easier, since your team will be able to work on different parts in parallel. It also reduces duplication — for instance, even though elements like the navigation bar appear on each page, they have to be written only in one place, and can be included everywhere else.

Like any other website, HTML, CSS and JavaScript files are used for markup, styling and user interaction. Describing these is beyond the scope of this documentation, but here are some resources you can check out to learn these languages.

In order to separate content from design, the starter pack uses Pug templates. Pug is a markup language, just like HTML, but adds several powerful features that simplify your code. While working with this starter pack, you would be working only with Pug (no HTML) and it will eventually be converted to HTML. This HTML would then be uploaded to iGEM servers and browsers would be able to display it. This is covered in more detail under Building and Deployment.

Similarly, the starter pack uses SCSS as a replacement for CSS. SCSS is an extension of CSS, so it's written in the exact same way as CSS, but it brings more features like variables, templates and mixins that help organize the code better. Just like Pug is converted to HTML, SCSS is converted to CSS before it can be uploaded.

3.3.3 The Development Server

To see the starter pack in action, run the following command to start a local server:

```
npm start
```

This will soon open a browser window at http://0.0.0.0:8080 where you'll see the homepage of your wiki.

On Windows, you might need to manually visit http://localhost:8080 instead of http://0.0.0.0:8080.

The homepage is built out of *src/index.pug* so you can go and start playing around with that file. As soon as you make a change and save the file, the page in your browser will automatically refresh and you can see your change there.

src/index.pug will be described in more detail in a later section.

3.3.4 Building and Deployment

The starter pack uses Webpack for bunding the assets under `src/`. All the files in the `src/` folder are compiled and bundled into HTML, CSS and JavaScript in a folder called `dist` that can be finally uploaded to iGEM servers.

To do so, run:

```
npm run build
```

outside the `src/` folder. A folder called `dist` will be created with HTML, CSS and JavaScript files.

These can be directly uploaded using WikiSync. A Python script called `wikisync.py` comes with the starter pack and can be used without any changes.

WikiSync requires your credentials to be stored as environment variables called `IGEM_USERNAME` and `IGEM_PASSWORD`. More information about this is available with the documentation for iGEM WikiSync.

After exporting these environment variables, run:

```
python3 wikisync.py
```

to run WikiSync and deploy your wiki to iGEM servers.

3.4 Usage Guide

We've split the Usage Guide into distinct segments according to the organization of a typical iGEM team.

3.4.1 Project Structure

The Starter Pack includes the following files and folders:

```
.github/
src/
utils/
.gitignore
.travis.yml
package.json
package-lock.json
README.md
requirements.txt
webpack.common.js
webpack.development.js
webpack.production.js
wikisync.py
```

The `src`, `utils` and `.github` folders contain several files and folders as well, which will be discussed in later sections. So many files and folders might seem overwhelming at first, but this setup will make your life much easier and you'll get used to it in no time.

Two more folders will be created here as you work on your wiki.

1. The `dist` folder contains distribution code, which is HTML, CSS, JS and media that has been created by combining all the files spread across various templates in the `src/` folder. Creating this folder will be described later, but just keep in mind that this is the folder that you can put on a server, not the `src` folder.
2. The `igem` folder will contain HTML, CSS, JS and media that has been processed specifically for iGEM servers. This folder is created by WikiSync and it is the contents of this folder that are finally uploaded to iGEM servers.

We will first talk about the `src` folder, and then gradually cover the rest.

The `src` Folder

The *src* folder contains the source files for your wiki. This includes HTML, Pug, CSS, SCSS, JavaScript, images, videos, fonts and everything else you want to add to your wiki.

It contains the following folders and files:

```
src/
        assets/
        citations/
        css/
        js/
        pages/
        templates/
        index.js
        index.pug
        nav.yml
```

1. `assets/`: Contains all media and documents. Everything other than code.
2. `citations/`: Files corresponding to those in `pages/` in case citations are required there.
3. `css/`: CSS and SCSS code.
4. `js/`: JS code.
5. `pages/`: Pug files that generate pages like `/Description` or `/Design`.

6. `templates/`: Pug files that contain code common across all pages, such as navbar, footer and a template for each file in `pages/`.
7. `index.js`: Entry point for Webpack. Leave untouched if unfamiliar with it.
8. `index.pug`: Homepage
9. `nav.yml`: Navigation menu contents. `utils/nav.py` generates a dictionary that is used to create the navigation on each page.

Templates and Pages

The first line of `src/pages/Sample.pug` is:

```
extends ../templates/contents.pug
```

This means that `Sample.pug` "extends" `contents.pug`. In this way, all files in `pages/` are based on the `contents.pug` template.

`templates/contents.pug` and `pages/Sample.pug` are described with comments here. Please leave a comment there in case any clarification is required.

More information about Pug templates is available here.

