Создаем блог используя Jekyll и GitHub Pages

Недавно я перенес блог с WordPress на Jekyll — это генератор статических сайтов. Jekyll был специально разработан для создания минималистичных блогов, которые затем можно разместить на GitHub Pages. Написание статей и создание тем для Jekyll удивительно просты, однако настройка сайта заняла у меня значительно больше времени, чем ожидалось.

В этой статье мы рассмотрим следующие моменты:

- как быстрее всего запустить блог на Jekyll;
- как избежать основных ошибок при работе с Jekyll;
- как перенести материалы из WordPress, как использовать свое доменное имя и как писать статьи в любимом редакторе;
- как создавать темы для Jekyll используя Liquid;
- несколько новых функций из Jekyll 2.0, включая поддержку Sass, CoffeeScript и коллекций.

Назначение Jekyll

Том Престон-Вернер (Tom Preston-Werner) создал Jekyll, чтобы вести блог используя простой статический сайт на HTML. Всё содержимое сайта хранится на GitHub, который, к тому же, осуществляет контроль версий. Целью было избавиться от сложности, свойственной другим блог-платформам, чтобы при этом можно было вести блог как настоящий хакер.

Октокот, который является маскотом Jekyll. (Изображение принадлежит: GitHub)

Jekyll берет контент написанный в маркдаун, применяет к нему шаблоны и выдает на выходе готовый, полностью статический сайт. GitHub Pages отдает сайт прямо из GitHub-репозитория, так что вам не нужно иметь дело с хостингами.

Вот примеры сайтов, созданных с использованием Jekyll:

- Блог Барри Кларка (Barry Clark) (это мой)
- Блог Зака Холмана (Zach Holman)
- CSS Wizardry
- Jekyll
- Лэндинг и внутренние страницы HealthCare.gov
- Development Seed

Преимущества статики

- **Простота**
  Jekyll сводит все к абсолютному минимуму, избавляясь от сложных составляющих:

  - **Никаких баз данных** В отличие от WordPress и других систем управления контентом (CMS), Jekyll не использует базы данных (БД). Все страницы перед публикацией преобразуются в статический HTML. Это прекрасно с точки зрения скорости загрузки страницы, так как во время загрузки не происходит никаких запросов к БД.

  - **Никаких CMS** Просто пишите в Markdown — Jekyll сам применит шаблоны к контенту и сгенерирует статический сайт. GitHub может исполнять роль CMS, если нужно, потому что он позволяет редактировать контент.

  - **Быстрый** Jekyll быстрый, потому, что в нём нет ничего лишнего и он не использует базы данных — он просто собирает статические страницы. Мой основной шаблон Jekyll Now создает всего три HTTP-запроса, включая картинки и иконки социальных сетей!

  - **Минималистичный** Большинство сайтов на Jekyll не содержит никакой лишней функциональности или возможностей, которые вы не используете.

- **Контроль представления** Тратьте меньше времени на сложные шаблоны, написанные другими людьми, и больше — адаптируя простой базовый шаблон или создавая свой собственный.

- **Безопасность** Большинство уязвимостей, которые есть у платформ вроде WordPress, отсутствуют в Jekyll, потому что здесь нет CMS, нет баз данных или PHP. Так что вам не нужно тратить массу времени устанавливая обновления, закрывающие дыры в безопасности.

- **Удобный хостинг** Это просто удобно, если вы уже используете GitHub, вот и все. GitHub Pages бесплатно соберет и выложит сайт, использующий Jekyll, и одновременно реализует для него контроль версий.

Давайте попробуем

Есть несколько способов разобраться с Jekyll, у каждого свои особенности. Вот несколько вариантов:

- Установите Jekyll локально используя консоль, создайте заготовку нового сайта командой `jekyll new`, соберите его командой `jekyll build` и выложите. (Веб-сайт Jekyll показывает процесс.)
- Клонируйте репозиторий с заготовкой на локальную машину, установите Jekyll локально из консоли, внесите правки, соберите локально, выложите.
- Форкните репозиторий с заготовкой, измените, выложите.

