Camosun College

ICS 211 - Web Applications

Lab 1 - Generating a Static Site Using Hexo and Markdown

Due Date: Lab Demo and Lab Quiz due by Sept. 21

Background and Theory

You are going to be introduced to several technologies in this Lab. Many will be used throughout this course.

Node.js

Node.js is the execution runtime for JavaScript. It's a way to run JavaScript outside of a browser. It essentially turns JavaScript into a server-side scripting language, similar to PHP. It executes JavaScript very, very fast! Node.js contains an extensive module system that adds a lot of functionality. Many applications are also built on-top of Node.js.You can download and install Node.js by going to the Node.js website (https://nodejs.org/en/). You want the **LTS** release. Documentation for Node.js is located at https://nodejs.org/dist/latest-v8.x/docs/api/ (https://nodejs.org/dist/latest-v8.x/docs/api/).

NPM

NPM is the Node Package Manager. Node.js packages are pre-written modules that add some sort of functionality to your application. The public repository of these packages is located at https://www.npmjs.com/ (https://www.npmjs.com/). There are many commands you can use with npm which are detailed in its documentation (https://docs.npmjs.com/). You will explore some of these commands in this lab.

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It has largely replaced XML (Extensible Markup Language) as a format for exchanging data. It is not specific to JavaScript and can be used by any language (despite its name). JSON is basically made up of name-value pairs. Each name and value is surrounded by double-quotes (non-string values are an exception) and separated by a semi-colon. The entire name-value pair is surrounded by curly braces. For example, `{ "instructor" : "jason" }`. Multiple name-value pairs are separated by a comma. For example, `{ "instructor" : "jason", "course" : "ICS 211"}`. Multiple name-value pairs can be put in an array. For example, `[{ "instructor" : "jason"}, { "course" : "ICS 211"}]`.

package.json

Every Node.js project has a *package.json* file in the root folder of the application. This file contains the modules the application depends on as well as some other meta-data. You can go to https://docs.npmjs.com/files/package.json ([https://docs.npmjs.com/files/package.json) to see explanations of all the possible data you can have in this file. All the data is stored as JSON. When the `npm install` command is run in a folder with a *package.json* file, a *node_modules* folder is created and all the modules listed in *package.json* are installed in the *node_modules* folder ready for the application to use. In addition, a *package-lock.json* file is created which "locks" the version numbers of the modules

installed. When distributing your application, you typically don't include the *node_modules* folder. Instead, you distribute the *package.json* and *package-lock.json* files and have the end user run `npm install`. This reduces the size of your application when you distribute it and also guarantees the right modules and versions are installed for the end user.

YAML

YAML stands for YAML Ain't Markup Language. YAML is a superset of JSON (meaning all JSON is valid YAML) and is mainly used for configuration data (again, replacing XML).

YAML is a lightweight language and there really isn't much to the syntax:

- YAML files end in the '.yml' extension
- YAML is case sensitive
- YAML doesn't allow the use of tabs; always use spaces (typically 2 spaces for each indent)
- Sequences of values start with a dash followed by a space
- A key-value pair mapping is done by separating the key with a colon and space from the value
- Like Python, it uses indentation to indicate structure

Many new technologies are using YAML instead of XML or plain text for their configuration files. Hexo and Docker Compose both use YAML. For the gory details on YAML, see http://yaml.org/spec/1.2/spec.html (http://yaml.org/spec/1.2/spec.html).

Markdown

Markdown is a lightweight markup language. It is a shorthand way to write HTML. It uses plain text for formatting. For example, to make this **bold**, I surround it with two asterisks. Markdown is commonly used for README files and posting in forums but can be used for almost any documentation. *This lab is written in Markdown!* Markdown is typically converted into HTML. It can also be converted into epub, mobi, or pdf. The most popular implementation of Markdown is *GitHub Flavored Markdown (GFM)*. A good starting tutorial for GFM is here (https://guides.github.com/features/mastering-markdown/) and a cheat sheet is here (https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf).

Static Site Generators (SSGs)

SSGs take content that you create and generate a website from it. Similar to CMS, you can use pre-made themes and plugins for the HTML, CSS, and JavaScript. You don't need to write any of that yourself! StaticGen (https://www.staticgen.com/) contains a list of several common SSGs. The one that you will use in this lab is called Hexo (https://hexo.io/). It is written in JavaScript and uses the Node.js runtime to compile your markdown.

