20 May 2013

[Jekyllいつやるの？ジキやルの？今でしょ！](http://melborne.github.io/2013/05/20/now-the-time-to-start-jekyll)

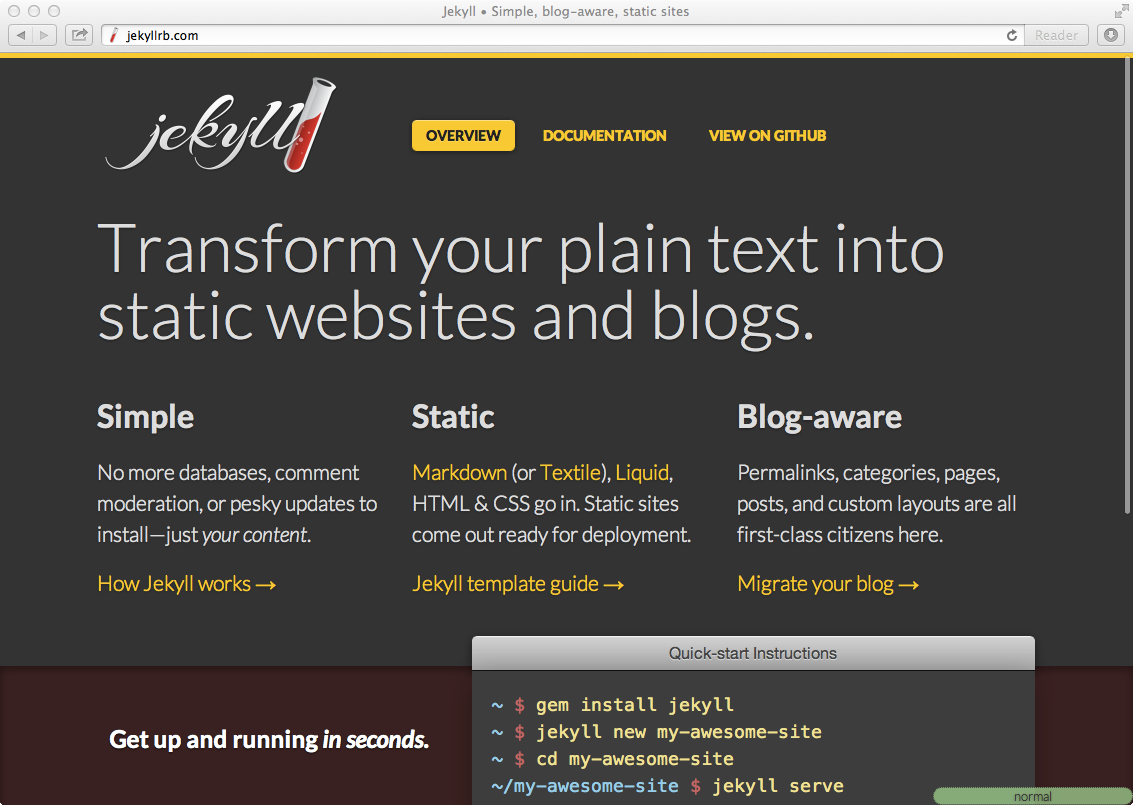
す、すいません… ベタなタイトルで…。

Jekyll Version 1.0

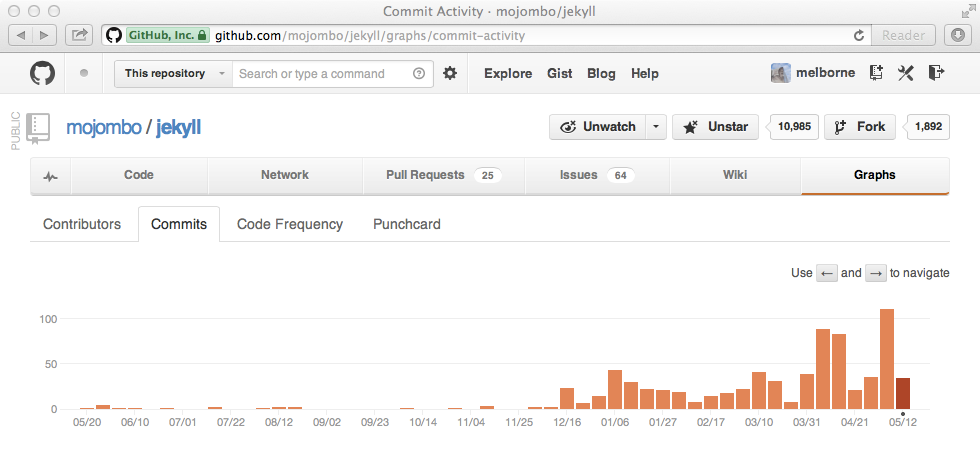
2013年5月5日にJekyllのVersion 1.0がリリースされました。

[Jekyll • Simple, blog-aware, static sites](http://jekyllrb.com/)

[Jekyll 1.0 Released - Musings of Parker J. Moore](http://blog.parkermoore.de/2013/05/06/jekyll-1-dot-0-released/)



2011年12月のVersion0.11.2以降、Jekyllの開発は停滞気味でした。しかし、[parkr (Parker Moore)](https://github.com/parkr" \o "parkr (Parker Moore))氏がJekyllのcontributorとして参加したことを機に、昨年12月15日頃から新たな胎動が始まり、同22日に大きな問題を解決したVersion0.12.0がリリースされました。その後も怒涛の勢いでコミットが投げられ、遂に、5月5日にVersion 1.0がリリースされたのでした。Githubにおける[Commit Activity](https://github.com/mojombo/jekyll/graphs/commit-activity" \o "Commit Activity · mojombo/jekyll)グラフを見るとその変化がよくわかると思います。