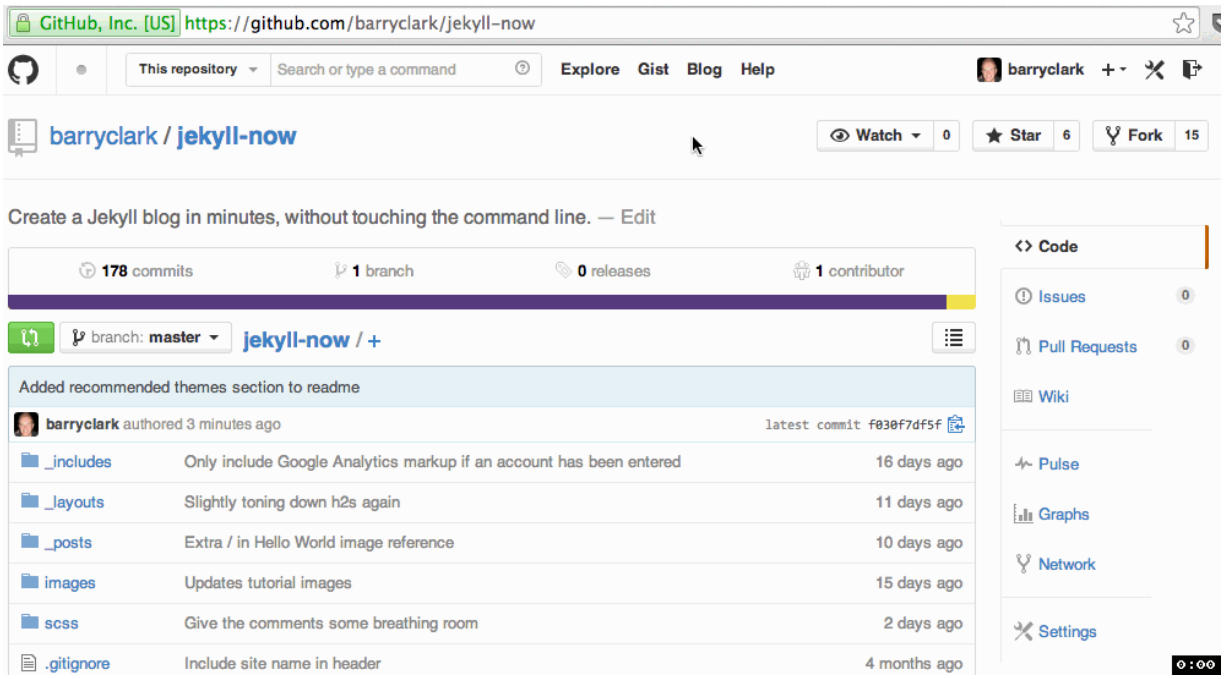
Давайте начнем с самого простого и быстрого варианта: форкнем репозиторий с заготовкой. Это позволит запустить проект в считанные минуты, и нам не придется устанавливать все зависимости. Вот что мы сделаем на GitHub.com прямо в браузере:

1. Создадим сайт используя Jekyll
2. Бесплатно разместим на GitHub Pages.
3. Изменим так, что бы он включал наше имя, аватар и ссылки на социальные сети.
4. Опубликуем наш первый пост!

1. Форкните заготовку

Начнем с создания форка репозитория — это хорошая практика, о которых мы упоминали в статье. Этот способ направит нас по верному пути и сбережет уйму времени.

Я уже создал нам репозиторий. Откройте Jekyll Now и нажмите кнопку «Fork» в верхнем правом углу страницы, чтобы создать форк темы блога в вашу учетную запись на GitHub.



Инструкция к шагам 1 и 2. (Картинка в высоком разрешении)

Начинать с форка отличная идея, ведь это позволит выяснить что из себя представляет Jekyll без необходимости поднимать локальное окружение для разработки, устанавливать зависимости и разбираться с процессом сборки.

Проблема №1: Создание сайта на базе Jekyll через терминал может раздражать и занимать много времени, так как надо **установить и настроить зависимости**, например, Ruby и RubyGems. Позвольте GitHub Pages собирать ваш сайт, пока не возникнет реальная причина делать сборку локально.

