Maintenance Documentation for ----------

This is a website maintenance documentation for ----------. This file is available in PDF and MD format in the home directory of the website. Click here for the PDF version and click here for the MD/HTML version

Contents

1. Web Technologies

NOTE: This website has been built with the assumption that the end user who will be maintaining this website has negligible knowledge about web development, and thus this section might be skipped over. This section rather serves to warn the potential user of the possible failures that can occur due to unnecessary tampering.

This website is built on, and supports the following web technologies:

1. HTML (HyperText Markup Language version 5)
2. CSS (Cascading stylesheets version 3)
3. JS (Javascript version ECMAScript 6)
4. JQuery (version 3.3.1)
5. PHP (Hypertext preprocessor version 7.2.10)
6. MD (Markdown flavour basic 1.1.3+)
7. JSON (JavaScript Object Notation)
8. AJAX (Asynchronous Javascript and XML)

All of these web technologies are the most major ones used. Any changes to the website would require some knowledge of these depending on the specific portion of the website to be changed. This documentation will attempt to explain which specific technology is used where. This would attempt to give an overview to prevent unnecessary tampering with the source code where it is not required and does not presume to go into depth about how each of these technologies are used.

Other than these major technologies, smaller libraries are also used for achieving specific tasks. Those are also listed afterwards, along with the specific effect they are used to achieve.

1.1 HTML

HTML is the basic language of the web and is necessary to build any website. This website however only includes one html based file:

```
index.php
```

This file looks somewhat like this:-

```
.
.
.

    </div>
</article>
<!---------------------------------------------- Research Page ------------------
<article class="page research">
    <div class="top r-top">
        <h1><span class="raled">RESEARCH</span></h1>
    </div>
    <div class="r-mid source">
    </div>
    <div class="foot">
        <div>
            <span class="raled">Nano Materials Research Lab</span>
            <br>
            <span class="monster">IISER Kolkata</span>
        </div>
        <div class="iiserk monster"><a href="https://www.iiserkol.ac.in" target="_b
    </div>
</article>
<!---------------------------------------- Publications Page ------------------
<article class="page publications">
.
.
.
```

As evident, this file is just used to define an underlying structure for the whole website and does not contain any content whatsoever (except in a few cases which presumably can be ignored). This is is more of a skeleton of the website rather than the actutal website. Any changes made here will most probably not be reflected in the actual website. However **this file is the backbone of the entire website and any tampering with this file, renaming, or deleting this file might cause the entire website to crash.** It is suggested to practice extreme caution if it is imperative to edit this file, and not attempt any changes without proper knowledge of HTML, JavaScript, and PHP.

1.2 PHP

PHP is the language of the backend and does server side processing for the website. PHP is often used alongside html and thus causing the file name extension of HTML files to be changed from .html to .php, as is the case with this website.

1. The html file `index.php` contains a small snippet of PHP code in the beginning which helps generate the "Gallery" page dynamically. It looks like this:

   ```
   <?php
   require_once('scripts/gallery.php')
   ?>
   ```

   Add or change anything within the `<?php ?>` at your own risk only if you have an idea of what you are doing.

2. The scripts directory contains two PHP scripts:

   i. `gallery.php` - Used to dynamically populate the gallery page.
   ii. `mail.php` - Used to send query emails from the contact us page to the provided email id.

   Both of these scripts have a very specific purpose and it is suggested to edit only what is necessary. Erroneous editing of these files might cause the gallery to stop displaying images or the mails to not get routed properly to the designated reciever

   Only scenario for editing these scripts is when the email address where the queries are to be sent needs to be changed. The exact steps on how to achieve that will be explaned in later sections.

1.3 CSS

CSS is the designing language of the web. How the entire website looks is achieved by implementing CSS (v3+). There are two CSS files in the `css` directory:

1. `jquery-gallery.css` - This is the stylesheet for the slideshows present on the website (on the home page and the gallery page). **Changing anything here will not cause the website to crash. *However this can change how the slideshows look, and the user is advised to not tamper with anything here to avoid unnecessary, unexpected or unwanted changes in the design of the slideshows.*** (This is a publicly available library)
2. `main.css` - As evident from the name, this file contains all the style rules for this website,

and **has over a thousand lines of code**. Again **changing this file will only change how the website looks, but sometimes very small changes can be catastrophic and thus the user is advised not to tamper with this file much**. Only scenario for editing this file would be to change the color scheme for the website, which is explained in later sections.