「オープンソースは生きてる（生き物である）」ってことを実感させられる出来事です。

新機能

上記の活動を経て、新Jekyllには次のような比較的大きい新たな機能・改善が導入されました（0.12.0からの機能も含む）。

サブコマンド方式が採用され、new, build, serve, importなどのサブコマンドが導入された。

jekyll newコマンドにより、簡単にブログのひな形が作れるようになった。

下書き記事を書きやすい環境ができた（Drafts）。

記事の引用機能が導入された（Excerpt）。

ページネーションができるようになった（Paginator）。

コードのシンタックスハイライトおよびgistタグが組み込みとなった。

公式サイトが大幅リニューアルされ、ドキュメントが充実した。

Pygments.rbの採用によりビルドが高速化された（0.12.0より）。

他ブログからの移行を支援するmigrate機能をjekyll-migrateというサブgemに集約した。

このブログもJekyll（+Github Pages）で構築されていますが、個人的に特にうれしいのは公式サイトにおけるドキュメントの充実っぷりと、Pygments.rbの採用です。ドキュメントは分かりやすく整理され、かつ記述も丁寧で素晴らしいです。現状英語のみですが一読をお薦めします。

[Jekyll Docs](http://jekyllrb.com/docs/home/)

また、後者についてですが、Jekyllではビルドの度に全ての記事をコンパイルするので、このブログのようにコードを大量に含むブログのビルドには、そのシンタックスハイライト処理のために悲しいくらいの時間が掛かります。これがPygments.rbの採用により大幅に改善されました。つまりこのサイトでは９分（！）が１分になりました[1](http://melborne.github.io/2013/05/20/now-the-time-to-start-jekyll/" \l "fn:1)。詳細は以下の記事が参考になります。

[Jekyll の 0.12.0 でシンタックス ハイライトが便利になった - てっく煮ブログ](http://tech.nitoyon.com/ja/blog/2012/12/25/jekyll-0-12-0/)

Jekyllをいつやるか

Version 1.0によりJekyllの完成度は格段に高まりました。jekyll newで簡単にブログのひな形ができちゃうし、移行ツールも充実してるし、Githubの無料ホスティング[GitHub Pages](http://pages.github.com/" \o "GitHub Pages)もあるし、プログラマが自分に適した快適なブログ環境を構築するのに、もはや何の障壁もありません。

では、Jekyllをいつやるのか？

今やる

そのうちやる（ジキやル）

やる予定なし

環境が整ったって言ったって、ドキュメントは英語だし、簡単なチュートリアルくらい無いとそう簡単にはねぇ…。

日本語の電子書籍でもあればやるキッカケにもなるんだけどねぇ…。

ちなみに、Rubyの公式サイトも今やJekyllを使って構築されているんですね…。

[オブジェクト指向スクリプト言語 Ruby](http://www.ruby-lang.org/ja/) このサイトはRubyとJekyllによって構築されています。 また、コンテンツはRubyコミュニティによって管理されています。 このサイトに関する質問やコメントはGitHubへのissueまたはpull request、もしくはwebmaster までお願いします。

…

そんなわけで…

^ ^;

30分のチュートリアルでJekyllを理解する

１年前に自分はこのブログでJekyllのチュートリアルの記事を書きました。

[30分のチュートリアルでJekyllを理解する](http://melborne.github.io/2012/05/13/first-step-of-jekyll/)

この度、これを加筆・訂正しJekyllの最新バージョン（Version 1.0.2）に対応させましたので、ここに公開致します。

前記事からの追加変更点は次のとおりです。

Jekyll Version 1.0.2に対応するよう記述を修正。

Jekyllの特徴について追記。

Jekyll newコマンドについて追記。

Excerpt（記事引用機能）について追記。

categoriesについて追記。

gistタグについて追記。

下書き（Drafts）機能について追記。

ページネーション、他のブログからの移管について追記。

その他細部の修正。

加えて前回同様、本記事を電子書籍化しました。



（目次）

１章　Jekyllとは

２章　Jekyllを知る

３章　記事の作成

４章　デフォルト設定の変更

５章　CSSによるスタイル設定

６章　グローバル変数「site」

７章　Liquid制御構文

８章　グローバル変数「page」

９章　Liquidによるシンタックスハイライト

１０章　プラグイン

１１章　その他の機能

１２章　まとめ

電子書籍は、いつものようにGumroadを通して販売し、EPUB形式に加えkindleで扱えるMOBI形式を同梱しています。価格は105円です[2](http://melborne.github.io/2013/05/20/now-the-time-to-start-jekyll/" \l "fn:2)。

興味のある方、無くても暇な方、投げ銭したい方、購入ご検討いただければ大変にうれしく思いますm(\_\_)m

[M’ELBORNE BOOKS](http://melborne.github.io/books/)

１章　Jekyllとは

[Jekyll](http://jekyllrb.com/)は、ブログのような静的サイトを構築するためのRuby製ファイルジェネレータです。Jekyllの特徴としては、以下の様な点が挙げられます。

HTML簡略記法としてMarkdownまたはTextile記法が使える。

テンプレート言語Liquidを使って、シンタックスハイライト、gist、その他のタグの埋め込みによる拡張が簡単にできる。

Githubが提供するホスティングサービス「GitHub Pages」のエンジンとして利用されているので、Jekyllで構築されたサイトを無料で簡単に公開できると共に、gitによりそのソースおよび記事データの一元管理ができる。

Jekyllは静的サイトを構築するため軽いサイトが構築しやすい。

HTML整形後の文章をローカル環境で確認することができる。

一方で、Jekyllはあくまでファイルジェネレータであり、[Rails](http://rubyonrails.org/)のように、コマンド一つでサイトの基礎を全部構築してくれるフレームワークではありません。むしろ[Sinatra](http://www.sinatrarb.com/" \o "Sinatra)の思想に近く、静的サイト版Sinatraといった趣のツールです。従って、Jekyllを使ってブログサイトを構築しようとする場合、その規約に従って、自分で１つずつファイルを用意していく必要があります。