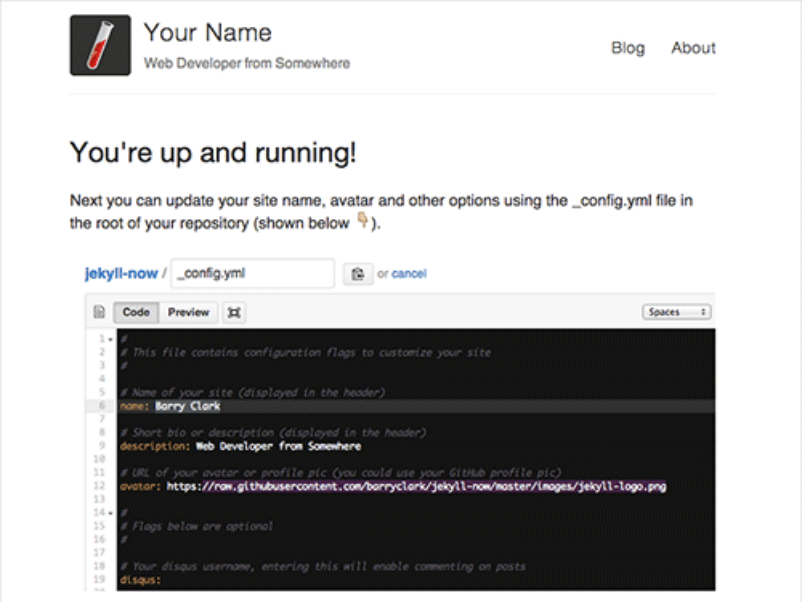
2. Разместим сайт на вашем GitHub-аккаунте

Как пользователь GitHub, вы можете бесплатно создавать «пользовательские» сайты (в отличии от веб-сайтов «проектов»), которые будут доступны по адресу `http://yourusername.github.io`. Идеально подходит для размещения блога на Jekyll!

Лучше всего в этом то, что вы просто помещаете Jekyll-блог в ветку `master`, после чего GitHub Pages сам соберет статический сайт и будет его раздавать. Вам вообще не нужно беспокоиться о процессе сборки — об этом уже позаботились.

Нажмите кнопку «Settings» в форке репозитория (меню справа) и измените имя репозитория на `yourusername.github.io`, заменив `yourusername` на ваше имя пользователя на GitHub.

Скорее всего, сайт станет доступен немедленно, это можно проверить открыв `http://yourusername.github.io`. Если ещё не доступен — не беспокойтесь, на Шаге 3 мы выясним как принудительно вызвать сборку.



Базовая тема блога непосредственно после форка будет выглядеть следующим образом. (Источник:Jekyll Now) (В высоком разрешении)

Уф! Мы быстро продвигаемся, мы уже запустили сайт на Jekyll! Давайте сделаем шаг назад и рассмотрим наиболее часто встречающиеся проблемы, которые стоит иметь ввиду, размещая Jekyll-блог на GitHub Pages.

Проблема №2: нужно понимать различия между размещением на GitHub сайта пользователя и страницы проекта. Для пользовательского сайта (который мы делаем) не нужно создавать никакие ветки: `master` и так сконфигурирована нужным образом, чтобы обрабатывать с помощью Jekyll всё, что в неё помещают, и создавать статический сайт. Нет нужды создавать ветку `gh-pages`.

Проблема №3: использование сайта для проекта немного все усложняет, так как сайт будет находиться в поддиректории. URL будет выглядеть так: `http://yourname.github.io/repository-name`, что вызовет проблемы с шаблонами в Jekyll: например, битые ссылки на картинки и невозможность посмотреть сайт локально.

Проблема №4: для Jekyll существует масса плагинов, но GitHub Pages поддерживает всего несколько из них. Если подключить плагин, который не поддерживается, Jekyll не сможет собрать сайт, так что строго придерживайтесь списка поддерживаемых плагинов. Благо мой любимый плагин есть в списке: Jemoji — он позволяет добавлять в статьи emoji, как на GitHub или Basecamp.

3. Настраиваем сайт

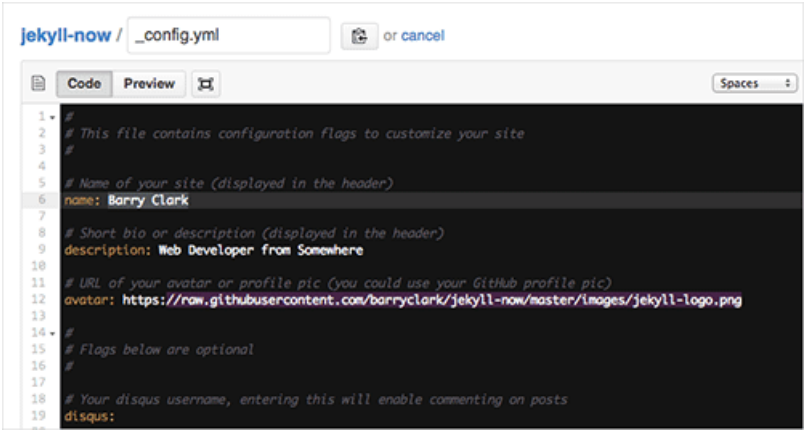
Теперь можно изменить имя сайта, его описание, аватар и прочие настройки отредактировав файл `_config.yml`. Эти пользовательские переменные для удобства были вынесены отдельно, и они подтянутся в тему сайта во время сборки.