1.4 JS, JQuery, JSON & AJAX

JS and Jquery+AJAX are used together for scripting this website and most of the heavy lifting is done by these, like:

- Causing the menu to work
- Making the content of the website publishable in markdown format
- Making scrolling animations
- Making the slideshows
- Making all the pages to be able to be loaded dynamically without refreshing the page, giving a seamless user experience
- Submitting the query form without ever having to refresh the page (Using AJAX)

All of the JS files used for these purposes are stored in the `js` directory. The files are:

1. `jquery-gallery.js` - Used for making slideshows from a list of images (This is a publicly available library)
2. `markdown.js` - Used for parsing Markdown into html code. This is what enables the user to write code in simple markdown, while it converts it into browser compatible, valid html. (This is a publicly available library)
3. `main.js` - Used to do rest of the work mentioned above. The scope of this script is too vast to explain, but the user would do well to understand that **any tampering with this file can be catastrophic.** An overview of all the function of this script:
    i. Choose which page to go to when a certain menu item is clicked
    ii. In mobile view, display the menu when the hamburger icon is clicked
    iii. Change the menu positioning to be a bit more non intrusive when the page is scrolled
    iv. Remove the preloader (the bubbling beaker) when the page has loaded **(Note that on load, actually only the home page and contact us page are loaded. The other pages are loaded only when they are accessed from the menu to reduce load times)**
    v. Populating the home page with content from the markdown files, making the slideshow on the homepage
    vi. Populating the contact us page with contact details provided in the markdown files
    vii. Submitting the query form through an asynchronous request to prevent refreshing the page, for a seamless user experience
    viii. Loading specific pages when their corresponding menu buttons are clicked byparsing the markdown files and storing them in strings, and only putting them into the pages when requested

JSON, on the other hand, is used for storing the content of this website, along with JS. Most of the content is however written in Markdown format and the JS or JSON file types are primarily for ease of access throush JS. ***These are perhaps the only files in the whole website structure which are supposed to be edited*** **(However caution still needs to be excercised when editing)** All of the JS and JSON files used for storing the content are present in the `txt` directory, and one file present in the `gallery` directory:

1. Inside `txt` :
    i. `group.js` - Contains the details of all the members of the group, mostly in JSON format. *More details on this are available later.*
    ii. `text.js` - Contains all of the content to be loaded into the website, mostly in MARKDOWN format. *More details on this are available later.*
2. Inside `gallery` :
    i. `a.json` - Contains an array of 1s and 0s, `1` indicating which section needs to be a slideshow inside the gallery, `0` indicating which section needs to be a grid. *More details on this are available later.*

1.5 Markdown

Markdown is a very simple markup language, primarily used for generating web pages with very little coding skills. **It has a much gentler learning curve than HTML (used for web documents), and is much less resource intensive than LATEX (used for print documents), which are two of the most popular markup languages in use today**, which is why Markdown was chosen as the language for this website. Resources for learning markdown are abundant on the web (An especially great tutorial series is available here), I however will provide a reproduction of the documentation of Github Flavoured Markdown available at markdown-here's repository later in this documentation. A pdf version of Github's official Markdown cheatsheet is also provided here. (This documentation is infact also written in Markdown and the markdown format is available here)

The only file that uses markdown in the entire website is the `text.js` file inside the `txt` directory. It is a JS file that stores the content of the website in markdown format.

1.6 Other libraries

Other libraries which are also used in the website:

- **Google fonts API** - Used to serve the three different fonts used in the website: `Montserrat` , `Raleway` & `Source Sans Pro`
- **Fontawesome** - Used for generating the various icons used in the website like the `facebook icon` , the `hamburger icon` and the `close button icon`

- **JQuery** - CDN used to enable jquery for the website

2. Directory Structure

The directory structure of the website is as follows:

- css
  - jquery-gallery.css
  - main.css
- gallery
  - 1 A Glimpse of Our Lab
    - (corresponding image files)
  - 2 Microscopic and Optical Images
    - (corresponding image files)
  - 3 Lab Fun
    - (corresponding image files)
  - 4 Lab Visitors
    - (corresponding image files)
  - a.json
- img
  - (all of the relevant images except for the gallery)
- js
  - gallery.json
  - jquery-gallery.js
  - main.js
  - markdown.js
- scripts
  - gallery.php
  - mail.php
- txt
  - group.js
  - text.js
- .gitignore
- index.html
- index.php
- markdown-cheatsheet.pdf
- README.md
- README.pdf

3. Markdown Tutorial

This is intended as a quick reference and showcase. For more complete info, see John Gruber's original spec and the Github-flavored Markdown info page.