しかしながら、Sinatraに[Padrino](http://jp.padrinorb.com/" \o "Padrino で素敵なウェブ開発を - Padrino Ruby Web Framework)があるように、Jekyllにもそれをフレームワーク化するツールがあります。[Octopress](http://octopress.org/" \o "Octopress)と[Jekyll-Bootstrap](http://jekyllbootstrap.com/" \o "Jekyll-Bootstrap)です。

Octopress

[GitHub Pages](http://pages.github.com/)その他のホスティングにおいて、最小の労力でブログサイトを構築したいなら、Octopressを選ぶべきです。僅か数ステップで設定が完了し記事を書き始めることができるようになります。各種のPluginも簡単に利用できます。Octopressでブログを構築するのなら、次のサイトが参考になります。

[Big Sky :: githubとjekyllとoctopressで作る簡単でモダンなブログ](http://mattn.kaoriya.net/software/lang/ruby/20111017205717.htm)

[Octopressのインストールから運用管理まで - T.I.D.](http://tokkonopapa.github.com/blog/2011/12/30/octopress-on-github-and-bitbucket/)

Jekyll-Bootstrap

Octopressは簡単な一方で、少しお仕着せのところがあり自由度が少なく感じます。自分の好みにあわせてデザインを変更したり、他の人の作ったデザインを適用したりしたい人は、Jekyll-Bootstrapをより気に入ると思います。Octopressよりもデザインテーマのモジュール化が進んでいて、簡単にその切り替えができるように作られています。現状ではまだテーマが少ないですが（http://themes.jekyllbootstrap.com/を参照のこと）、他の人のテーマを参考に、そのいいとこ取りをして自分のテーマを用意することもできます。本家サイトの解説が非常に充実していますので、その使い方は概ねこれで足ります。日本語の解説は次のサイトが参考になります。

[ruby と jekyll と jekyll-bootstrap で静的サイトを作る - KRAKENBEAL RECORDS](http://krakenbeal.blogspot.jp/2012/05/ruby-jekyll-jekyll-bootstrap.html)

jekyll newコマンド

Jekyll version 1.0よりnewというサブコマンドが導入されました。これはコマンド一つでブログサイトのひな形を構築するいわゆるscaffoldingを実現します。生成されるひな形にはコメントシステムやソーシャルサービスとの連携機能などはないものの、ブログの基本的要素は既に揃っています。ブログを運用しつつ全体を理解しながら少しづつ独自の改良を加えていく、といった場合には上記フレームワークに頼らないこの方法が最適でしょう。newの使い方については第１１章で触れます。

２章　Jekyllを知る

これらのフレームワークは大変便利でJekyllに対する理解がそこそこでも、ブログの運営上大きな支障はありません。しかし、その一方でそのレールから少し外れたことをしようとすると、途端に立ちいかなくなるという問題があります。よくある話です。

本書の目的は、チュートリアルを通してJekyllの概要を理解することです。Jekyllに対する理解が深まれば、上記フレームワークを使った場合でもその運用が楽になるはずです。

チュートリアルでは上記フレームワークを使わずに、Jekyllだけでブログの基礎となるファイル群を１つずつ構築していきます。ここではUnix系OSの使用を前提にしていますので、他OSの場合は適宜読み替えて下さい。Jekyllの対応バージョンは1.0.2です。

トップページ

gem install jekyllでJekyllを入手したら、ブログを作るディレクトリを用意します。

% mkdir jk

jkディレクトリに移動してトップページを作ります（jkはJekyllの略です）。JekyllではMarkdownが使えるので、index.mdを作ります。Markdownの文法については以下を参考にしてください。

[blog::2310 » Markdown文法の全訳](http://blog.2310.net/archives/6)

% cd jk

% echo #Welcome to my JK Home Page! > index.md

そしてjekyll buildコマンドを実行します。

% jekyll build

（version1.0でサブコマンドが導入されました。）

treeします。

% tree

.

├── \_site

│   └── index.md

└── index.md

1 directory, 2 files

\_siteというディレクトリ以下にindex.mdがそのままコピーされてしまいました。つまりhtmlに変換されていません。ルートのindex.mdを直しましょう。先頭に---を２本入れます。

---

---

#Welcome to my JK Home Page!

もう一度jekyll buildしてtreeします。

% jekyll build

% tree

.

├── \_site

│   └── index.html

└── index.md

1 directory, 2 files

今度はうまくいきました。Jekyllではこの先頭の---で挟まれた領域をYAML Front-Matter(YAML前付け)といいます。ここにそのドキュメントの書誌事項を書きます。JekyllはYAML Front-Matter付きのドキュメントをその変換の対象とし、それ以外のドキュメントはそのまま\_siteディレクトリ以下にコピーするのです。

では結果をブラウザで確認します。次のようにします。

% jekyll serve

Configuration file: none

Source: /Users/keyes/Google Drive/playground/jk

Destination: /Users/keyes/Google Drive/playground/jk/\_site

Generating... done.