Themes

All CMSs and SSGs come with Themes (in fact, many of the *same* themes). Themes are the *visual blueprint* for your site. Themes dictate how your content will be laid out, site colors, site fonts, and how menus, images and other content will be integrated into your site. The advantage of using a Theme is that all that work is done for you. In addition, many themes are free. The main disadvantage of using a theme is that you are getting a *cookie cutter* layout, dictated by the theme author. There may be some features of the theme that you want to customize. This requires expert knowledge in whatever technologies are used to make the theme (typically HTML, CSS, JavaScript and if a WordPress theme, PHP).

Tasks

Task 1 - Setting up Hexo

Note: This Task is intended to be done as a group with the Instructor

1. First, ensure you have Node.js and npm installed on your machine. On Windows, open up a Git Bash Shell and verify by typing:

   ```
   node --version && npm --version
   ```

   As of Fall 2018, Node.js LTS should be at version 8.x

2. Install Hexo. Hexo is an NPM (Node Package Manager) module. To install Node.js packages, you use the command `npm install`. You can install packages locally, in your project folder or globally which means they are installed in a global location. On Windows, this will be in the AppData/Roaming Profile. To install globally, you use the `-g` switch. In a Git Bash Shell type the following:

   ```
   npm install -g hexo-cli
   ```

3. Now you are going to create your Hexo project. Name the project **the first letter of your first name followed by your last name all in lowercase**. For example, if my name were John Smith, my project's name is *jsmith*. In a folder where you want to store your website, i.e. a folder for this course, in a Git Bash Shell, type the following:

   ```
   hexo init <your Hexo Project's name>
   ```

   Wait for it to finish.

4. Change into the directory it created. For example: `cd jsmith`

5. Now take a directory listing: `ls -la`

   It should have the following files and folders:

   ```
   . (parent project folder)
   |-- node_modules/
   |-- scaffolds/
   |-- source/
   |      +-- _posts/
   |-- themes/
   |      +-- landscape/
   |-- .gitignore
   |-- _config.yml
   |-- package.json
   |-- package-lock.json
   ```

   Let's examine what each is:

   - **package.json**, **package-lock.json**, and the **node_modules/** are to do with the installation of the modules required for the application. This is described in the *package.json* part of the *Background and Theory* section of the lab above.
   - **scaffolds/** Hexo uses files in this folder as templates for new pages and photos.
   - **source/** this folder is where you put your site's content. The files in this folder are processed by Hexo. The **_posts/** sub-folder is where new blog posts go.
   - **themes/** contains the theme used for the application. Currently, defaults to landscape. You will change the theme in **Task 3**.
   - **_config.yml** this is the site configuration file. It uses YAML. Many site settings are configured here. You will modify this file in this lab.
   - **.gitignore** tells git to not commit the listed files.

6. Even before making any changes, you should have a ready application. Type: `hexo server` in the Git Bash Shell. This starts a local Node.js web server.

7. In a browser, go to `http://localhost:4000/`. There's your website! And you didn't have to code anything!! Of course, it isn't very personalized. Let's fix that!

Note: The rest of the Tasks for this lab should be done individually

Task 2 - Adding Content

Disabling Browser Cache:

1. When developing websites on your machine, it is a good idea to disable the browser's cache. This prevents the problem of the browser loading web pages from its cache; ignoring any changes you might have made. **Activate the Tab that has your website in it.**

    - **Firefox:** Activate Web Developer Tools and activate the Network Tab by hitting `CTRL+SHIFT+e`. On the right hand side, you'll see a box labeled `Disable cache`. Check that box and leave the Web Developer Tools activated.
    - **Chrome:** Activate Developer Tools by hitting `F12`. Click on `Network` and check the box labeled `Disable cache`. Leave the Developer Tools activated.

BrowserSync:

2. In addition to disabling your browser's cache, there's a tool called **BrowserSync** that can ease your workflow. Whenever you make any changes in any file in your project's sub-folders, BrowserSync will automatically re-start the Hexo server and refresh the browser for you. What an awesome tool!! Hit `Ctrl+C` in the Git Bash Shell that you started the Hexo Server in to stop it. Now type `npm install hexo-browsersync --save-dev`. This will actually modify your project's *package.json* file and add *hexo-browsersync* under `devDependencies`. This means that the package is not required to run the project; it's only required for developing it.

