EXERCISE CHEAT SHEET

Below you'll find step-by-step instructions for executing the exercise part of the workshop **Creating a Tech Writing Portfolio with Docs as Code Tooling** by Svetoslav Pandeliev.

Check Jekyll Installation

1. Open a command prompt on Windows (or terminal on macOS) and run the following command:

   ```
   jekyll -v
   ```

2. You should see a version number in response.

   > In case you don't have Jekyll installed, check Creating a Tech Writing Portfolio with Docs as Code Tooling - Preliminary Setup.

Step 1: Fork theme repo

1. Go to github.com and search for `WhatATheme`. Choose the thedevslot/WhatATheme repository from the list of results.

2. Choose the **Fork** button on the right. This will create your own copy of the theme repository. Name your theme repository **my-portfolio**.

3. Run the following command in command prompt (terminal):

   ```
   git clone <url of your theme repository>
   ```

   This will copy the contents of your theme repository in a folder `my-portfolio` on your computer.

Step 2: Build your portfolio site locally

1. In the command prompt (terminal), navigate to the theme repository folder `my-portfolio`.

2. Run `bundle install` and wait for all dependencies to be installed.

3. Run `bundle exec jekyll serve`. When the build process is complete, you'll see a URL in the command prompt (terminal) output: http://127.0.0.1:4000. If you paste this in a browser, you'll see a local preview of your portfolio site.

Step 3: Customize the theme

Open the `_config.yml` and update the following fields. Leave all other fields unchanged.

Field Name	Value	Info
`title`	`<site title>`	Title of your site
`description`	`<desc>`	Desc or subtitle
`email`	`<your email>`	Contact email

Field Name	Value	Info
twitter-username	<your twitter user>	No url, just the user name
linkedin-username	<your linkedin user>	No url, just the user name
github-username	<your github user>	No url, just the user name
author-name	<your name>	Your name
author-about	<your desc>	Short desc about you

Adjust navigation

1. Open the `navbar.html` file in the `_includes` folder.
2. Find the section marked with a comment `<!-- navbar items | right side -->`.
3. Delete not needed sections, leave only **HOME**, **ABOUT**, **CONTACT**.
4. Add a new line for **PROJECTS** in the same section:

```
<a class="navbar-item" href="{{site.url}}{{site.baseurl}}/#project">PROJECTS</a>
```

> You can copy the HTML code from the **CONTACT** line and adjust the values as needed.

Adjust site content to reflect new navigation

1. Open `project.html` in the `_layouts` folder.
2. Copy the main section marked with a comment `<!--Main Section-->`.
3. Create a new file `project.html` in the `_includes` folder.
4. Paste the main section from the `_layouts/project.html` file in the `_includes/project.html` file.
5. Open the `default.html` file in the `_layouts` directory and add `{% include project.html %}` under `{ % include contact.html % }`.

> This will add the **Projects** section just after the **Contact** section in the scrolling home page.

Update **CONTACT** and **ABOUT**

1. Open the `about.html` file in the `_includes` folder.
2. Find the HTML code for the **Blog** button (lines 20-24) and delete it.
3. Open the `contact.html` file in the `_includes` folder.
4. Find the HTML code for the **Tweet me on Twitter** button (lines 18-24) and delete it because it has an icon already underneath.

Adjust footer

1. Open the `footer.html` file in the `_includes` folder.

2. Find the section marked with a comment `<!--Footer Main Section-->` (lines 6-53) and delete it.
3. Update copyright statement to include site owner name.

Step 4: Push changes to GitHub

- Through the **Source Control** pane if using VS Code.
- Via command prompt (terminal), note the git commands below:
  - `git status` shows which files are new, changed, tracked, etc.
  - `git stage` stages your changes, you can choose if you want to stage all or only some of them.
  - `git commit -m "commit msg"` creates a commit with staged changes and a commit message as specified in `"commit msg"`.
  - `git push` pushes commits to your repository.

Step 5: Enable GitHub Pages

1. Enable via the **Settings** tab of your theme repository **my-portfolio**.
2. Check the result of the published website on the Internet.

Step 6: Add GitHub Actions and tryout

1. Go back locally.
2. Create the needed setting for the linter action. It'll check spelling and style of your content against a style guide:
   - Go to `.github/workflows` in your project (create if it doesn't exist) and create a new file `lint.yml`.
   - Paste the code from section Usage in the new file and save the file.
   - Create a new file `.vale.ini` in the `my-portfolio` folder.
   - Get the code for the file from section Repository structure + add lines for the Microsoft package from https://vale.sh/hub/microsoft/.
3. Create the needed setting for the link validation action:
   - Go to `.github/workflows` in your project and create a new file `links.yml`.
   - Paste the code from section Alternative approach in the newfile and save it.
   - Create a new file `lychee.toml` in the `my-portfolio` folder. Add the directories where links should be skipped from the spellcheck with the following line: `exclude_path = ["./_includes", "./_layouts", "./_posts"].`
4. Push to GitHub again and verify the two actions are working properly in the **Actions** tab of your theme repository **my-portfolio**.

Bonus exercise ideas

- Move social icons to the `hero` section and remove the `contact` section altogether:

> Move the social icons from the **Contact** section to the **hero** section, under the title and description of the site, replacing buttons that are there by default - need to copy the `html` code for the social icons from the `contact.html` file to the `showcase.html` file, add another row in the beginning for the envelope icon and mail, and adjust their class from `has-text-black` to `has-text-white` so they are visible on the black background. At the same time, need to remove all unnecesary `html` parts from the `showcase.html` file.

- Fix formatting of the project pages - text alignment, etc.
- Remove opacity setting of the hero image.
- Custom 404 page, unify background image with the homepage.
- Update photo styling in the **About**, remove dotted line frame.
- Update credits to include credits for images, graphics, etc.
- Change the default Twitter icon to the new **X** icon.

> Need to find the Font Awesome shorthand of the new **X** icon (for example, `fa-x-twitter`) and replace this in all files where the Font Awesome shorthand of the default twitter icon appears. Also need to update the Font Awesome version that is used by the site. Open the `head.html` file in the `_includes` folder and replace line 5 with: `<link rel="stylesheet" href="https://use.fontawesome.com/releases/v6.4.2/css/all.css">`.