



GoGEM: How to Wiki the Darmstadt Way

iGEM Team TU Darmstadt 2021

Table of Contents

Introduction.....	3
What is WordPress?	3
What is GoGEM?	3
Why you should put in the “extra” effort.....	3
Beginning development with WordPress.....	4
Setting up a WordPress instance.....	4
Modifying your WordPress instance	6
Developing a Theme.....	7
Starting with a custom theme.....	7
Setting up your Workplace.....	10
Editing the theme.....	10
Use the Browser Luke.....	13
Display the source code.....	13
...the smart way	13
How to GoGEM.....	17
Installation and usage of GoGEM.....	17
The GoGEM config	19
Things to watch out for	20
A list of plugins we recommend.....	21
Academic Blogger’s Toolkit	21
Add from Server	21
EditorsKit.....	21
FileBird Lite	21
FlatPreloader	21
Simple iFrame.....	21
SVG Support.....	21
Ultimate Blocks.....	21
Epilogue.....	22

Introduction

What is WordPress?

WordPress is the most used content management system (CMS) in the world, as of 2021. A CMS allows an administrator to easily curate and manage the contents of a website. WordPress is open-source software and was originally designed for powering web blogs, but through a plugin store and a vivid community basically every kind of website can nowadays be powered with it.

Exactly this community, the broad usage, and usage spectrum inspired us to develop a system that **would allow future iGEM teams to build their wiki on-top of WordPress**, instead of the obsolete iGEM internal editor.

Through the relatively new "Gutenberg-Editor" it is now even possible to create a whole website only by drag-and-drop of predefined blocks. This is an extraordinary opportunity for iGEM. Gone are the days where the whole team had to learn HTML, CSS and sometimes even JavaScript to build the wiki. **With WordPress a single person knowing HTML, CSS and JavaScript in the team is enough to create a modern wiki.**

What is GoGEM?

Although the before mentioned features were available for at least a couple of years, until now there was no way to utilize these capabilities for iGEM; Introducing **GoGEM**, a command line interface (CLI) tool that **automatically transfers a website designed in WordPress to iGEM**, while also taking care of uploading all referenced files directly to the iGEM servers and rewriting all links, ensuring full functionality directly out of the box.

With GoGEM it is finally possible to build an iGEM wiki like every other website.

GoGEM does this by converting the dynamic WordPress page, which relies on a database and a PHP backend, to a static page which then gets uploaded to the iGEM Servers.

Why you should put in the "extra" effort

It may seem like unnecessary effort to set-up a whole WordPress instance and development environment, and it may not be the way to go for every team. The decision if you want to use this system is yours, but the advantages are obvious:

- You will actually have an environment that is meant for developing a website.
- Your team will only need a small group of people that want to learn coding. In principle a single person could develop the theme and set-up the wiki, although out of experience we would not recommend doing so.
- Through not directly editing HTML and CSS while filling your wiki with content you will prevent errors that would potentially be wiki breaking.

Beginning development with WordPress

In principle the usage of WordPress is free of charge. As stated, WordPress is open-source software, and therefore freely available for everyone. In the following I will describe how to setup a development environment with a program called "Local". Local is a free of charge program that runs a locally hosted webserver on your PC. This allows you to run a local instance of WordPress and enables you to develop without the need to be online.

In theory it would be possible to develop the whole wiki from start to finish with Local, but this could be problematic because no one else will be able to work on the wiki. Therefore, we strongly recommend your team to rent a cheap webhosting service which allows you to set up a WordPress instance that is reachable from the internet.

The setup remains practically the same, but can slightly differ between providers. This is why we stick to explaining the setup process with Local.

Setting up a WordPress instance

After downloading and installing Local you will be greeted by a screen looking similar to Figure 1. Through a click on the green button in the lower left corner you will be presented with a wizard leading you through the creation of a new website. Simply enter a name for your WordPress instance, choose the preferred settings on the next page and enter a username and password for the admin account of your website.

Our quick two cents to this: Because WordPress is such a popular backend for websites there are also many attackers specialized on capturing WordPress Pages. Therefore, **choose a username and password that is not used anywhere else** and advise your teammates to do the same. This is why we also advise on backing up your page at least once a week.

You will not have to worry about security issues later on, because we will convert the dynamic WordPress page to a static one which will be hosted on the iGEM servers.

After completing the wizard Local will do the rest, including the registering of a hostname on your local machine. On windows this requires administrator privileges.

When the setup completes you will see a window similar to the one in Figure 2.

To see what is displayed on your fresh WordPress instance click on "Open Site".