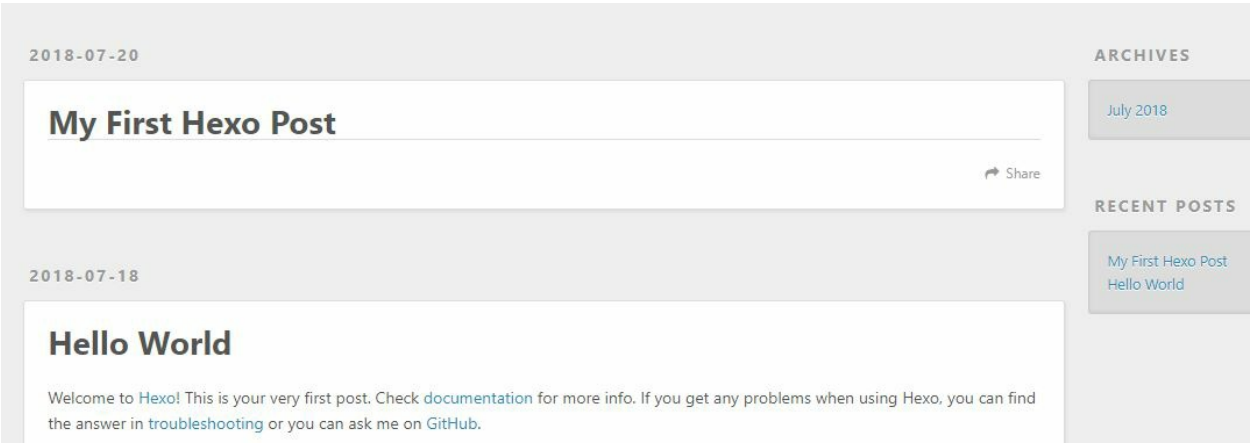
3. Now run `hexo server` like before. You'll see some additional output in the console to indicate BrowserSync is running.

Adding a Basic Text Post:

4. Adding new content to your site is pretty easy. Open a *new* Git Bash Shell in your project's folder (don't stop the hexo server/BrowserSync in your other Shell). Type: `hexo new "My First Hexo Post"`.

    Note: An alternative to `hexo new` is `hexo new draft <title>`. This means your post isn't published right away. You can keep working on it and publish it later with `hexo publish <title>`.

5. If you disabled the cache and have BrowserSync running, you should see the Browser automatically refresh. You should now see a new empty post on your webpage with the title `My First Hexo Post` and it will also be added as a link under `Recent Posts` on the right-hand side.

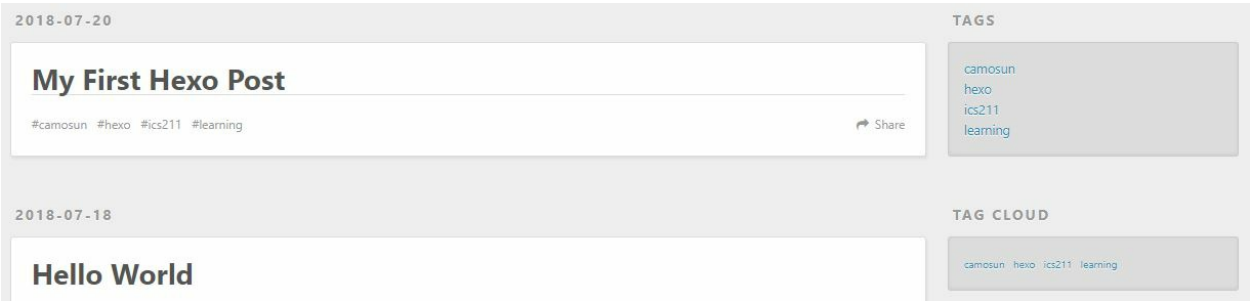


6. After creating the post, a new file will in your project's *source/_posts* folder. It will be the same name as the title of your post with dashes between each word. You'll notice it has a `.md` extension indicating it's a markdown file! Open this file in Visual Studio Code.

7. At the beginning of this file is what is known as *front matter* (the front matter is actually generated from files in the *scaffolds* folder). This is written in YAML. The title and date are already filled in for you. The other option here is tags. You can add tags based on the subject of the post. For example, something like:

   ```
   tags:
     - learning
     - hexo
     - ics 211
     - camosun
   ```

   Add some tags now. If you still have your site in the browser with BrowserSync running, you should see the changes loaded as soon as your markdown file is saved.