Изменение `_config.yml` (или любого файл в репозитории) вызовет повторную сборку сайта. Увидеть результат можно будет через пару секунд по адресу `http://yourusername.github.io`. Если после Шага 2 сайт не появился, то он появится после этого действия.

Вперед, адаптируйте сайт под себя, изменяя переменные в `_config.yml` и коммитя изменения.

Вы можете править файлы одним из трех способов. Выберите тот, который вас больше устраивает:

- Редактируйте файлы непосредственно в браузере на сайте GitHub.com в вашем репозитории `username.github.io` (как показано ниже).
- Используйте сторонний редактор, поддерживающий работу с GitHub, например, Prose, разработанный Development Seed. Он оптимизирован для работы с Jekyll и позволяет легко редактировать Markdown, создавать черновики и загружать изображения.
- Клонируйте репозиторий, внесите правки локально, затем пушните все обратно в репозиторий на GitHub (у Atlassian даже есть гайд на эту тему).



Редактируем _config.yml на GitHub.com. (Источник: Jekyll Now) (В высоком разрешении)

Проблема №5: Не думайте, что нужно выполнять `jekyll build` локально, чтобы внести изменения в сайт на Jekyll — GitHub Pages сделает это за вас. Достаточно поместить файлы, которые нужно собрать, в ветку `master` репозитория с вашим сайтом или в `gh-pages` любого другого репозитория, после этого GitHub Pages соберет их используя Jekyll.

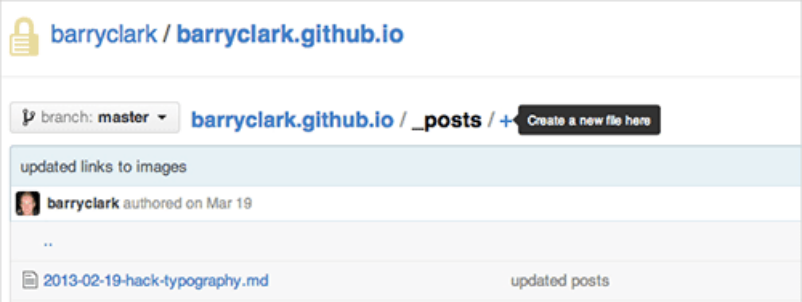
4. Публикуем первую статью

Теперь сайт настроен, работает и отлично выглядит. Можно опубликовать первую статью:

1. Отредактируйте `/_posts/2014-3-3-Hello-World.md`, удалите «рыбу» и введите свой текст. Если нужно освежить в памяти основы использования Markdown, используйте шпаргалку Адама Причарда (Adam Pritchard).
2. Измените имя файла, что бы оно включало текущую дату и заголовок

поста. Jekyll требует определённый формат именования: year-month-day-title.md.
3. Обновите заголовок. Переменные в начале файла называются вводным блоком, мы рассмотрим их более подробно немного позже. В данном случае они определяют заголовок статьи и используемый шаблон. Существуют и другие переменные, которые можно использовать во вводном блоке, например, permalink, tags и category.

Если захотите создать новую статью на GitHub.com прямо в вашем браузере, просто перейдите в директорию /_posts/ и нажмите иконку «+». Главное — не забывайте придерживаться формата имени файлов и добавлять вводный блок, чтобы файлы обрабатывались Jekyll.



Создание новой статьи на сайте GitHub.com. (В высоком разрешении)

Проблема №6: Единственная проблема с Jekyll, с которой я столкнулся при создании блога — отсутствие CMS, так что я не мог просто залогиниться в CMS, чтобы сделать быстрые правки, находясь за чужим компьютером. Оказывается, блог на Jekyll будет иметь CMS, если использовать GitHub Pages, так как роль CMS исполняет сам GitHub. Можете редактировать статьи в браузере даже с телефона, если захотите. И, хотя это не так удобно как в других CMS, — это не помешает вам внести изменения даже если вы окажетесь далеко от своего компьютера.