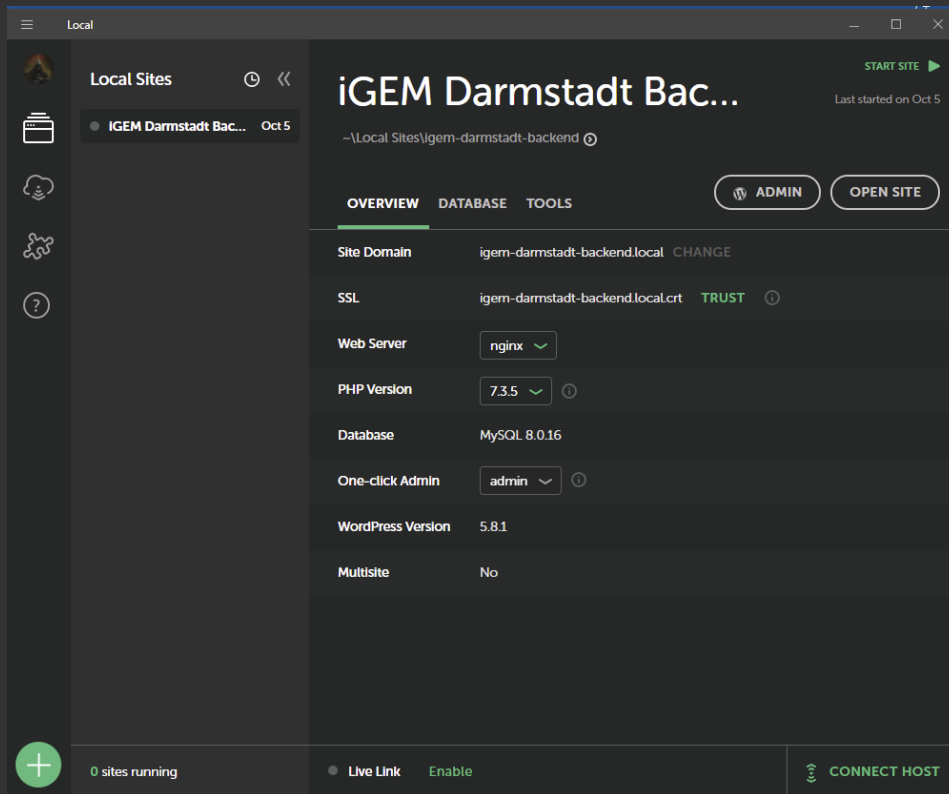


Figure 1 Home screen of Local. In the column on the left you have an overview of all your local sites, on the right is an overview over the chosen settings of the selected site.

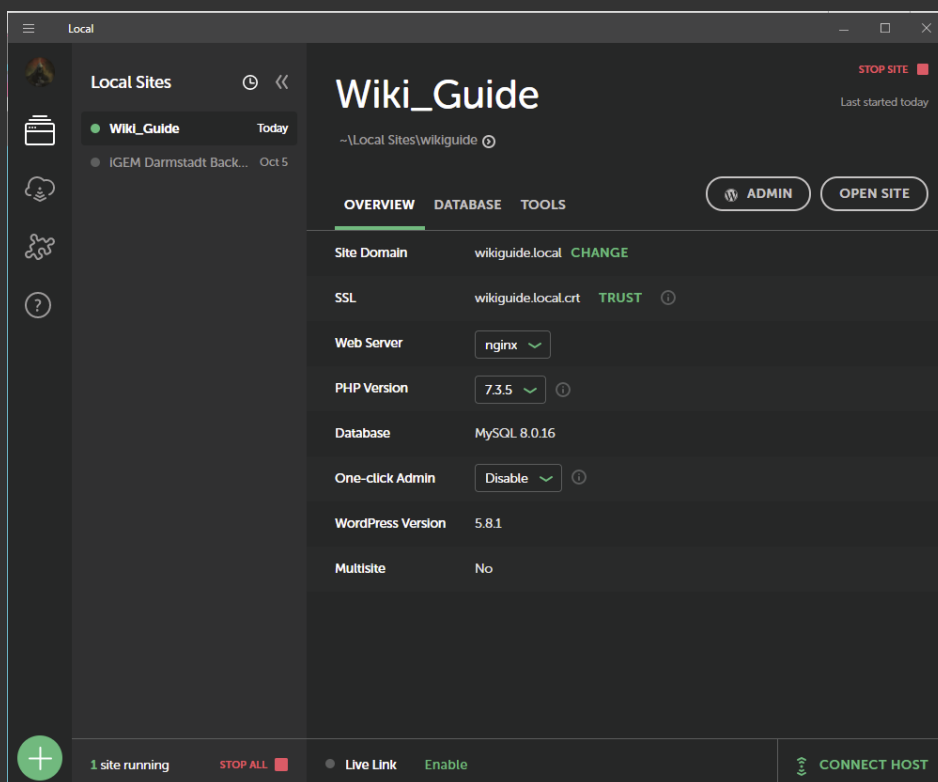


Figure 2 Congratulations! Your first WordPress page has just gone live! Well not really if you follow along on Local, but it is reachable from your PC.

Modifying your WordPress instance

Back in Local, click on the second button labeled “Admin”. You will see the WordPress login mask; Enter the credentials you chose earlier.

After logging in you will see the WordPress Admin-Dashboard. From here you will do most of the behind-the-scenes work!