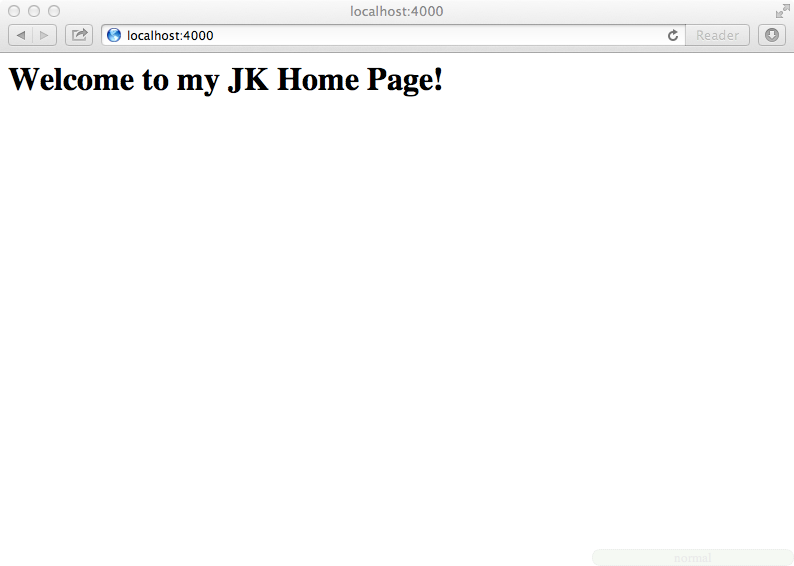
[2013-05-16 21:48:12] INFO WEBrick 1.3.1

[2013-05-16 21:48:12] INFO ruby 2.0.0 (2013-02-24) [x86\_64-darwin12.2.0]

[2013-05-16 21:48:12] INFO WEBrick::HTTPServer#start: pid=11794 port=4000

（version1.0で従来のjekyll --serverからserveまたはserverサブコマンドを利用するようになりました。）

4000番ポートでWEBrickが立ち上がりました。http://localhost:4000を開きます。



うまくいきました。

レイアウト

さて、ブラウザでview sourceなどして、ちょっとこのソースを確認してみます。

<h1 id='welcome\_to\_my\_jk\_home\_page'>Welcome to my JK Home Page!</h1>

当然ながらHTMLによる構造化レイアウトができていません。早速用意します。

まずはルートに\_layoutsディレクトリを作ります。layoutファイルをdefault.htmlとしてここに配置します。

% mkdir \_layouts

% touch \_layouts/default.html

default.htmlをエディタで開いて、次のような内容にします。

<!DOCTYPE html>

<head>

<meta http-equiv="Content-type" content="text/html; charset=utf-8">

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

</head>

<body>

{{ content }}

<p>- rendered with layout template -</p>

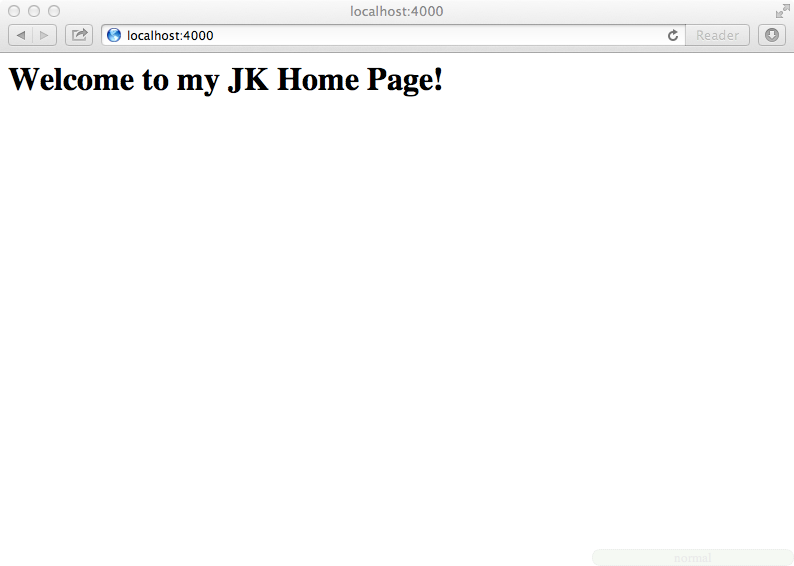
</body>

html内の{{ }}はLiquidテンプレートによる埋め込み指定です。Jekyllでは[Liquid](https://github.com/Shopify/liquid" \o "Shopify/liquid)というテンプレートエンジンが使えます。titleではpage.titleにより対象ページのタイトルを、bodyではcontentにより対象ページのコンテンツを埋め込むように指定しています。ここではこのlayoutが読み込まれたことを分かるようにするために、rendered with layout templateという文字を入れました。

再度サーバーを起動して変化を見ます。ここでは--watchオプションを付けて起動してみます。

% jekyll serve --watch

（version1.0では従来のjekyll --autoからserveサブコマンドに–watchオプションを渡すようになりました。）

ブラウザで確認します。 

残念ながら変化がありません。

これはindex.mdでどのテンプレートを使うかをちゃんと指定していないからです。ここでYAML Front-Matterの出番です。index.mdを次のように直します。

---

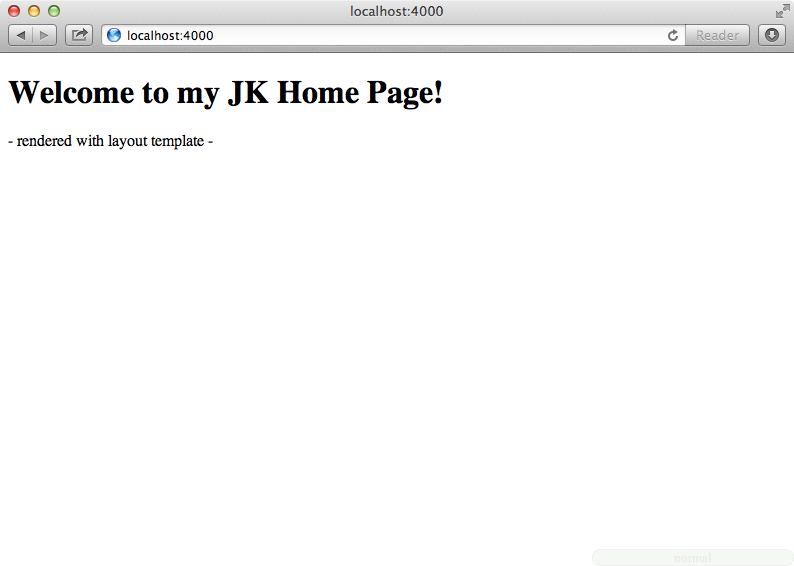
layout: default

title: Oh! My JK

---

#Welcome to my JK Home Page!