NOTE: This tutorial is reproduced from https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet. Credits go to the respective authors.

The most important sections, pertaining to this specific website, in this tutorial are the first six sections. The rest of the sections are not required that much for this website

---

Table of Contents

Headers

```
# H1
## H2
### H3
#### H4
##### H5
###### H6

Alternatively, for H1 and H2, an underline-ish style:

Alt-H1
======

Alt-H2
------
```

H1

H2

H3

H4

H5

H6

Alternatively, for H1 and H2, an underline-ish style:

Alt-H1

Alt-H2

Emphasis

```
Emphasis, aka italics, with *asterisks* or _underscores_.
```

```
Strong emphasis, aka bold, with **asterisks** or __underscores__.

Combined emphasis with **asterisks and _underscores_**.

Strikethrough uses two tildes. ~~Scratch this.~~
```

Emphasis, aka italics, with *asterisks* or *underscores*.

Strong emphasis, aka bold, with **asterisks** or **underscores**.

Combined emphasis with **asterisks and *underscores***.

Strikethrough uses two tildes. ~~Scratch this~~.

Lists

(In this example, leading and trailing spaces are shown with with dots: ·)

```
1. First ordered list item
2. Another item
··* Unordered sub-list.
1. Actual numbers don't matter, just that it's a number
··1. Ordered sub-list
4. And another item.

···You can have properly indented paragraphs within list items. Notice the blank li

···To have a line break without a paragraph, you will need to use two trailing spac
···Note that this line is separate, but within the same paragraph.··
···(This is contrary to the typical GFM line break behaviour, where trailing spaces

* Unordered list can use asterisks
- Or minuses
+ Or pluses
```

1. First ordered list item
2. Another item
    - Unordered sub-list.
3. Actual numbers don't matter, just that it's a number

    i. Ordered sub-list

4. And another item.

   You can have properly indented paragraphs within list items. Notice the blank line above, and the leading spaces (at least one, but we'll use three here to also align the raw Markdown).

   To have a line break without a paragraph, you will need to use two trailing spaces.
   Note that this line is separate, but within the same paragraph.
   (This is contrary to the typical GFM line break behaviour, where trailing spaces are not required.)

- Unordered list can use asterisks
- Or minuses
- Or pluses

Links

There are two ways to create links.

```
[I'm an inline-style link](https://www.google.com)

[I'm an inline-style link with title](https://www.google.com "Google's Homepage")

[I'm a reference-style link][Arbitrary case-insensitive reference text]

[I'm a relative reference to a repository file](../blob/master/LICENSE)

[You can use numbers for reference-style link definitions][1]

Or leave it empty and use the [link text itself].

URLs and URLs in angle brackets will automatically get turned into links.
http://www.example.com or <http://www.example.com> and sometimes
example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

[arbitrary case-insensitive reference text]: https://www.mozilla.org
[1]: http://slashdot.org
[link text itself]: http://www.reddit.com
```

I'm an inline-style link

I'm an inline-style link with title

I'm a reference-style link

I'm a relative reference to a repository file

You can use numbers for reference-style link definitions

Or leave it empty and use the link text itself.

URLs and URLs in angle brackets will automatically get turned into links.
http://www.example.com or http://www.example.com and sometimes
example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

Images

```
Here's our logo (hover to see the title text):

Inline-style:
![alt text](https://github.com/adam-p/markdown-here/raw/master/src/common/images/ic

Reference-style:
![alt text][logo]

[logo]: https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48
```

Here's our logo (hover to see the title text):

Inline-style:


Reference-style:


Code and Syntax Highlighting

Code blocks are part of the Markdown spec, but syntax highlighting isn't. However, many renderers -- like Github's and *Markdown Here* -- support syntax highlighting. Which languages are supported and how those language names should be written will vary from renderer to renderer. *Markdown Here* supports highlighting for dozens of languages (and not-really-languages, like diffs and HTTP headers); to see the complete list, and how to write the language names, see the highlight.js demo page.

```
Inline `code` has `back-ticks around` it.
```

Inline `code` has `back-ticks around` it.