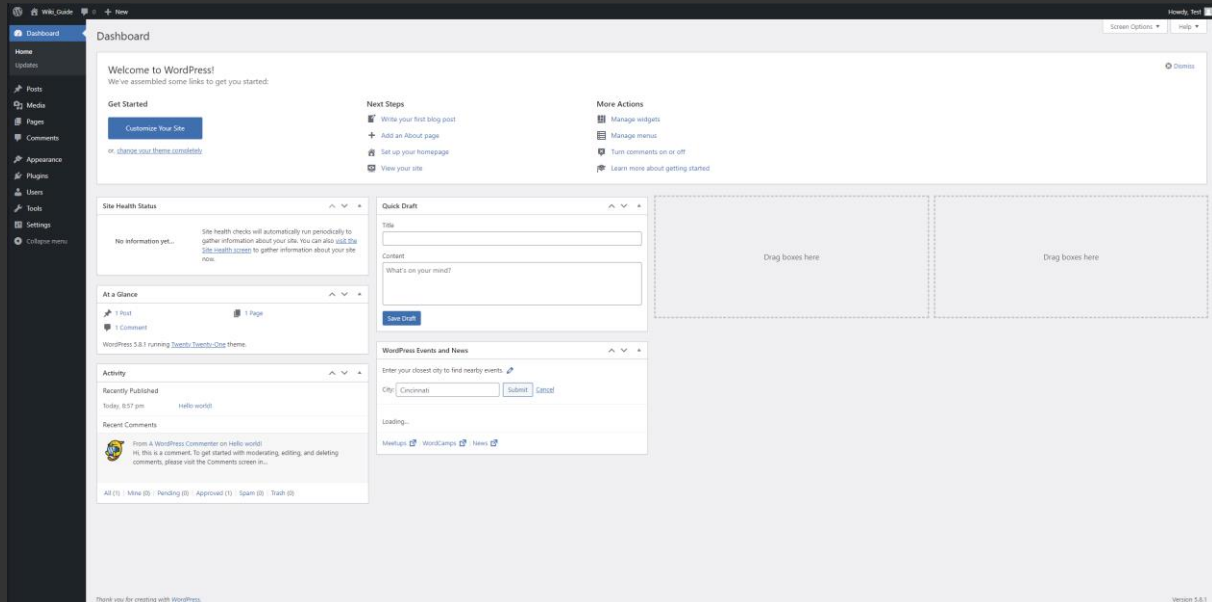


Figure 3 The WordPress Dashboard; The heart of your wiki.

Because ultimately a static wiki is generated, you will have to limit yourself to using only WordPress functionalities that do not require server-side data manipulation, or storage later on when the wiki is uploaded to iGEM. This means that things like comments, accounts etc. will not be possible. This is also why we will focus on only a few of the options presented on the left side.

Most important will be the “Pages” button. Here you will find all of your pages which will be on the iGEM wiki later on. Because we are building a static wiki, the “Posts” and “Comments” pages can effectively be ignored. The second most important page is “Plugins”. In need for a reference manager or a content toggle? Why reinvent the wheel? Take a look at the mass of available WordPress plugins. The chances that someone before you already has been in need of this exact thing and wrote a plugin for it are very high.

This is the most important lesson for the wiki backend: **Do not try to be perfectionistic, you won't be able to achieve a perfect wiki, so, do not try! Obey the 20:80 rule; With 20% of the time, you will be able to achieve 80% of the result.**

If you find something that kind of works for your usage case, take it and use it. Do not try to build something better from scratch. If there is time later on you can build upon the third-party solution and customize it to your liking.

Developing a Theme

By now you will most likely have clicked around the Dashboard a bit and familiarized yourself with it (if not, you should).

Under the “Appearance” page you will probably have seen that the WordPress default theme is selected, and you will most likely have looked into the “Theme Store”. If you found a theme that suits you and your team: Perfect, read the license, obey it and use it! That’s it, we are done here, read the GoGEM section and enjoy!

If, however, your team has the desire to build a custom theme you should follow along.

Starting with a custom theme

A good point for WordPress theme development is the underscores theme. This is a theme that is especially designed to be as verbose as possible. It accomplishes this through a well commented code, which will enable you to understand how it works, and how to modify it to your liking.

While generating the Theme, take care that you check the “_sassify!” checkbox (Figure 4).



Figure 4 Generate your underscores-based theme. Make sure to check the “_sassify!” checkbox!

This option will ensure that the backbones of your theme are created with SASS as stylesheet language. SASS stands for “Syntactically Awesome Style Sheet” and is

a language that is built on-top of CSS. This means it will get translated back to CSS on compilation of your theme, but enables you to write it in a nicer and more organized way. We will come back to this later. After generating your theme, you will get a zip file with the necessary files. To open the folder where they belong, go to Local and click onto the little arrow bellow the site name (Figure 5)

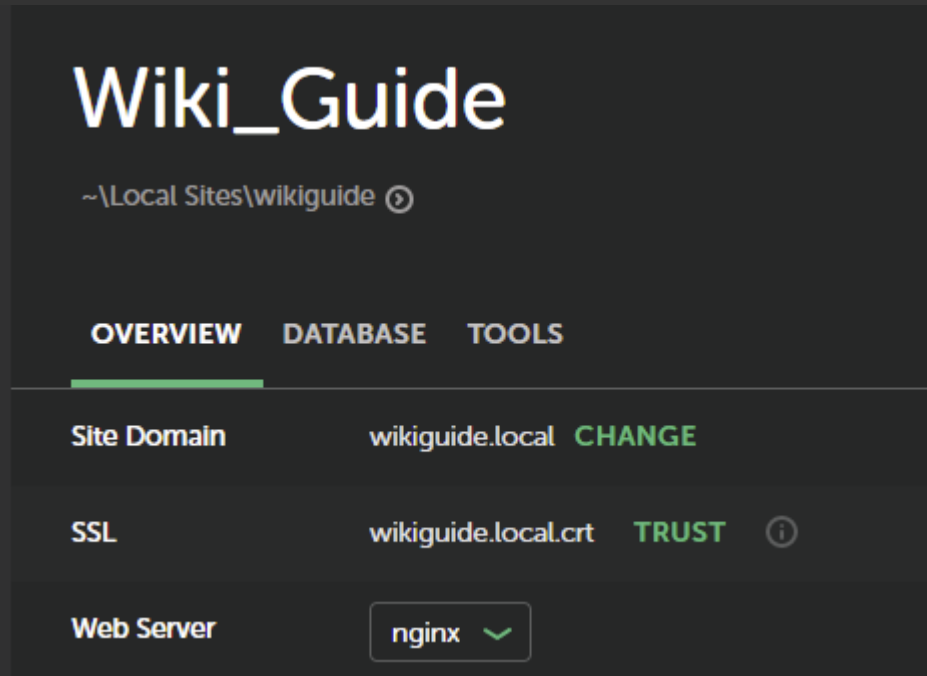


Figure 5 Where is the site located on your PC? Just follow the path below the site name, it is as easy as clicking the little arrow.

You will be presented with three folders, although only the app folder is of concern to us. Open it, do the same with the “public” folder that will be presented to you and et voilà: The guts of your site (Figure 6).

To use your theme, you will have to extract the zip file you got from underscores into the “wp-content/themes” directory. When done, refresh your WordPress-Dashboard and you will see that the theme is now selectable in the “Appearance” page.