ここでは同時にタイトルもセットしました。サーバーはそのままにブラウザをリロードします。



今度はうまくいきました。--watchオブション指定により、変更が直ちに反映されました。ソースも見てみます。

<!DOCTYPE html>

<head>

<meta http-equiv="Content-type" content="text/html; charset=utf-8">

<title>Oh! My JK</title>

</head>

<body>

<h1 id='welcome\_to\_my\_jk\_home\_page'>Welcome to my JK Home Page!</h1>

<p>- rendered with layout template -</p>

</body>

いいですね。

３章　記事の作成

さてトップページができたので、記事を書きます。Jekyllでは記事のことをPostと呼びます。ルートに\_postsディレクトリを作成し、次のようなファイル名で記事ファイルを作ります。サーバーはそのままに別のターミナルを立ち上げて、以下を実行します。

% mkdir \_posts

% touch \_posts/2013-05-16-Jekyll-Tutorial.md

エディタで次のようにMarkdownで記事を書きます。Front-Matterを忘れずに。

---

---

##Jekyll Tutorial

Jekyll is a simple, blog aware, static site generator.

jekyll buildコマンドを実行してからtreeします。

% jekyll build

% tree

.

├── \_layouts

│   └── default.html

├── \_posts

│   └── 2013-05-16-Jekyll-Tutorial.md

├── \_site

│   ├── 2013

│   │   └── 05

│   │   └── 16

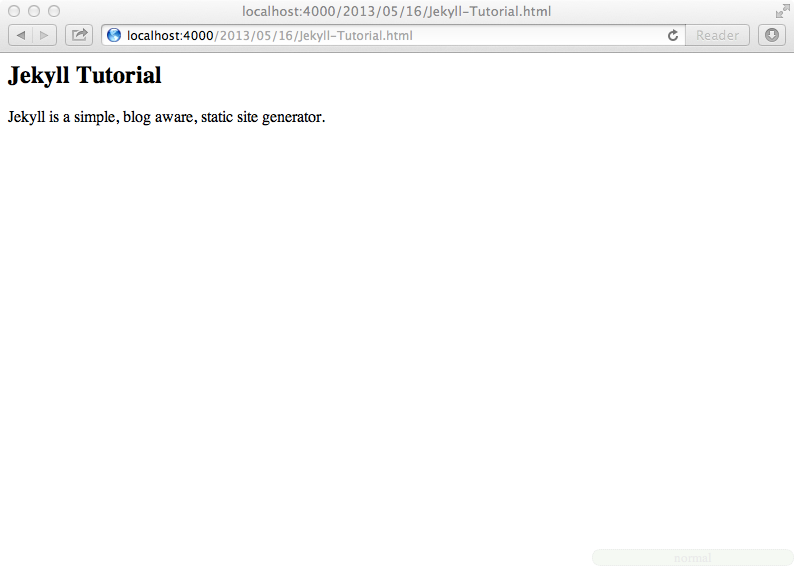
│   │   └── Jekyll-Tutorial.html

│   └── index.html

└── index.md

6 directories, 5 files

\_siteディレクトリ以下に/2013/05/16/Jekyll-Tutorial.htmlというファイルが生成されたのが分かります。つまり\_post以下に生成したファイルのファイル名のうち、その日付部分はファイルパスとして展開されるのです。ブラウザでhttp://localhost:4000/2013/05/16/Jekyll-Tutorial.htmlを開いてみましょう。



うまくいってるようです。

Postのレイアウト

次に記事にも専用のレイアウトを用意します。\_layoutsディレクトリ以下にpost.htmlを作って、次のような内容にします。

% touch \_layouts/post.html

---

layout: default

---

<header>

<div>{{ page.date | date\_to\_string }}</div>

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

</header>

<div class='main'>

{{ content }}

</div>

<footer>

<p>- rendered with post template -</p>

</footer>

Front-Matterでdefault.htmlを読み込むようにします。これによりpost.htmlはdefault.htmlのサブテンプレートになります。header部分に投稿日とタイトルを表示するようLiquidで指定します。

なお、{{ page.date | date\_to\_string }}はJekyllで拡張されたLiquidのフィルターという機能を使って、出力の整形を行なっています([Templates](http://jekyllrb.com/docs/templates/))。default.htmlと同様に、このテンプレートが読み込まれたことを見るためにrendered with post templateを追加しておきます。

そして記事側(/2013-05-16-Jekyll-Tutorial.md)でこのテンプレートを読めるように、そのFront-Matterを記述します。

---

layout: post

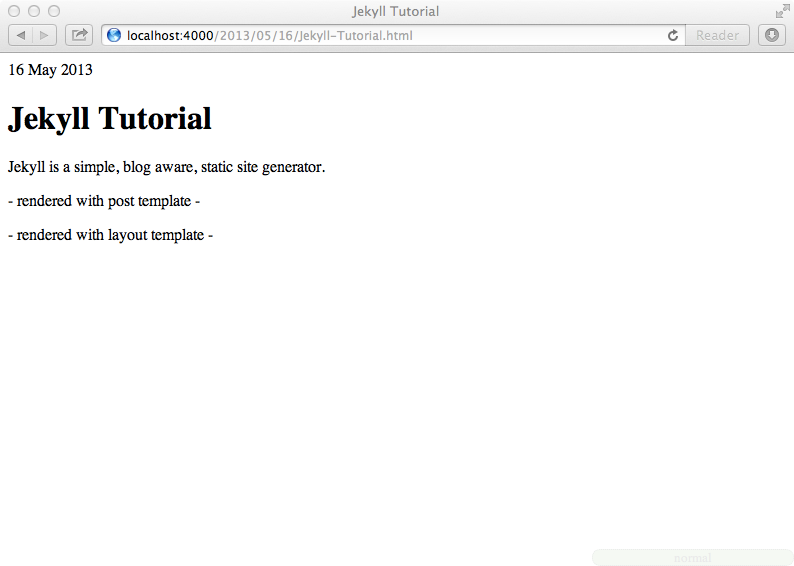
---

Jekyll is a simple, blog aware, static site generator.

