Why I love data files in Jekyll

Chen Hui Jing / @hj_chen
Surname: Chen
First name: Hui Jing

@hj_chen
What are data files?

- **Custom data** that can be accessed via the Liquid templating system
- Supported formats:
  - YAML
  - JSON
  - CSV
  - TSV
- Helps **avoid repetition** in your templates
- Keeps the focus on the **content**
Using data files (1/2)

Place data files (supported formats only) in the _data folder

```
<table>
<thead>
<tr>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>meetups.yml</td>
</tr>
<tr>
<td>speakers.yml</td>
</tr>
<tr>
<td>videos.yml</td>
</tr>
</tbody>
</table>
```

Jekyll will recognise them during site generation

Data will be accessible via `site.data.NAME_OF_FILE`
Using data files (2/2)

speakers.yml

```yaml
gloria:
  name: "Gloria Soh"
  twitter: "Gloriasohss"
  shortcode: "gloria"
  bio: "Gloria is a CSS queen who is bitcoin rich"

jacob:
  name: "Jacob Tan"
  twitter: "jacobtyq"
  shortcode: "jacob"
  bio: "Plays too much Factorio... and Developer by"

sarah:
  name: "Sarah Lienert"
  twitter: "Sazzar"
  shortcode: "sarah"
  bio: "Sarah likes to bake things for Talk.CSS an"
```

speakers.html

```html
<ul class="l-speakers c-speakers">
  {% for speaker in site.data.speakers %}
    <li class="l-speaker c-speaker">
      <figure>
        <img class="c-speaker__img" src="{{ site.url }}/assets/speaker.jpg">
        <figcaption><a class="c-speaker__link" href="{{ speaker[1].github }}"{% if speaker[1].twitter %}">
          <span class="c-speaker__link">@{{ speaker[1].twitter }}</span>
        </a>{% else %}
        </figcaption>
      </figure>
      <p class="c-speaker__bio">{{ speaker[1].bio }}</p>
    </li>
  {% endfor %}
</ul>
```
Liquid templating engine

Open-source template language created by Shopify and written in Ruby

Features include:

- Logical and comparison operators (==, !=, <, >, <=, >=, or, and)
- Truthy and falsy
- Control flow
- Iteration
- Variables
- Filters
Truthy and falsy in Liquid

All values in Liquid are truthy except `nil` and `false`. Strings and arrays, even when `empty`, are truthy.

Check for emptiness using `blank` or `empty`.
# Control flow

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td>Executes a block of code only if a certain condition is true</td>
</tr>
<tr>
<td>unless</td>
<td>The opposite of if - executes a block of code only if a certain condition is not met</td>
</tr>
<tr>
<td>elsif / else</td>
<td>Adds more conditions within an if or unless block</td>
</tr>
<tr>
<td>case / when</td>
<td>Creates a switch statement to compare a variable with different values. case initialises the switch statement, and when compares its values</td>
</tr>
</tbody>
</table>
Relevant questions

- How to express logic "and", "or", "not"?
- Does Liquid have a does not contain or not in array operator?
unless

Your best substitute for not

- Controls rendering when conditional returns nil or false
- Prevent rendering of empty HTML elements
- Useful when used in conjunction with front matter variables
## Iteration

<table>
<thead>
<tr>
<th>Iterator</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>for</td>
<td>Repeatedly executes a block of code</td>
<td>limit, offset, range, reversed</td>
</tr>
<tr>
<td>cycle</td>
<td>Loops through a group of strings and outputs them in the order that they were passed as parameters (must be used within for loop)</td>
<td>cycle group</td>
</tr>
<tr>
<td>tablerow</td>
<td>Generates an HTML table (has to be wrapped with &lt;table&gt; tags)</td>
<td>limit, offset, range, cols</td>
</tr>
</tbody>
</table>
% for post in site.posts limit:1 %
<article class="c-upcoming-event {{ post.css }}">
  {% unless post.special %}
  <header class="c-event__header">
    <h2 class="c-event__title">
      {{ post.title }}
    </h2>
    <p><strong>Event location:</strong> {{ post.location }}</p>
    <p><strong>Event date:</strong> {{ post.eventDate }}</p>
  </header>
  <a href="https://www.meetup.com/SingaporeCSS/events/267908738">
    The SingaporeCSS monthly meetup is an event organised by SingaporeCSS, the meeting group for all things CSS on this little red dot. Join our 3rd event - no more sister discussion and tantalizing reflections. If you have stuck with any CSS issues, there is someone someone who can help you.
  </a>
  {{ post.content | markdownify }}
</article>
{% endfor %}
Display list of 3 posts before the latest one