Select it and take a look at your homepage through clicking on the name of your site in the upper left corner of the dashboard. You will be presented with the “underscores” default theme (Figure 7).

mbuth > Local Sites > wikiguide > app > public

Name	Date modified	Type	Size
wp-admin	09.09.2021 04:20	File folder	
wp-content	14.10.2021 20:03	File folder	
wp-includes	09.09.2021 04:20	File folder	
.htaccess	13.10.2021 22:57	HTACCESS File	1 KB
index.php	06.02.2020 07:33	PHP Source File	1 KB
license.txt	01.01.2021 01:19	Text Source File	20 KB
readme.html	06.07.2021 14:23	Chrome HTML Do...	8 KB
wp-activate.php	21.01.2021 02:37	PHP Source File	7 KB
wp-blog-header.php	06.02.2020 07:33	PHP Source File	1 KB
wp-comments-post.php	17.02.2021 14:08	PHP Source File	3 KB
wp-config.php	13.10.2021 22:57	PHP Source File	3 KB
wp-config-sample.php	21.05.2021 12:40	PHP Source File	3 KB
wp-cron.php	30.07.2020 21:14	PHP Source File	4 KB
wp-links-opml.php	06.02.2020 07:33	PHP Source File	3 KB
wp-load.php	15.05.2021 19:38	PHP Source File	4 KB
wp-login.php	06.04.2021 20:39	PHP Source File	45 KB
wp-mail.php	14.04.2020 13:32	PHP Source File	9 KB
wp-settings.php	02.06.2021 01:09	PHP Source File	22 KB
wp-signup.php	07.05.2021 22:16	PHP Source File	31 KB
wp-trackback.php	08.10.2020 23:15	PHP Source File	5 KB
xmlrpc.php	08.06.2020 21:55	PHP Source File	4 KB

Figure 6 The files that define your WordPress page.

Wiki Guide

Just another WordPress site

[Sample Page](#)

Hello world!

Posted on [October 13, 2021](#) by [Test](#)

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

Posted in [Uncategorized1](#) [Comment](#)[Edit](#)

Search

Recent Posts

Figure 7 The underscores homepage will look roughly like this. Awful, isn't it?

Setting up your Workplace

There is one step left before you can actually edit your wiki. And, lucky you, it's the programmer's favorite thing in life: Setting up your development environment.

We personally recommend Visual Studio Code as a foundation, simply because it is really lightweight and highly customizable.

Take your time to set up this editor to your liking (i.e., choose a theme you enjoy looking at, choose a font you like, etc.) you will spend a lot of time over the next weeks or months looking at this editor, so make sure you do not have a baseline grudge against it.

If you are done setting up your editor there is only one thing left. Remember how I said that SASS will be translated back to CSS when you compile your theme? Well for this we will have to install a program that actually knows how to do this. Head over to [SASS](#) and install it. Check out the "Learn SASS" section as well, here you will find information on how to modify the SCSS files, as well as how to compile them into a working CSS file that can be interpreted by your browser.

Now we can start looking at the theme itself.

Editing the theme

If you installed Visual Studio Code correctly you will have an entry in the right-click context menu (at least on windows) that reads "Open with Code". Enter the theme folder and open the folder in Code. You will be presented with a view similar to this (Figure 8). On the left you will have an overview of the folder you just opened, the mostly blank space on the right is where you will spend the better part of your next weeks. The console below it is an interface to the standard command line of your system. We will use this for compiling the theme and for interaction with GoGEM.

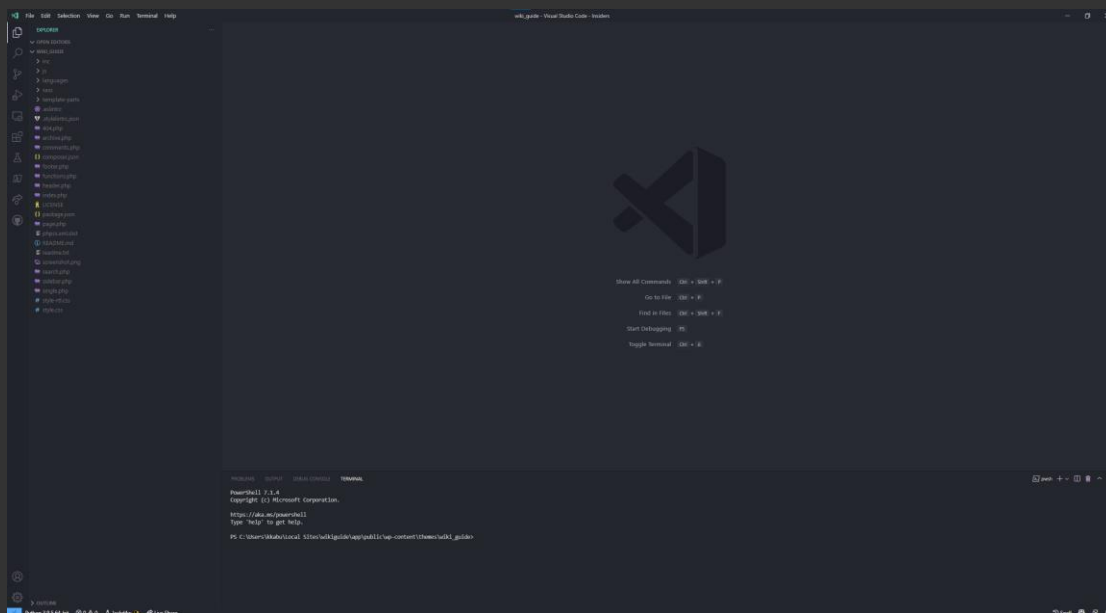


Figure 8 Opening your theme for the first time in VS Code.