Необязательные шаги

Использование своего доменного имени

Настройка доменного имени, что бы оно указывало на GitHub Pages — это простое действие, состоящее из двух шагов:

1. Создайте в корневой директории репозитория файл CNAME так, что бы он содержал нужное доменное имя (например, www.yourdomainname.com).

2. У регистратора доменного имени добавьте в настройках DNS запись CNAME, указывающую на GitHub Pages:

    - type: CNAME
    - host: www.yourdomainname.com
    - answer: yourusername.github.io

- TTL: `300`

Затем настойчиво обновляйте What's My DNS, пока не распространится информация о новой записи. Если возникнут проблемы, обратитесь к документации: «Настройка пользовательского доменного имени при работе с GitHub Pages».

Импорт статей из WordPress

Прежде чем импортировать статьи в блог, их надо сначала экспортировать из WordPress, возможно, немного адаптировав (например, обновив ссылки на изображения), и только затем импортировать их в ваш блог на Jekyll. К счастью есть несколько инструментов, которые могут в этом помочь.

Для **экспорта статей из WordPress**, я очень рекомендую WordPress to Jekyll Exporter Бена Балтера (Ben Balter), который позволяет сделать всё в один клик. Он экспортирует весь контент WordPress-блога, включая статьи, изображения и мета-данные, конвертирует, где необходимо, в подходящий для Jekyll формат и выдает в виде ZIP-архива. Спасибо тебе, Бен.

Ещё один вариант — экспортировать все содержимое WordPress через меню «Tools» панели администрирования, а затем импортировать используя Jekyll's importer.

Затем, нужно **обновить ссылки на изображения**. Плагин, написанный Беном Балтером, экспортирует все изображения в папку. Затем их нужно будет скопировать туда, где вы храните изображения для своего Jekyll-блога, это может быть папка `/images` или CDN.

После этого перед нами встанет задача обновить все ссылки на изображения в статьях. Так как я обновлял всего пять-шесть статей, мне отлично подошли быстрый поиск и замена, но если материалов много, тогда, возможно, стоит написать скрипт или подобрать уже готовый, например, скрипт Паула Стаматиуса (Paul Stamatiou).

И, наконец, нужно **импортировать комментарии**. Так как Jekyll — платформа для статических сайтов, он не поддерживает комментарии, однако, решения вроде Disqus отлично подходят для такого случая! Я рекомендую импортировать комментарии из WordPress в Disqus, затем, если вы используете Jekyll Now, можете ввести имя пользователя Disqus в `_config.yml` и все заработает.

Написание статей локально в любимом текстовом редакторе

Если вы предпочитаете писать статьи в Sublime, Vim, Atom или другом редакторе, всё, что нужно сделать — клонировать репозиторий, создать новый пост на Markdown в директории `_posts` и затем запушить изменения на GitHub. GitHub Pages автоматически пересоберёт сайт как

только файл с маркдауном попадет в репозиторий, и новая статья появится в блоге сразу по окончании сборки.

1. Сначала выполните команду `git clone git@github.com:yourusername/yourusername.github.io.git`, или клонируйте репозиторий используя GitHub Mac.
2. Создайте файл для новой статьи в папке `_posts`. Не забудьте назвать его в соответствии с форматом `year-month-day-title.md` и добавить в начало вводный блок.
3. Закоммитьте файл статьи и пушните в репозиторий. Может быть полезным посмотреть основы Git от компании Atlassian.
4. Вот и все! Подождите пока GitHub Pages пересоберет сайт. Обычно это занимает не больше 10 секунд, если у вас, конечно, не слишком много статей.

Проблема №7: Опять же, не нужно собирать сайт локально, чтобы написать и опубликовать статью. Можно просто локально написать статью и пушнуть её со всеми картинками в репозиторий, после чего GitHub Pages сам пересоберет сайт на сервере.

Создание темы для Jekyll

Хотите изменить тему? Вот кое-что, что вам нужно знать:

Сборка сайта локально

Если вы захотите создать довольно сложную тему, разумно проводить её разработку и тестирование локально. Это необязательно, можно просто пушить изменения в репозиторий и GitHub Pages всё соберёт, однако видеть изменения в процессе работы может быть полезно.

Сначала нужно установить Jekyll и его зависимости. Запустите `gem install jekyll`, затем `gem install jemoji jekyll-sitemap`. Должны быть установлены Ruby, RubyGems и Kramdown. Полный список зависимостей Jekyll.