<h2>Past meetups</h2>
<ul class="1-past-events c-past-events">
  {% for post in site.posts offset:1 limit:3 %}
  <li class="1-past-event c-past-event">
    <a class="c-past-event__link" href="{{ postpermalink }}">
      <h3>{{ post.title }}</h3>
    </a>
  </li>
  {% endfor %}
</ul>
## Variables

Possible to create new Liquid variables in addition to predefined ones

<table>
<thead>
<tr>
<th>assign</th>
<th>Creates a new variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>capture</td>
<td>Captures the string inside of the opening and closing tags and assigns it to a variable</td>
</tr>
<tr>
<td>increment</td>
<td>Creates a new number variable, and increases its value by one every time it is called</td>
</tr>
<tr>
<td>decrement</td>
<td>Creates a new number variable, and decreases its value by one every time it is called</td>
</tr>
</tbody>
</table>

Custom front matter variables can be defined and accessed via the page.\texttt{.VARIABLE\_NAME} syntax
Establishing relationships across data files

**speakers.yml**
- **Gloria Soh**
  - Name: Gloria Soh
  - Twitter: @Gloriasohs
  - Shortcode: "gloria"
  - Bio: "Gloria is a CSS queen who...

- **Jacob Tan**
  - Name: Jacob Tan
  - Twitter: @jacobtyq
  - Shortcode: "jacob"
  - Bio: "Plays too much Factorio...

- **Sarah Lienert**
  - Name: Sarah Lienert
  - Twitter: @SazzarJ
  - Shortcode: "sarah"
  - Bio: "Sarah likes to bake thi...

**videos.yml**
- **s2502**
  - Title: "CSS grid for noobs"
  - Link: "https://youtu.be/0Laanm#
  - Shortcode: "s2502"
  - Description: "Shirlaine and Glo...

- **s2601**
  - Title: "Frontend development for...
  - Link: "https://youtu.be/xj-Mbc#
  - Shortcode: "s2601"
  - Description: "Ardy shares how f...

- **s2602**
  - Title: "Faux sub-grid"
  - Link: "https://youtu.be/ChnITW#
  - Shortcode: "s2602"
  - Description: "Using CSS grid bu...

**meetups.yml**
- **25**
  - Colour: "lime"
  - Hex: "#00FF00"
  - Rgba: rgba(0, 255, 0, 1)

- **Videos**
  - Ref: s2501
  - Ref: s2502

- **Speakers**
  - Ref: ed
  - Ref: gloria
  - Ref: shirlaine

- **26**
  - Colour: "indigo"
  - Hex: "#4b0082"
  - Rgba: rgba(75, 0, 130, 1)
  - Text: "#fff"

**Videos**
- Ref: s2601
- Ref: s2602
Using `assign` to simplify custom data (1/2)

40:

```yaml
colour:
  name: lightseagreen
  hex: "#20b2aa"
  rgba: rgba(32, 178, 170, 1)
videos:
  - ref: s4001
  - ref: s4002
  - ref: s4003
  - ref: s4004
speakers:
  - ref: alex
  - ref: ujjwal
  - ref: linhan
  - ref: mike
```

```liquid
{% assign meetup = site.data.meetups[page.meetup] %}
{% for videos in meetup.videos %}
  {% assign video = site.data.videos[video.ref] %}
  <div class="c-videos">
    /* other code */
    {% for videos in meetup.videos %}
      {% assign video = site.data.videos[video.ref] %}
      <div class="c-video">
        <a class="c-video__link" href="{{ video.link }}">
          <img class="c-video__img" src="{{ site.url }}"></a>
        <p class="c-video__desc">{{ video.description }}</p>
      </div>
    {% endfor %}
  </div>
{% endfor %}
/* moar code */
</div>
```
Using assign to simplify custom data (2/2)