The theme has two “parts” that will take the most of your time, and a few more annoying ones that you won't have to interact with that often.

We will start with the PHP files that are situated at the root of the theme.

PHP is a programming language that has been around for over 25 years and is mostly used to programmatically create websites. It allows a developer (you!) to create pages with different functionality and appearance, depending on the conditions that are present when the website is accessed. This is especially useful for us because we will have a number of pages on our wiki and it won't be necessary to rewrite the whole structure for every page on creation, and more importantly, on every change; 20:80 – remember?

Oh, and a quick heads up if you google PHP and the first thing you find is “PHP is dead”; Don't be discouraged to learn at least a bit of PHP: The legend that PHP is a dead language has been around for at least five years and it is still found on most websites, simply because it is a relatively easy language that is well established.


Okay, back to the structure;

There are five PHP files that will be used for creation of the pages:

- Header.php: Defines the actual header of your page; including the display of your custom logo and the main navigation bar.
- Page.php: This file defines the main content of most of your pages.
- Index.php: This file defines the content of your front page.
- Sidebar.php: Defines how your sidebar will look.
- Footer.php: Defines how the footer looks.

Pretty intuitive, isn't it? – There is another PHP file that is important: the functions.php. This file tells WordPress what your theme is, which files are part of it and where to find them. I have good news for you: This is probably the file that has the worst commenting and is therefore the most intriguing. But don't worry, all it does is calling functions from WordPress, so it's worth to take a look at the [WordPress Developer Documentation](#). There you will find most of the function calls which are done in the default PHP files.

The second important component is the SASS folder. It contains all the files that will be compiled into the style.css file; This is the file that describes how your wiki will look. Therefore, it is important to understand how you can manipulate the SCSS files so that your final page will look the way you want it to. The good thing is: **This is easy!** The style.scss is very well commented again. Basically, all it does is importing the corresponding files out of the subfolders. This enables the folder structure to be logically organized and enables you to find the style definitions at the place where you would assume them to be (i.e., the style for the body will most likely be under base and elements).



Now you know the basics about changing your theme regarding structure and looks.

The last and – in our personal opinion – most dreaded section is modifying the functionality of your page. For this you will be using JavaScript.

JavaScript is a scripting language that allows you to create dynamic functionality which gets evaluated in the user's browser. JS is the way you will make things like a scroll spy or a content toggle work. Because JavaScript gets executed on the client-side, we can use it in our static iGEM wiki.

The only thing you will have to do – apart from actually writing the JS code – is registering your script in the functions.php file. There are already two scripts registered, so just create an analogue entry for your script and you are good to go.

Use the Browser Luke...

Now that you know where to edit your theme it is time to take a look at what to edit. For this we will use the most powerful tool at your disposal: Your browser.

A quick heads up: you will probably want to look at your site through different browsers: Use a Chromium based one like Chrome or Opera, Firefox and your systems native browser (Edge or Safari), all of these are based on an architecture in the background, which can sometimes result in bugs that are only visible in browsers utilizing the same architecture.

Display the source code...

All your browser does is to receive the code that describes the structure and looks of a website and to translate it into something that the user can look at (if this is a pleasing experience is, however, absolutely up to the person writing the code). But all browsers are also capable of displaying the code they receive directly. Try right-clicking onto your WordPress page and then clicking on "View Source". You will be presented with a rather intimidating looking screen of code. Take a look, you will probably not understand everything, but if you can read English, you should get a rough idea of how the page is structured.

...the smart way

Although debugging in the source code view is possible, it is very much not a smart way of doing it. Go back to your WordPress homepage and try right-clicking again, this time select the entry "Inspect element". This will open up your second place of residence for the remaining time of iGEM: The developer console (Figure 9 & 10).

The developer console is basically the source code view on steroids. In the left part of the console, you will see the same source code you just looked at in the source code view, although this time you can collapse elements you are not interested in – which drastically reduces clutter – and there is syntax highlighting, dramatically improving the speed at which you can read through code and will be able to find things, both are tools which you will not want to miss anymore.

On the right there is the "Styles" tab selected. This tab shows which styling rules actually affect the inspected element, and more importantly from which file they come (this is possible through something called a source-map, try googling it if you are interested in how this works and what it does). Here you can also see which rules overwrite other rules, something that is especially important once transferring your wiki to iGEM. This is because iGEM actually uses a relatively invasive stylesheet for its page styling which will inevitably sweeten up your day more than once while trying to style your wiki accordingly.

The colorful thing at the bottom shows you which spaces around the element itself are claimed by the element. These are divided into padding, border and margin. If you do not know what this is, refer to looking them up. In general, the advice

“Google it” holds true, that’s basically your bread and butter as a developer. W3Schools is a good starting point for programming questions.