タイトルはテンプレート側で用意するようにしたので、ここでは消します。

サーバーを再起動して変化を見ます。

% jekyll serve --watch



ソースも見ます。

<!DOCTYPE html>

<head>

<meta http-equiv="Content-type" content="text/html; charset=utf-8">

<title>Jekyll Tutorial</title>

</head>

<body>

<header>

<div>16 May 2013</div>

<h1>Jekyll Tutorial</h1>

</header>

<div class='main'>

<p>Jekyll is a simple, blog aware, static site generator.</p>

</div>

<footer>

<p>- rendered with post template -</p>

</footer>

<p>- rendered with layout template -</p>

</body>

うまくテンプレートが読み込まれました。

４章　デフォルト設定の変更

前のソースにおいてはファイル名からJekyll Tutorialというタイトルが自動的にセットされていることが分かります。Front-Matterを使って、タイトルとついでに投稿日も変えてみましょう。\_posts/2013-05-16-Jekyll-Tutorial.mdを修正します。

---

layout: post

title: Jekyllのチュートリアル

date: 2013-05-17

---

Jekyll is a simple, blog aware, static site generator.

日付を変えたのでhttp://localhost:4000/2013/05/17/Jekyll-Tutorial.htmlにアクセスします。



投稿日とタイトルが変わりました。つまりYAML Front-Matterの記述によってデフォルトの設定が上書きされました。この場合、元のMarkdownファイルのファイル名における日付と、パスの日付が一致しなくなる点注意が必要です。

\_config.ymlによる設定

個別ページの設定はそのFront-Matterで行うということが分かりました。では全体的な設定はどこでするのでしょうか。それはルートに\_config.ymlというファイルを用意して、そこで行います。試しに記事のパス形式を変えてみます。ここではサーバーの再起動が必要です。

% echo permalink: /:categories/:year-:month-:day/:title > \_config.yml

% jekyll build

% tree

.

├── \_config.yml

├── \_layouts

│   ├── default.html

│   └── post.html

├── \_posts

│   └── 2013-05-16-Jekyll-Tutorial.md

├── \_site

│   ├── 2013-05-17

│   │   └── Jekyll-Tutorial

│   │   └── index.html

│   └── index.html

└── index.md

5 directories, 7 files

\_siteディレクトリを見てわかるように、記事のパスが変わりました。

更に、記事にカテゴリーを付けてみます。ここでは\_posts/2013-05-16-Jekyll-Tutorial.mdをblog,technologyカテゴリーに属するよう設定します。記事ファイルを開いてそのFront-matterにcategories設定を追加します。

---

layout: post

title: Jekyllのチュートリアル

date: 2013-05-17

categories: blog technology <-- ここを追加

---

Jekyll is a simple, blog aware, static site generator.

再度ビルドします。

% jekyll build

% tree

.

├── \_config.yml

├── \_layouts

│   ├── default.html

│   └── post.html

├── \_posts

│   └── 2013-05-16-Jekyll-Tutorial.md

├── \_site

│   ├── blog

│   │   └── technology

│   │   └── 2013-05-17

│   │   └── Jekyll-Tutorial

│   │   └── index.html

│   └── index.html

└── index.md

5 directories, 7 files

\_siteディレクトリを見ると、記事が/blog/technology/パス以下に配置されているのが確認できます。先に進む前にcategories設定は削除しておいて下さい。

５章　CSSによるスタイル設定

さて、そろそろCSSによるスタイル設定をします。説明の最初でYAML Front-Matterを含まないファイルはそのまま\_siteディレクトリにコピーされると説明しましたが、CSSファイルはそのようにして配置されるファイルの一つです。ここではassets/cssというディレクトリをルートに作って、単純なCSS定義をしたstyle.cssというファイルを用意します。

% mkdir -p assets/css

% touch assets/css/style.css

body {

background-color: #dee;

}

h1 {

color: #11d;

}

h2 {

color: #161;

}

header {

border-bottom: 10px solid #a33;

}

footer {

background-color: #a33;

color: #fff;