8. The three dashes `---` mark the beginning and end of the YAML. After the second set of dashes, you can write your post using Markdown. **Write one now!** Refer to the *Markdown section* in the *Background and Theory section* of this lab for some useful links on Markdown. Note that if you can't think of anything to write and just want to fill in some text to see what it looks like, there's a useful tool for that. It's called the **Lorem Ipsum Generator** (Lorem Ipsum means placeholder text). You can find it at https://loremipsum.io/ (https://loremipsum.io/). You select the number of paragraphs you want then copy the text. Keep a note of this tool, you may need to use it again for other labs!

Adding Assets:

9. Now what if you want to post a PDF or an image? Hexo has the concept of **Asset Folders**. This feature is disabled by default. To enable it, you need to change a setting in your project's **_config.yml** file. Open this file and find the `#Writing` section. Change the `post_asset_folder` setting to `true`.

10. Create a new post: `hexo new "ICS 211 Lab 1"`. If you look under your project's *source/_posts* folder, you'll notice that not only has the markdown file been created for this second post (like before) but a folder with the same name has also been created. This is called the post's *Asset folder* and it is in here you place any assets like PDFs or images.

11. Move the PDF of this lab (or a PDF of your choosing) and place it in this Asset folder. **Note down the pdfs name*

12. Now, open the *ICS-211-Lab-1.md* file. First, add some tags: `- ics211, - lab1` for example. Then add this line below the three dashes (---):

    ```
    {% asset_link ics-211-lab-1-rev0.pdf ICS 211 Lab 1 %}
    ```

    This is called a **Tag Plugin**. This particular plugin (asset management) is built into the core Hexo framework. Hexo has many plugins you can add. You can find them here (https://hexo.io/plugins/) (some of these are for Hexo 2 not 3). For an image, the tag is somewhat similar. If you had an image named *example.jpg*, the tag plugin would look like:

    ```
    {% asset_img example.jpg This is an example image %}
    ```

    For other types of content, you can use the tag plugins listed here (https://hexo.io/docs/tag-plugins.html). Whenever you use these, Hexo will generate the appropriate HTML and use the correct path to the asset for the Static Site. If you still have your local hexo server with BrowserSync running, **you won't see anything by adding the tag**. These will only work when you have generated the Static Site. But before you do that, you need to customize your site.

Task 3 - Customizing Your Website

Changing the title, author, description:

1. Start by customizing the title of your website. Open your project's **_config.yml** file. You'll be making changes under the `Site` section. Change the title. For example, for a blogging site, perhaps give it some sort of catchy name. For a more professional portfolio site, perhaps you just want your name. If you can't think of anything right now, `ICS 211 Lab 1 Hexo Website` will do. Notice you can also add a subtitle. Here, you could add `by <your name>`.

2. To see the changes you just made, you will need to hit `CTRL+C` in the Git Bash Shell that you started the hexo server in. BrowserSync doesn't work in this case because *_config.yml* is in the parent folder. Re-run `hexo server` and do a manual refresh of the browser. You should see the changes.

3. Next change the author in **_config.yml**. Other fields you will notice here are **description**, **keywords**, **language**, and **timezone**. **language** and **timezone** can be left blank. The language defaults to English and as long as the timezone is set up properly on your system, it can also be left blank. **keywords** is for the keywords meta tag. Search Engines *used to* read this tag. But due to ongoing abuse, search engines now largely ignore it. You can leave it blank. You can write a single sentence summary of your website beside **description**. This will create some meta tags that could be used by social networking platforms (an example is the og tags. *og* stands for *open graph protocol* and is used by facebook). You can see what tags it generates by re-starting the hexo server and viewing source.

Changing the theme:

4. The next thing you may want to change is the theme. Go to https://hexo.io/themes/ (https://hexo.io/themes/). There are 200+ themes here (although some go to dead links)!! Pick one that you like (don't worry about what language the example is in). You can click on any of the images of the themes to take a closer look. Most themes are designed for blogging sites. But some themes have been designed for specific purposes. For example, the `doc` theme is designed for a website that hosts documentation (a SSG is perfect for documentation sites). The `Corporate` theme is designed for small business. The `Milan` theme is designed for a small e-commerce site

(and costs money). It uses a 3rd party service, snipcart, for the shopping cart functionality. If you're having trouble deciding, `CleanBlog` is a good basic blogging theme you can use. `Edinburgh` is a good basic portfolio-based theme.