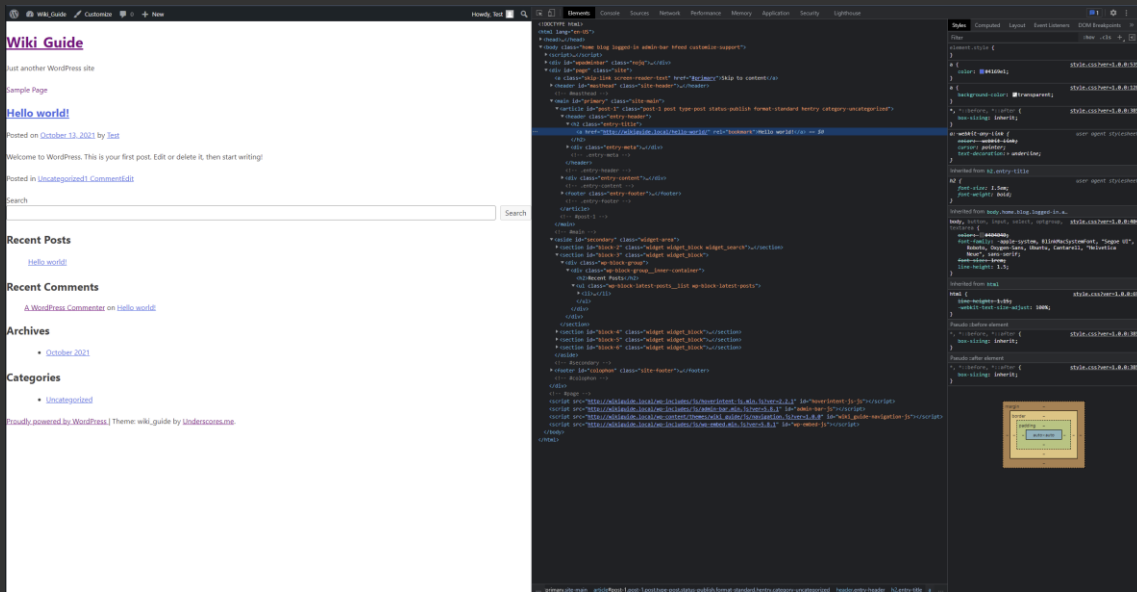


Figure 9 Inspecting the “Hello world!” Headline opens the developer console focused on the corresponding code fragment.

If you want even more insight into what exactly your browser does to render an element, open up the “Computed” tab in the right window (Figure 11). Here the browser shows which values it actually uses and why it does so. Although this information is mostly trustworthy it sometimes turns out to be flat wrong (at least in Opera), we found that this mostly occurs in combination with the use of the CSS “!important” keyword.

Read up on more features and quirks of your particular developing so you can use it effectively.

And while you are at it try also looking up on how to disable the cache in your browser. This will be helpful when developing and will definitely prevent many headaches of the “The page will just not change” type, which are tricking you into making crazy changes in the stylesheet, hoping to find the cause for the problem – which you won’t. After some time, your browser then decides to take a look at the new file you massacred for the last hour and before you know it your page looks like someone puked onto the pink Teletubby. How do we know? Well don’t ask.

With that you now actually have the knowledge that is needed for beginning the theme customization process.

And remember: RTFM and do not reinvent the wheel.

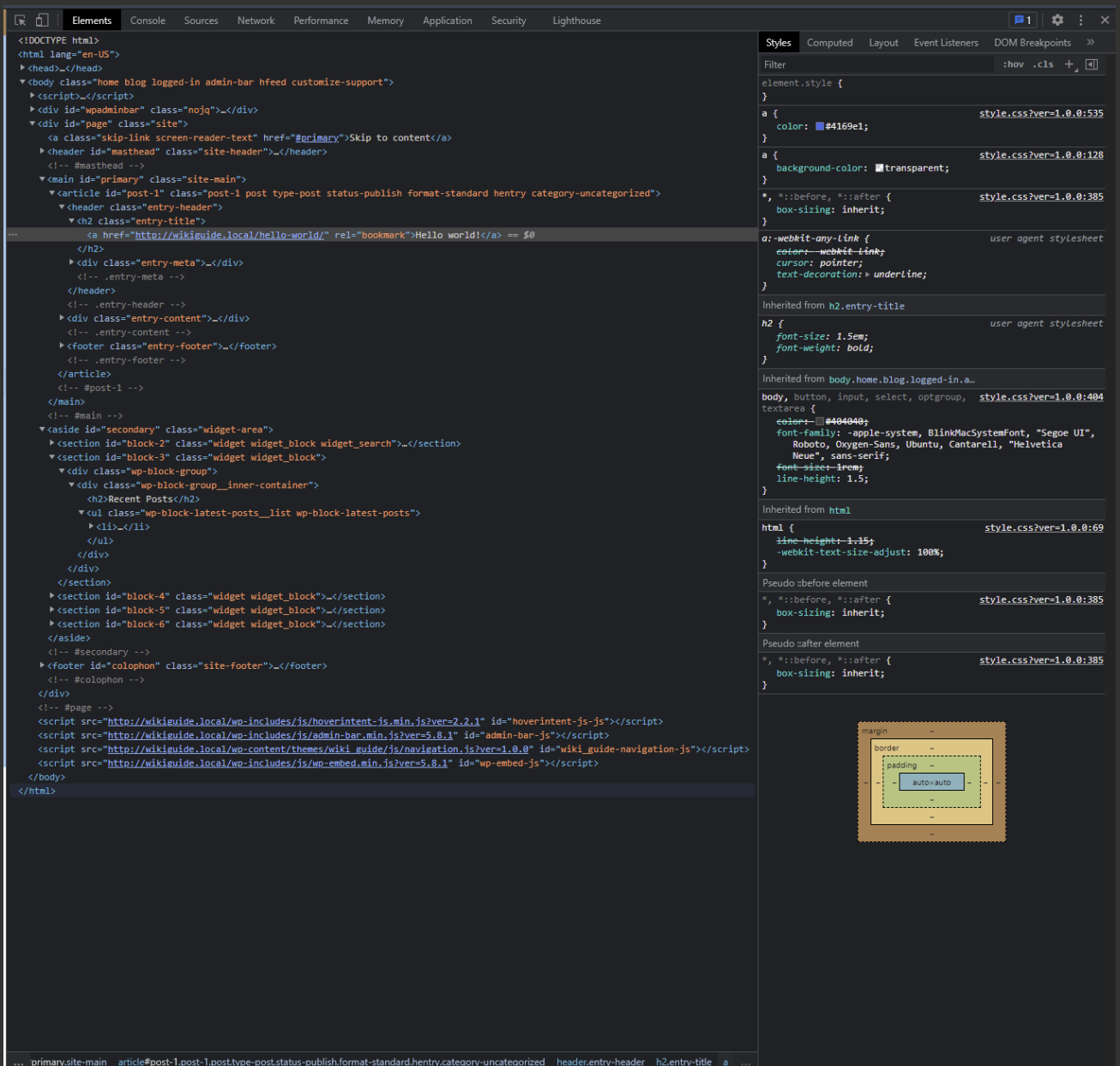


Figure 10 The dev console in detail. On the left the "Elements", "Console" and "Network" tab are the most important, on the right you will mostly use the "Styles" and "Computed" tabs.