Вот шаги для локального создания и просмотра сайта на Jekyll:

1. Сначала перейдите (`cd`) в директорию, в которой находится сайт.
2. Запустите `jekyll serve --watch`. (у Jekyll масса встроенных настроек.)
3. Откройте свой сайт по адресу `http://0.0.0.0:4000`.
4. Когда закончите, закоммитьте изменения и пушните все в ветку `master` соответствующего репозитория. GitHub Pages пересоберёт сайт.

Проблема №8: Имейте в виду, что `jekyll build` стирает все содержимое папки `/_sites/`. Первый шаг `jekyll build` — удаление всего, что есть в `/_sites/`, и сборка всех страниц с нуля. Так что не стоит хранить там файлы, если вы не хотите, чтобы они исчезли при следующей же сборке. В `/_sites/` должно быть только то, что генерирует Jekyll.

Структура директорий

Вот структура сайта на Jekyll:

```
/Users/barryclark/Code/jekyll-now
├─ CNAME # Содержит доменное имя (опционально)
├─ _config.yml # Файл конфигурации Jekyll
├─ _includes # Сниппеты кода, которые можно использовать в шаблонах
│  ├─ analytics.html
│  └─ disqus.html
├─ _layouts
│  ├─ default.html # Основной шаблон. Включает <head>, <navigation>, <footer>, и т.д.
│  ├─ page.html # Шаблон для статических страниц
│  └─ post.html # Шаблон для постов в блоге
├─ _posts # Все посты — тут!
│  └─ 2014-3-3-Hello-World.md
├─ _site # После сборки сайта Jekyll генерирует HTML в этой директории. Это то, что отдается браузе
│  ├─ CNAME
│  ├─ LICENSE
│  ├─ about.html
│  ├─ feed.xml
│  ├─ index.html
│  ├─ sitemap.xml
│  └─ style.css
├─ about.md # Статическая страница "О проекте", которую я создал.
├─ feed.xml # XML для работы RSS
├─ images # Содержит все картинки
│  ├── first-post.jpg
├─ index.html # Лэндинг
├─ scss # Sass со стилями сайта
│  ├─ _highlights.scss
│  ├─ _reset.scss
│  ├─ _variables.scss
│  └─ style.scss
└── sitemap.xml # Карта сайта
```

Шаблонизатор Liquid

Jekyll использует язык шаблонов Liquid. Про него нужно знать две важные вещи:

Во-первых, в начале каждого файла есть **вводный блок YAML**, он определяет шаблон для вывода страницы и такие переменные, как `title`, `date` и `tags`. Кроме того, он может содержать пользовательские переменные.

Теги шаблонизатора Liquid используются для создания циклов, условных операторов и для вывода материалов.

Например, пусть каждая статья в блоге использует шаблон из `/_layouts/post.html`:

```
---
layout: default
---

<article class="post">

  <h1>{{ page.title }}</h1>

  <div class="entry">
    {{ content }}
  </div>

  <div class="date">
    Написано {{ page.date | date: "%B %e, %Y" }}
  </div>

  <div class="comments">
    {% include disqus.html disqus_identifier=page.disqus_identifier %}
  </div>
</article>
```

Вначале файла располагается вводный блок YAML, окруженный тремя дефисами. В нем мы определяем, что файл должен быть обработан как содержимое шаблона `default.html`, который содержит шапку и подвал сайта.

Разметка Liquid использует двойные фигурные скобки для вывода данных. Самые первые Liquid-теги в примере выше `{{ page.title }}` и `{{ content }}` — они выводят заголовок и содержание статьи. В Jekyll доступно множество переменных для использования в шаблонах.

Одинарные фигурные и квадратные скобки используются для создания условных операторов, циклов и включения сниппетов кода. В этом примере я добавил в конец шаблона блок комментариев Disqus, который находится в файле `/_includes/disqus.html`.

_config.yml

Это файл конфигурации Jekyll, который содержит все настройки блога. Что хорошо в `_config.yml`, так это то, что в нем можно задавать пользовательские переменные, которые затем можно будет использовать в шаблонах сайта.