5. Once you have a theme picked, you need to **click on the title of the theme, not the image**. This will typically bring you to the theme's github page. Follow whatever directions are there. Typically, it involves cloning the repository into your project's theme folder. For example, for the CleanBlog theme, run this in a Git Bash Shell in your project's folder:

   ```
   git clone https://github.com/klugjo/hexo-theme-clean-blog.git themes/clean-blog
   ```

   This is just an example, make sure to follow the directions of the theme you chose.

6. **Delete the *.git/* folder of your theme**. In a Git Bash Shell:

   ```
   rm -rf .git/
   ```

   In the next lab, you will create a Git repository for your project and having a sub-folder with a git repository will cause issues.

7. You will also need to change the theme being used in your project's **_config.yml** file. Under the `Extensions` section, change the theme from landscape to your new theme. It must match the name of the folder that was created when you cloned the theme into your project.

   Note: Most themes come with their own *_config.yml* file. This is different from your site-wide *_config.yml* file. The theme's *_config.yml* file is for specific settings for the theme. **Don't confuse the two!!** In the theme's *_config.yml* file you can change things like menus, images used, social networking accounts, and third-party tools (like Disqus, a commenting service).

8. Now run `hexo server`. You should see your new theme activated. If you're happy with it, you can delete the default *landscape* theme folder under your project's *themes* folder.

Task 4 - Generating Your Static Site

1. Right now, the *hexo server* loads your website to a local server. But it hasn't generated the static content yet that will deployed on a remote server. **To do so, you issue the command `hexo generate`**. If the *hexo server* is running, hit `CTRL+C` to stop it and type `hexo generate`.

   Note: You can even run *hexo generate* so that it monitors your project and automatically re-generates the static files when it detects changes. To do so, you run it with the `-w` switch.

2. Once you issue the command, the console will show you all the static files it generated for your website. These will all be stored under a new sub-folder in your project's folder called *public/*.

Task 5 (Optional - Not for Marks) - Adding a Third Party Service

Warning: This *optional* task requires you to sign up for a third party service. Your credentials may be stored out of country. If you are uncomfortable with this, do not do this task.

Note: In order to do this task, your chosen theme must have Disqus integration built in. If it doesn't, skip this task.

1. As discussed, a huge disadvantage of using a SSG is its inability to deal with dynamic content. For example, if you are using a SSG for a blogging site and you wish people to make comments on your posts, how do you that? Fortunately, there are many third party services out there that allow you to add dynamic behavior to your static site.

One such service is called **disqus**. Disqus is a blog commenting hosting service. They literally host comments! Go to https://disqus.com/ (https://disqus.com/) and click on the *Get Started* button.

2. You need to sign up to Disqus. It's possible you may already have an account if you have ever visited a site that uses Disqus. If not, make an account here. You will have to agree to their Terms of Service.
3. Once you sign in, you will be presented with a page with two large white boxes. Click on the one that says *I want to install Disqus on my site*.
4. Next, will be a page that shows your name as the site owner and it asks you your Website Name and Category. Come up with a name and fill it in the text field. It has to be unique. This is called the **Disqus shortname**. It's part of a URL, so stick to lowercase letters. Write down or remember what you used here.
5. Pick a Category. What will you be blogging about? Then click on the *Create Site* button.
6. The next page is the *Select a Plan* page. At the bottom, below the Plus plan, is the Free one (smaller grey square). Click on the *Subscribe Now* button in this square.
7. You don't need to do any other steps on the Disqus site. Instead, open up your theme's *_config.yml* file (the one in your theme's folder). In this file, look for a *Comments* section. Then beside where it says `disqus`, enter in your shortname from the previous step.
8. If not already running, run `hexo server`. If you click on any of your posts, you should see a Disqus comment section at the bottom!!

Lab Submission (see top of lab for due date)

1. Demo to the lab instructor (10 marks):
    - show that you have added a post to your Hexo site and it contains tags
    - show that the title of your site has been customized
    - show that you have changed the theme
    - show you have generated static content for your site by showing your project's *public/* folder in Windows Explorer
2. Complete the Lab 1 Quiz in D2L (10 marks).