3.4.2 Contents

Structure of a Regular Page

Since an iGEM wiki is mostly for documenting a research project, almost all the pages of your wiki will have the same general theme. The homepage and a few others might be different, but we'll come to those later.

Visit the Sample page at http://0.0.0.0:8080/Sample and then take a look at the source code for it in `src/pages/Sample.pug`. This has also been described with comments here.

You'll notice correspondence between the code and the page, but also notice that all the details about formatting are neatly hidden away. The build system has been configured to generate all the surrounding code, so you only have to edit this text. Links, colors and animations can be edited in the CSS files.

The structure of `Sample.pug` looks something like this:

```
extends ../templates/contents.pug

block headVars
    - var title = "Sample"
    - var requireMathJax = true

block article

    :markdown-it(html)
        //- ...

//- DO NOT MODIFY THIS LINE AND ANYTHING BEYOND.

prepend citations
    - var citations = [...]
```

So there are six major sections, which are described in more detail below. The most important thing to notice is the indentation of various code blocks. Please keep that in mind as you read the guide.

extends ../templates/contents.pug As described above, this just says that `Sample.pug` is based on the `contents.pug` template.

block headVars This section contains the title of the page, a summary and a variable which denotes whether MathJax is required on this page or not.You can change these variables and use them as you like in `contents.pug`.

block article This is the main body of the page, where all the content lives. The structure of this section follows markdown syntax, which has been described in the next section.

//- DO NOT MODIFY THIS LINE AND ANYTHING BEYOND. This section will contain citations taken from the `citations/` folder. This is described later and can be ignored for now. Everything you add manually in this section will be overwritten with citations. If you create any custom elements, make sure they're above the `DO NOT MODIFY` line.

Markdown Reference

Our wiki generator fully supports markdown, a simple markup language which allows you to focus on the content while writing, hiding away all the formatting and styling details.

Here is an example of how Markdown can be used in the Starter Pack

```
block article
    :markdown-it(html)
        # Headings
        ## Level 2 Heading
        ### Level 3 Heading
        #### Level 4 Heading
```

HTML can also be written in this section:

```
block article
    :markdown-it(html)
        <mark>Hello</mark>
```

For more examples, please take a look at the Sample page of the template, while going through its code.

For full documentation of Markdown, visit the Markdown Guide.

To preview your text, visit Markdown Preview.

We have also defined some additional syntax that is included below.

Custom Syntax

This section covers some special syntax that doesn't come with markdown, but was made in order to extend its capabilities as per our requirements.

For a working example, please take a look at the Sample page of the template, while going through its code.

Images

Since the Starter Pack is built on Webpack, images cannot be added through the regular Markdown syntax.

Images can be added using the following syntax:

```
block article
    +image(1, "Description--header.jpg", "Caption", 100)
```

Notice that there is no `:markdown-it` block here. Images are added outside Markdown blocks, since this is not Markdown syntax.

The syntax used here is a Pug mixin, which is defined in `src/templates/mixins.pug`. The arguments are described below:

- **n:** Figure number, relative to the page.
- **URL:** Filename relative to `src/assets/img/`
- **Caption:** Please describe the image in a short phrase/sentence for screenreaders.
- **Width:** Width of the image as a percentage of the content body. Optional. Default: 90%.

The filename is automatically resolved to the relative path and another line is added above this mixin call. You can ignore this line and if you want to change the image after adding it, either change it in the `imgpath` line, or remove the `imgpath` line and change in the mixin call.

Definitions

Tooltip definitions are elements that show a popup on hover, containing a definition of the term highlighted.

Sample syntax is shown below. Notice the ~ (tilde) character.

```
:markdown-it(html)
    <dfn>Term ~ Definition</dfn>

    This can come <dfn>anywhere in ~ the text</dfn>.
```

Tables

Tables are made using the `markdown-it-multimd-table` plugin, so in order to create a table, you're required to indent one level back, mention the plugin, and then indent inside again.

```
:markdown-it(html)
    This is a regular paragraph, which precedes the table.
    When you want to insert a table, indent one level
    back and specify the plugin.
    Then indent inside again and start writing the table.

:markdown-it(html plugins ='markdown-it-multimd-table'])
| This is | the table | header row |
| ------  | --------- | ---------- |
| 1       | 2         | 3          |
| 4       | 5         | 6          |

[Table 1: Caption goes here.]
```

(continues on next page)

(continued from previous page)

```
:markdown-it(html)
    And when you're done, go back to the regular markdown filter.
```

More detailed examples are provided on our Sample page of the template (code).

Citations

The Starter Pack makes adding citations really easy with a custom `.yml` file.

If the article you want to cite has a DOI, the Starter Pack can directly pull data from the CrossRef database from the DOI, and you don't need to include anything else.

If it's an article without a DOI, a webpage, or a book, you will have to include all elements of the citation.