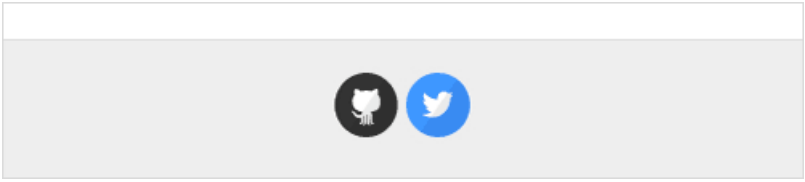
Например, я использую пользовательские переменные из `_config.yml` в блоге Jekyll Now, чтобы можно было удобно добавлять SVG-иконки в подвал сайта.

Вот переменные в `_config.yml`:

```
# Этот код добавляет иконку в подвал для каждого имени пользователя в списке
footer-links:
  github: barryclark
  twitter: baznyc
```

И вот как они используются в `/_layouts/default.html`:

```
<footer class="footer">
  {% if site.footer-links.github %}<a href="http://github.com/{{ site.footer-links.github }}">{% include svg
  {% if site.footer-links.twitter %}<a href="http://twitter.com/{{ site.footer-links.twitter }}">{% include svg
</footer>
<figure>
```



Пример SVG-иконок в подвале

Переменные в ссылку на Twitter добавляются следующим образом: `http://twitter.com/{{ site.footer-links.twitter }}`, так что ссылка в футере будет указывать на вашу учетную запись в Twitter. Ещё одна вещь, которая меня радует в переменных, — то, что их можно использовать для опционального отображения элементов интерфейса. Например, иконок в футере вообще не будет, если вы не зададите значение переменной.

Проблема №9: Обратите внимание, что изменения в `_config.yml` обновляются во время сборки, а не в реальном времени. Это означает, что если вы запускаете локально `jekyll serve` и редактируете `_config.yml`, — изменения не применятся. Нужно остановить и снова запустить `jekyll serve`.

Шаблоны и статические страницы

Большинству простых блогов нужно всего два шаблона: один для постов (`post.html`) и один для статических страниц (`page.html`). И единственная разница между ними в Jekyll Now состоит в том, что `post.html` включает блок комментариев Disqus и дату, а `page.html` — нет.

Если вы создадите файл с расширением `.html` или `.md` в корневой директории сайта, он будет рассматриваться как статическая страница. Например `about.md` будет доступна по адресу `www.mysite.com/about`. Легко и просто!

Изображения

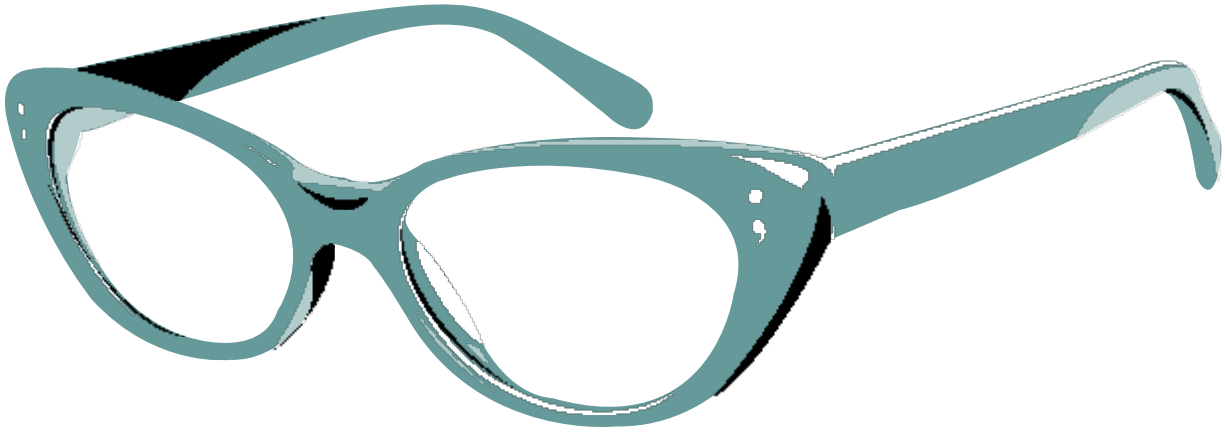
Я храню изображения в директории репозитория `/images/` и на данный момент не испытываю никаких проблем с производительностью. Если сайт размещен на GitHub Pages, изображения будут отдаваться с CDN GitHub'а и очень быстро загружаться. Я пока не вижу причин хранить их в другом месте, но, если уж мигрировать куда-нибудь вроде CloudFront, изменить ссылки совершенно не проблема.