}

\_layouts/default.htmlでstyle.cssを読み込むようにします。

<!DOCTYPE html>

<head>

<meta http-equiv="Content-type" content="text/html; charset=utf-8">

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

<link rel="stylesheet" href="/assets/css/style.css" type="text/css"> //ここを追加

</head>

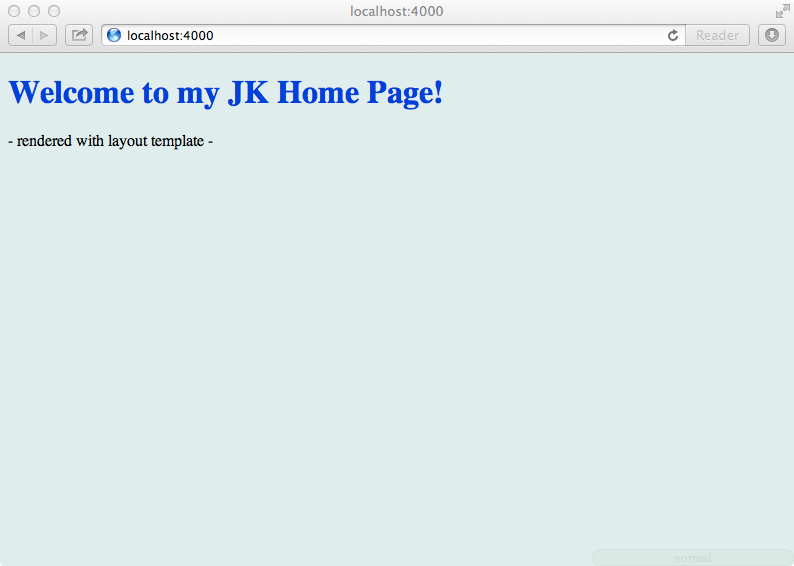
<body>

{{ content }}

<p>- rendered with layout template -</p>

</body>

ブラウザで見てみます。まずはトップページ（http://localhost:4000/）にアクセスします。



次に、記事のページ（http://localhost:4000/2013-05-17/Jekyll-Tutorial/）にもアクセスします。



うまく適用されているようです。

６章　グローバル変数「site」

Jekyllでは生成されたサイトに対する情報は、グローバル変数siteを通してアクセスできます。siteがどんな情報を持っているか、ちょっと見てみます。index.mdに戻って、次の一行を追加します。

---

layout: default

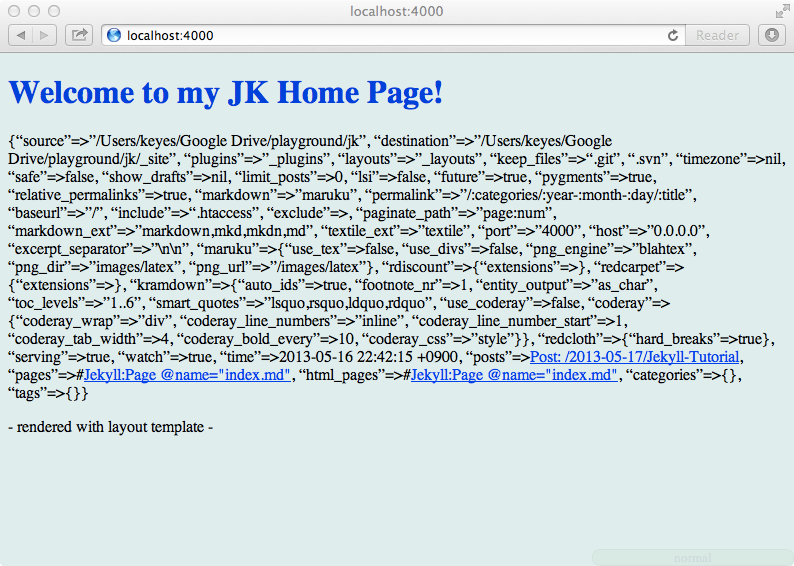
title: Oh! My JK

---

#Welcome to my JK Home Page!

{{ site }} <!-- ここを追加 -->

ブラウザで見てみます。



siteで読みだし可能な設定情報が表示されました。情報はHash形式で記録されていますが、これら情報へのアクセスはメソッド呼び出しのようにドットチェーンで行えます。幾つか見てみます。index.mdを書き換えます。

---

layout: default

title: Oh! My JK

---

#Welcome to my JK Home Page!

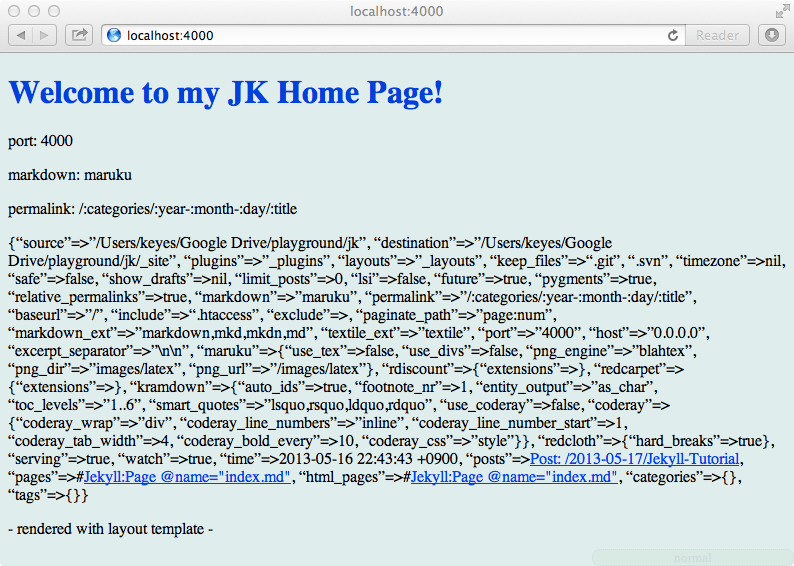
port: {{ site.port }}

markdown: {{ site.markdown }}

permalink: {{ site.permalink }}

{{ site }}

アクセスします。



.(ドット)でアクセスできました。このうちpermalinkは\_config.ymlでセットしたデータでした。\_config.ymlに別の設定を追加して読み出せるか試してみます。

permalink: /:categories/:year-:month-:day/:title

author:

name: Charlie

email: charlie@gmail.com

markdown: rdiscount

Markdownにrdiscountを使うように設定したので、gem install rdiscountする必要があるかも知れません。

index.mdにauthorに関する事項を追記します。

---

layout: default

title: Oh! My JK

---

#Welcome to my JK Home Page!