40:
  colour:
    name: lightseagreen
    hex: "#20b2aa"
    rgba: rgba(32, 178, 170, 1)
  videos:
    - ref: s4001
    - ref: s4002
    - ref: s4003
    - ref: s4004
  speakers:
    - ref: alex
    - ref: ujjwal
    - ref: linhan
    - ref: mike

{% if meetup.colour %}
{% assign colour = meetup.colour %}
  <div class="c-colour">
    {% if colour.text %}
      <div class="c-swatch" style="background-color:#{ colour.text }">
        <div class="c-swatch__txt">
          <p>{{ colour.name }}</p>
          <p>{{ colour.hex }}</p>
          <p>{{ colour.rgba }}</p>
        </div>
      </div>
    {% else %}
      <div class="c-swatch" style="background-color:#{ colour.hex }"></div>
    {% endif %}
  </div>
  <h4> CSS colour of the month </h4>
{% endif %}
Count of speakers with `forloop.length`

Variables can be used outside the `for` loop they were declared in.

```plaintext
{% for speaker in site.data.speakers %}{% assign count = forloop.length %}{% endfor %}

So far, {{ count }} lovely people have spoken at Talk.CSS, we want you to be one of them too. If you’re thinking, “but I have nothing to talk about”, we can help. Anything related to CSS will work. It could be something you debugged at work, or about a new property you read about, or something you built, or even a rant about why something doesn't work the way you want it.

 kì speakers.md kì
Using `capture` for complex strings

Variables created through `{{% capture %}}` are strings.

Can create complex strings using other variables created with `assign`.

```{% if page.css %}
{{% capture special_styles %}}/assets/css/{{ page.css }}.css{{% endcapture %}}
<link rel="stylesheet" href="{{ special_styles | prepend: site.baseurl }}">{{% endif %}}```
SingaporeCSS website workflow (1/2)

**speakers.yml**

```yaml
ollie:
  name: "Ollie Monk"
  twitter: "ollie_monk"
  shortcode: "ollie"
  bio: "A modern Renaissance man with a wide range of skills and interests."

jinjiang:
  name: "Zhao Jinjiang"
  twitter: "zhaojinjiang"
  shortcode: "jinjiang"
  bio: "A 0.1x engineer. Loves football. Does front end."```

**post.markdown**

```yaml
---
layout: post
title: "Talk.CSS #43"
date: 2019-08-07 19:00:00 +0800
location: H000
event-date: 2019-09-04 19:00:00 +0800
categories: meetup
meetup: 43
event-id: 261692436
novideos: true
---```
If you are also using YML...

There is a difference between lists and dictionaries.

- All members of a list are lines at the same indentation level starting with a "-  
- A dictionary is represented by key/value pairs in the form key: value, where the colon must be followed by a space
- Complicated data structures like lists of dictionaries, or dictionaries with lists as values are possible
videos.yml

s4301:
  title: "Lean animations"
  link: "https://youtu.be/_QN_ahQN01c"
  shortcode: "s4301"
  description: "Jinjiang shares some very practical tips on how to make your animations leaner and more efficient."

s4302:
  title: "An exploration of CJSS - A CSS based web framework"
  link: "https://youtu.be/x1U9tPwRiiw"
  shortcode: "s4302"

s4303:
  title: "Sometimes you gotta abuse Grid"
  link: "https://youtu.be/9DwzDzCNDwI"
  shortcode: "s4303"
  description: "Sheldon encountered a problem and used CSS Grid to solve it."

post.markdown

---

layout: post

---

title: "Talk.CSS #43"

date: 2019-08-07 19:00:00 +0800

location: H000

event-date: 2019-09-04 19:00:00 +0800

categories: meetup

meetup: 43

event-id: 261692436

---
SingaporeCSS website

Source code: https://github.com/SingaporeCSS/singaporecss.github.io

On the web: https://singaporecss.github.io
Thank you!

https://www.chenhuijing.com

@hj_chen

@hj_chen

@huijing

Font is Public Sans by USWDS