Мне нравится простота хранения изображений в директории `/images/`, добавлять картинки в пост тоже очень просто, код в маркдаун выглядит вот так:

```
![Image description](/images/my-image.jpg)
```

Поддержка препроцессоров

Jekyll на данный момент поддерживает Sass и CoffeeScript без необходимости использовать плагины или Grunt. Можно просто добавить файлы с расширениями `.sass`, `.scss` и `.coffee` в рабочую директорию, и Jekyll их обработает, сгенерировав в той же директории `.css` и `.js`



Время Sass'ить! (Источник: Sass)

Чтобы быть уверенным, что `.sass`, `.scss` и `.coffee` будут обрабатываться, добавьте в начало файлов две строки с тройным дефисом:

```
---
---
$color-coffee: #644C37;
```

Если вы используете `@imports` для разбиения Sass на модули, надо сообщить об этом Jekyll, добавив в `_config.yml` следующее:

```
sass:
  sass_dir: _scss
```

Кроме того, можно задать стиль вывода скомпилированного CSS:

```
sass:
  sass_dir: _scss
  style: :compressed
```

Расширенные возможности Jekyll

В Jekyll есть несколько мощных, более продвинутых фич, которые могут пригодиться, если вы захотите создать что-либо сложнее простого блога.

Файлы данных

Есть два способа интегрировать внешние данные в Jekyll. Первый — используя сторонние сервисы и API. Например, Disqus позволяет добавлять динамический контент на статический сайт используя внешние сервисы.

Второй способ — использование файлов данных. Jekyll может читать файлы в формате YAML и JSON из директории `/_data/` и позволяет использовать их в шаблонах как обычные переменные. Это весьма полезно для хранения переиспользуемых данных или настроек, которые вы не хотите помещать в `_config.yml`, чтобы он не разрастался сверх меры.

Кроме того, data-файлы дают возможность добавлять на сайт большие наборы данных. Можно написать скрипт, который будет разбивать их на несколько JSON-файлов и размещать их в директории `/_data/`. Ярким примером такого подхода является использование данных Google Analytics в Jekyll для ранжирования постов по популярности.

Коллекции

Два стандартных типа документов Jekyll — это посты (статьи для блога) и страницы (просто статический контент). Релиз Jekyll 2.0 добавил коллекции, которые позволяют создавать собственные типы документов. Например, можно использовать коллекции для создания фотоальбома, книги или портфолио.

Заключение

Jekyll подходит не для каждого проекта. Основным недостатком генератора статики является сложность внедрения динамических

функций на стороне сервера. Количество сервисов, которые можно интегрировать в проект, таких как Disqus, растет, но не все из них предоставляют гибкие настройки и возможность контроля. Jekyll не подходит для создания сайтов с базами пользователей, так как в нём нет ни баз данных, ни логики на стороне сервера для обработки регистрации или аутентификации.

Достоинства Jekyll — его простота и минимализм. Jekyll дает вам всё необходимое, чтобы создать сайт, ориентированный на контент, и который не подразумевает особой интерактивности. Это делает Jekyll идеальным для блога и портфолио и позволяет использовать его для реализации простых сайтов для клиентов.

Не дайте репутации Jekyll, как платформы для создания блогов для хакеров, отпугнуть вас. Создание красивых, быстрых, минималистичных сайтов с его помощью не требует элитных хакерских навыков или умения работать с командной строкой. Как я уже показал в пошаговом руководстве выше, его можно настроить за считанные минуты, посвятив остальное время работе над контентом и дизайном.

Ресурсы для дальнейшего изучения

- Jekyll Официальный сайт — это лучший ресурс для изучения Jekyll и создания сайта.
  Значительная часть информации для статьи была почёрпнута оттуда.

- Jekyll Now Этот ресурс упрощает создание блога на основе Jekyll, избавляя от необходимости разбираться вот множестве начальных настроек.

- Исходный код Jekyll Репозиторий содержит исходный код и дискуссии на тему Jekyll.

Оригинальная статья: Build A Blog With Jekyll And GitHub Pages

Статью вычитывали: Vladimir Starkov, iamstarkov, SilentImp, yoksel, FMRobot

Barry Clark
http://www.barryclark.co/
Twitter: https://twitter.com/barrycnyc
GitHub: https://github.com/barryclark

Антон Немцев
http://frontender.info/
Twitter: https://twitter.com/silentimp
GitHub: https://github.com/SilentImp