Figure 11 The "Computed" tab, showing which values your browser actually uses (most of the time).

How to GoGEM

After a whopping 15 pages of introduction, we are now finally at the part that you can only find in this document. But first: Congratulations, you survived (until now)!

Now that you have a super-duper awesome WordPress theme you are finally thinking about how you will transfer it over to iGEM. Well, you are lucky, because I thought the same thing and had no fricking clue.

That was the birth of GoGEM.

As you will surely know by now that GoGEM does all the stuff you really do not want to. It replaces all links between your pages, uploads all the media files you have used and even replaces the links referring to them with the new ones from iGEM, ensuring that your page will completely be hosted on the iGEM servers – this painful process is necessary in order to comply with the rule that absolutely all content has to be hosted on the iGEM servers.

Okay, enough self-praise. How do you utilize this tool? Well, it's not as complicated as you think.

Installation and usage of GoGEM

Head over to the GitHub repository of [GoGEM](#), there you will not only find pre-compiled and ready to run executables, but also the source code. For one this enables you to check that we are not trying to trick you into mining Bitcoins for us, but more importantly you could also help us maintaining, and further developing, this tool to improve the process of wiki development further.

Anyways, download a pre-compiled version fitting your operating system and save it in a location you can access with your command line of choice. Extract the zip file, which will reveal the executable and a config file. Take a look at the config file first, here things that most likely will not change that often are situated.

Open the created folder with your command line and try running GoGEM, this will return a short help message giving a quick overview of the available commands (Figure 12).

GoGEM follows a APPNAME COMMAND ARG -FLAG scheme, resulting in commands being reasonably intuitive.

To upload your WordPress page, you will need the *GoGEM upload* command, which reveals a command specific help window (Figure 13).

According to this help message the appropriate upload command would then be:

```
GoGEM upload -w "[URL_TO_YOUR_WORDPRESS_HOMEPAGE]" -y [YEAR] -t "[TEAMNAME]" -u "[USERNAME]"
```

And from here on GoGEM basically behaves like you would expect it to.

Please remember that spaces in the team's name will have to be replaced by underscores, and that the URL will have to be entered with the used protocol (i.e., http:// or https://). Following the presented scheme, you will be able to utilize GoGEM, even if the exact syntax changes in the future.

```
PowerShell 7 (x64)
PS C:\Users\kkabu\Desktop> .\GoGEM.exe
Upload your Wiki to iGEM

Usage:
  GoGEM [command]

Available Commands:
  checkCriteria Check Medal Criteria URLs
  completion    generate the autocompletion script for the specified shell
  fetchWP       Clone a WordPress Site to your PC, maintaining all static functionality
  help          Help about any command
  purge         CAUTION!! DESTRUCTIVE ACTION! Purge your Wiki Pages from the Server
  upload        Upload your WordPress Page to iGEM

Flags:
  --config string  config file (default is ./GoGEM.json)
  -h, --help       help for GoGEM
  -t, --toggle     Help message for toggle

Use "GoGEM [command] --help" for more information about a command.
PS C:\Users\kkabu\Desktop>
```

Figure 12 A short overview displaying the available commands.

```
PowerShell 7 (x64)
PS C:\Users\kkabu\Desktop> .\GoGEM.exe upload
Using config file: C:\Users\kkabu\Desktop\GoGEM.json
Error: required flag(s) "teamname", "username", "wpurl", "year" not set
Usage:
  GoGEM upload [flags]

Flags:
  -c, --clean           Clean (default true)
  -d, --delete         Delete the temporary files after upload (default true)
  -f, --force          Force
  -h, --help           help for upload
  -i, --insecure       Ignores HTTPS Certificate warnings
  -o, --offset string  Offset from your Teams Namespace root
  -p, --password string Password
  -r, --redirect       Creates redirects from upper to lowercase
  -t, --teamname string Teamname(required)
  -u, --username string Username(required)
  -w, --wpurl string   WordPress URL(required)
  -y, --year int       Year(required) (default 2021)

Global Flags:
  --config string  config file (default is ./GoGEM.json)

Error: required flag(s) "teamname", "username", "wpurl", "year" not set
PS C:\Users\kkabu\Desktop>
```

Figure 13 The help message for the "upload" command.

Well, that section was refreshingly short, wasn't it? – Yep, that is why I will use the opportunity to explain you a bit more about the config file. Still here? That's good, we are almost done!