Citations for `pages/Description.pug` go in `citations/Description.yml`. This is illustrated in the Sample citations file. The rendered citations can be seen on the Sample page.

A couple of things to keep in mind:

1. Each citation must be numbered with a comment preceding the citation.
2. The citation type must be specified, and must be one of `doi`, `article`, `webpage` and `book`.
3. The line specifying the citation type must begin with a hyphen (`-`).
4. Indentation is important.
5. Strings must be entered within double quotes.

Articles with a DOI

Given that the material you're citing has a DOI, making a citation is extremely simple. All you need is the DOI and the in-text citation. Everything else just works.

```
# 1
- doi: https://doi.org/10.1007/s00484-015-1117-4
```

A couple of things to keep in mind:

1. The citations extractor is based on the CrossRef API, so if CrossRef doesn't have the right data parsed, it won't work. Please deploy it to your own Github account and verify that the citations look right before sending a pull request.
2. You can use CrossRef Metadata Search and zbib for finding DOI's and cross-checking.

Articles without a DOI

Some old articles might not have a DOI. In this case, you will have to use zbib or the like to get citation entries in **APA format** and manually enter them.

```
# 2
- article:
    authors: "Allen, M. J., & Sheridan, S. C."
    year_published: 2015
    title: "Mortality risks during extreme temperature events (ETEs) using a␣
↪distributed lag non-linear model."
```

(continues on next page)

(continued from previous page)

```
    journal: "International Journal of Biometeorology"
    numbers: "62(1), 57-67."
```

A few things to keep in mind here:

1. All the fields shown above are mandatory.
2. Citations must follow the APA format.
3. Indentation is important.
4. Strings must be entered within double quotes.

Citing Websites

Unfortunately, it is not possible to pull citation data for websites at the moment. It needs to be entered manually as described here. You can use zbib or the like to get citation entries in **APA format**.

```
# 3
- webpage:
    published: "March 15, 2019"
    authors: "Pranav Ballaney"
    title: "Agriculture: Crop production: Sugarcane.
    accessed: "June 22, 2020"
    site_name: "TNAU Agritech Portal"
    url: 'https://google.com'
```

A few things to keep in mind here:

1. *published* and *authors* can be left out but all others are mandatory.
2. Citations must follow the APA format.
3. Indentation is important.
4. Strings must be entered within double quotes.

Citing Books

If your book doesn't have a DOI, it is not possible to pull citation data automatically. It needs to be entered manually as described here. You can use zbib or the like to get citation entries in **APA format**.

```
# 4
- book:
    authors: "Ingalls, B. P."
    year_published: 2013
    title: "Mathematical modeling in systems biology: An introduction."
    publisher: "MIT Press"
    Google_Books_URL: "https://books.google.co.in/books?id=OYr6AQAAQBAJ"
```

A few things to keep in mind here:

1. `Google_Books_URL` can be left out but all others are mandatory.
2. Citations must follow the APA format.
3. Indentation is important.
4. Strings must be entered within double quotes.

In-text Citations

In-text citations work just like links, but are formatted differently automatically. Take a look at Sample.pug and the published page to get an idea of how this works.

```
:markdown-it(html)

    In text citation for a research article with a DOI. [Rosano et al., 2019](
↪#citation1)

    In text citation for another research article with a DOI. [Allen & Sheridan,␣
↪2015](#citation2)

    In text citation for a book with no DOI. [Ingalls, 2013](#citation3)

    In text citation for a website with institutional author. [TNAU Agritech Portal,␣
↪n.d.](#citation4)

    In text citation for a website with an author. [Pranav, n.d.](#citation5)
```

Notice the hashtag in-text citations, along with the number. The number provided here has to correspond to the full citation in the yml file, otherwise links will break.

All citations are written in the APA format. You can read more about it here.

3.4.3 Theme

Use the Built-in Theme or Build your Own

The starter pack comes with a design template that you can directly use. By editing just the text on each page, you fulfill all the requirements of an iGEM wiki.

The starter pack also makes it really easy to develop your own theme. It completely separates the content of your wiki from its design, so a part of your team can work on the design, while everyone else can just write plain text files with the actual content.

3.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

3.5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

3.5.2 Documentation improvements

igem-wikisync could always use more documentation, whether as part of the official igem-wikisync docs, in docstrings, or even on the web in blog posts, articles, and such.

3.5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/igembitsgoa/igem-wikisync/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

3.5.4 Development

To set up *igem-wikisync* for local development:

1. Fork igem-wikisync (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:igembitsgoa/igem-wikisync.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

3.6 Authors

- Pranav Ballaney - https://github.com/ballaneypranav

3.7 Contributors

3.7.1 Documentation and Bug Reports

- Xinhe Xing - https://github.com/tamithia

3.8 Changelog

3.8.1 0.0.0 (2020-08-18)

- First release.