Blocks of code are either fenced by lines with three back-ticks ```, or are indented with four spaces. I recommend only using the fenced code blocks -- they're easier and only they support syntax highlighting.

```
```javascript
var s = "JavaScript syntax highlighting";
alert(s);
```

```python
s = "Python syntax highlighting"
print s
```

```
No language indicated, so no syntax highlighting.
But let's throw in a <b>tag</b>.
```
```

```
var s = "JavaScript syntax highlighting";
alert(s);
```

```
s = "Python syntax highlighting"
print s
```

```
No language indicated, so no syntax highlighting in Markdown Here (varies on Github
But let's throw in a <b>tag</b>.
```

Tables

Tables aren't part of the core Markdown spec, but they are part of GFM and *Markdown Here* supports them. They are an easy way of adding tables to your email -- a task that would otherwise require copy-pasting from another application.

```
Colons can be used to align columns.

| Tables        | Are           | Cool  |
| ------------- |:-------------:| -----:|
| col 3 is      | right-aligned | $1600 |
| col 2 is      | centered      |   $12 |
| zebra stripes | are neat      |    $1 |

There must be at least 3 dashes separating each header cell.
The outer pipes (|) are optional, and you don't need to make the
raw Markdown line up prettily. You can also use inline Markdown.

Markdown | Less | Pretty
--- | --- | ---
*Still* | `renders` | **nicely**
1 | 2 | 3
```

Colons can be used to align columns.

Tables	Are	Cool
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1

There must be at least 3 dashes separating each header cell. The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily. You can also use inline Markdown.

Markdown	Less	Pretty
Still	`renders`	**nicely**

1	2	3

Blockquotes

```
> Blockquotes are very handy in email to emulate reply text.
> This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh boy
```

> Blockquotes are very handy in email to emulate reply text.
> This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can *put* **Markdown** into a blockquote.

Inline HTML

You can also use raw HTML in your Markdown, and it'll mostly work pretty well.

```
<dl>
  <dt>Definition list</dt>
  <dd>Is something people use sometimes.</dd>

  <dt>Markdown in HTML</dt>
  <dd>Does *not* work **very** well. Use HTML <em>tags</em>.</dd>
</dl>
```

<dl>
  <dt>Definition list</dt>
  <dd>Is something people use sometimes.</dd>

  <dt>Markdown in HTML</dt>
  <dd>Does *not* work **very** well. Use HTML tags.</dd>
</dl>

Horizontal Rule

```
Three or more...

---

Hyphens

***

Asterisks

___

Underscores
```

Three or more...

Hyphens

Asterisks

Underscores

Line Breaks

My basic recommendation for learning how line breaks work is to experiment and discover -- hit <Enter> once (i.e., insert one newline), then hit it twice (i.e., insert two newlines), see what happens. You'll soon learn to get what you want. "Markdown Toggle" is your friend.

Here are some things to try out:

```
Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a *separate
This line is also a separate paragraph, but...
This line is only separated by a single newline, so it's a separate line in the *sa
```

Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a *separate paragraph*.

This line is also begins a separate paragraph, but...
This line is only separated by a single newline, so it's a separate line in the *same paragraph*.

YouTube Videos

They can't be added directly but you can add an image with a link to the video like this:

```
<a href="http://www.youtube.com/watch?feature=player_embedded&v=YOUTUBE_VIDEO_ID_HE
" target="_blank"><img src="http://img.youtube.com/vi/YOUTUBE_VIDEO_ID_HERE/0.jpg"
alt="IMAGE ALT TEXT HERE" width="240" height="180" border="10" /></a>
```

Or, in pure Markdown, but losing the image sizing and border:

```
[![IMAGE ALT TEXT HERE](http://img.youtube.com/vi/YOUTUBE_VIDEO_ID_HERE/0.jpg)](htt
```

Referencing a bug by #bugID in your git commit links it to the slip. For example #1.

---

4. Page Editing

This section seeks to explaing how to edit each of the pages in the website and will mainly concern itself with the three files `group.js` and `text.js` present in the `txt` directory and `a.json` present in the `gallery` directory. This will assume that the reader has a sufficient knowledge of Markdown.

If you are not familiar with markdown, go back to the previous section where a tutorial for Markdown is provided. If you don't remember some basics, read the cheatsheet provided with this documentation.