The GoGEM config

The config file is written in JSON, which brings a huge disadvantage with it: No comments. That is why we look at it here (Figure 14).

```
1  {
2  "URLs": [
3  {
4  "Medals": "#",
5  "Bronze #2 (Attributions)": "Attributions",
6  "Bronze #3 (Project Description)": "Description",
7  "Bronze #4 (Contribution)": "Contribution",
8  "Silver #1 (Engineering Success)": "Engineering",
9  "Silver #2 (Collaboration)": "Collaborations",
10 "Silver #3 (Human Practices)": "Human_Practices",
11 "Silver #4 (Proposed Implementation)": "Implementation",
12 "Gold #1 (Integrated Human Practices)": "Human_Practices",
13 "Gold #3 (Project Modeling)": "Model",
14 "Gold #4 (Proof of Concept)": "Proof_Of_Concept",
15 "Gold #5 (Partnership)": "Partnership",
16 "Gold #6 (Education & Communication)": "Communication",
17 "Awards": "#",
18 "Best Education": "Education",
19 "Best Hardware": "Hardware",
20 "Inclusivity Award": "Inclusivity",
21 "Best HP": "Human_Practices",
22 "Best Measurement": "Measurement",
23 "Best Model": "Model",
24 "Best Plant SynBio": "Plant",
25 "Best Software Tool": "Software",
26 "Best Supporting Entrepreneurship": "Entrepreneurship",
27 "Best Sustainable Development Impact": "Sustainable",
28 "Safety and Security Award": "Safety"
29 }
30 ],
31 "CustomRedirects": [
32 {
33 "Education": "communication",
34 "Sustainable": "excellence#GreenerLabs"
35 }
36 ],
37 "Order": [...
63 ],
64 "Fonts": [
65 {
66 "Philosopher": "url(https://2021.igem.org/wiki/images/e/ef/T--TU_Darmstadt--Philosopher.woff)",
67 "Montserrat": "url(https://2021.igem.org/wiki/images/4/42/T--TU_Darmstadt--Montserrat.woff)",
68 "Raleway": "url(https://2021.igem.org/wiki/images/5/53/T--TU_Darmstadt--Raleway.woff)"
69 }
70 ],
71 "LoginURL": "https://igem.org/Login2",
72 "LogoutURL": "https://igem.org/Logout",
73 "PrefixURL": "https://%d.igem.org/wiki/index.php?title=Special:PrefixIndex",
74 "MathJaxURL": "https://2021.igem.org/common/MathJax-2.5-latest/MathJax.js?config=TeX-AMS-MML_HTMLorMML"
75 }
76 }
```

Figure 14 A sneak peek into the config file, which allows you to use the tool even if iGEM changes URLs etc.

The config file is separated into different sections.

- URLs: Defines which URL Slug is necessary for which Medal / Award.
- CustomRedirects: Defines redirects from the map key to the map value.
- Order: Just a list of the URL keys, this is only there so the checkCriteria output is indeed ordered in some way.
- Fonts: Reference the font name to the font location on the iGEM servers.
- LoginURL: Begin of the iGEM Login-Chain, should not change.
- LogoutURL: Begin of the iGEM Logout-Chain, should not change.
- PrefixURL: URL of the page that allows you to search for all pages with a defined prefix (Usually found in the "Special Pages" section), should not change.
- MathJaxURL: URL to the MathJax file on the iGEM servers, **this one will change.**

You may or may not have noticed the different capitalization in everything URL related: This is sadly not by mistake, but actually necessary. iGEM does differentiate between the URL .../Attributions and .../attributions, leading you to different pages. GoGEM does create the according redirects from the capitalized URL to the all-lowercase URL, but this is something you will have to watch out for when editing the config file.

Things to watch out for

Okay, now to the funny part of things that you should be aware of.

GoGEM "statifies" your site by following all "internal" links that are on the entry page that was specified by the "wpurl" parameter. Therefore, it is necessary that all pages and files you want to upload are somehow reachable from your main page.

Remember how we told you that you could build your page completely with GoGEM? Well, that was not completely true: As of late 2021 it is not possible for GoGEM to automatically transfer your used fonts to the iGEM servers. This means that you will have to upload the font files manually to the iGEM servers and enter the according URL into the config file.

If you use SVG files then be aware that iGEM sometimes rejects them because they presumably contain code. Through experimentation we found that using an SVG compressor seems to solve the problem in most cases.

A list of plugins we recommend

Academic Blogger's Toolkit

Although not maintained at the moment (late 2021), this plugin proved really useful to us. It adds, amongst other things, a reference manager which is able to import your typical BibTeX file. This is very helpful when writing longer wiki pages because the numbering is done automatically, which drastically lowers the workload when adding a new citation.

Add from Server

Is your hosting service limiting the file size you can upload via WordPress? If you have a way of accessing the file system of your server you can upload larger files directly and then register them in the WordPress database with this tool.

EditorsKit

As the name implies, this plugin adds a lot of functionality regarding text editing. It allows the addition of non-breaking spaces through the visual interface, the ability to justify your texts again and so on.

FileBird Lite

This plugin enables you to create a folder structure in the WordPress Media Library. Especially for larger projects, like an iGEM wiki, this is absolutely necessary if you want to keep an overview over all the media files.

FlatPreloader

As the name suggests this plugin allows you to create an animated loading screen which will be shown when your page is accessed.

Simple iFrame

Adds an iFrame block to the Gutenberg-Editor, which allows to reference videos from the iGEM video universe.

SVG Support

Allows you to upload SVG files to WordPress. This is not natively supported in order to prevent foreign code execution.

Ultimate Blocks

Adds a variety of blocks that are useful for the design of a wiki. Including content toggles and image sliders.

Epilogue

Of course, this guide is not comprehensive. It does not strive to be, and frankly it cannot be. The intend of this work is to give an idea on how the creation of an iGEM wiki could be approached. It tries to accomplish this through presenting the way we choose to develop our wiki. This way is by no means perfect or the only way. Nevertheless, if you choose to follow this approach, or found the guide helpful, **please refer to it on your wiki**. This is the only way this work can stay present in the iGEM process, helping teams in the years to come.

Although GoGEM is present in the 2021 iGEM software repository, the development repo will stay this [GitHub repository](#). Please only link to this repo, in order to prevent the spread of a possibly outdated version. If you want to contribute or fork the project please do so to and from this repo only.

We really hope this guide gave you an idea on how to approach the intimidating task of writing your Wiki, maybe even on the basis of a WordPress page.

Whichever way you choose, we wish you all the best,

Kai Kabuth

and the 2021 TU Darmstadt iGEM team