port: {{ site.port }}

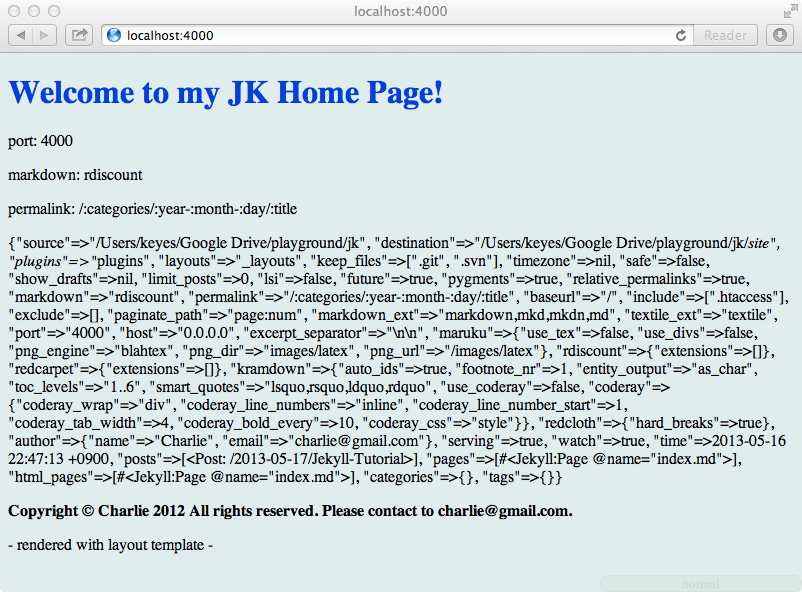
markdown: {{ site.markdown }}

permalink: {{ site.permalink }}

{{ site }}

\*\*Copyright © {{ site.author.name }} 2013 All rights reserved. Please contact to {{ site.author.email }}.\*\*

サーバーを再起動してからアクセスします。



Copyrightの注意書きでnameとemailのメソッドチェーンが出来ています。

７章　Liquid制御構文

変数siteからはpostsを介してPostに関する情報にもアクセスできます。Liquidテンプレートではif, case, forなどの制御構文が使えるので、これを使って各Postの内容を読みだしてみます。

その前に記事をもう一つ増やしておきます。

% touch \_posts/2013-05-18-Ruby-is-Great.md

エディタで記事を書きます。

---

layout: post

title: Rubyは最高

tags: [ruby, language]

---

##Rubyとは

Rubyは、手軽なオブジェクト指向プログラミングを実現するための種々の機能を持つオブジェクト指向スクリプト言語です。

そしてindex.mdを次のようにします。

---

layout: default

title: Oh! My JK

---

#Welcome to my JK Home Page!

<!-- 以下を追加 -->

<ul>

{% for post in site.posts %}

<li>

<a href="{{ post.url }}">{{ post.date | date\_to\_long\_string }} : {{ post.title }}</a>

</li>

{% endfor %}

</ul>

<!-- -->

port: {{ site.port }}

markdown: {{ site.markdown }}

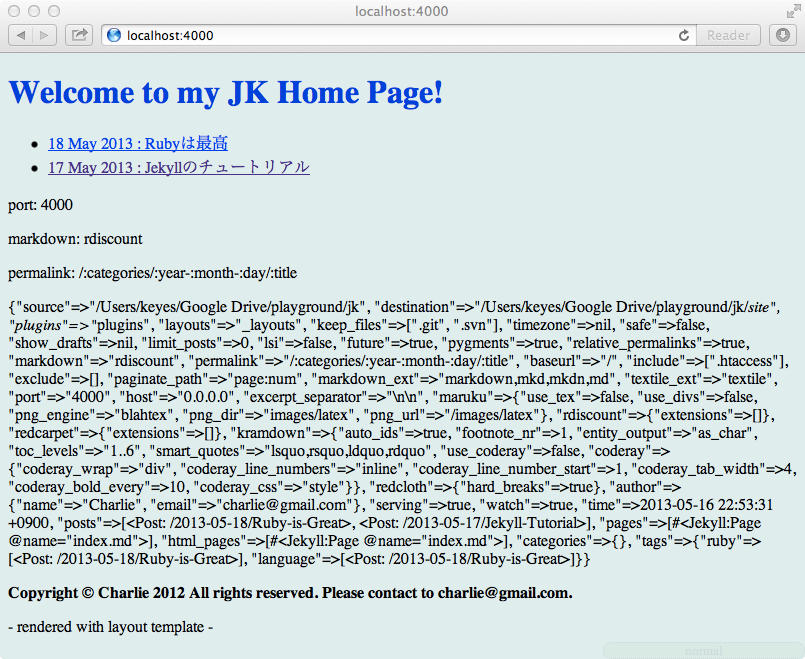
permalink: {{ site.permalink }}

{{ site }}

\*\*Copyright © {{ site.author.name }} 2013 All rights reserved. Please contact to {{ site.author.email }}.\*\*

for post in site.postsでpostオブジェクトをイテレートして、url, date, titleの各要素を読み出し、リンクを生成します。

ブラウザで見てみます。



クリックしてみます。



いいですね。

記事の引用

一歩進んで、version1.0で導入されたexcerptという機能を使って、記事の引用を見出しとともに表示させてみましょう。

index.mdを修正します。

---

layout: default

title: Oh! My JK

---

#Welcome to my JK Home Page!

<ul>

{% for post in site.posts %}

<li>

<a href="{{ post.url }}">{{ post.date | date\_to\_long\_string }} : {{ post.title }}</a>

<p>{{ post.excerpt }}</p> <!-- ここを追加 -->

</li>

{% endfor %}

</ul>

port: {{ site.port }}

markdown: {{ site.markdown }}

permalink: {{ site.permalink }}

{{ site }}

\*\*Copyright © {{ site.author.name }} 2013 All rights reserved. Please contact to {{ site.author.email }}.\*\*

ブラウザで見てみます。



自動で記事の最初の行が引用として読み出されました。

Rubyの記事がタイトルが含まれた見づらいものになってしまったので、手動で設定してみます。記事のFront-matterにexcerpt属性をセットします。

---

layout: post

title: Rubyは最高

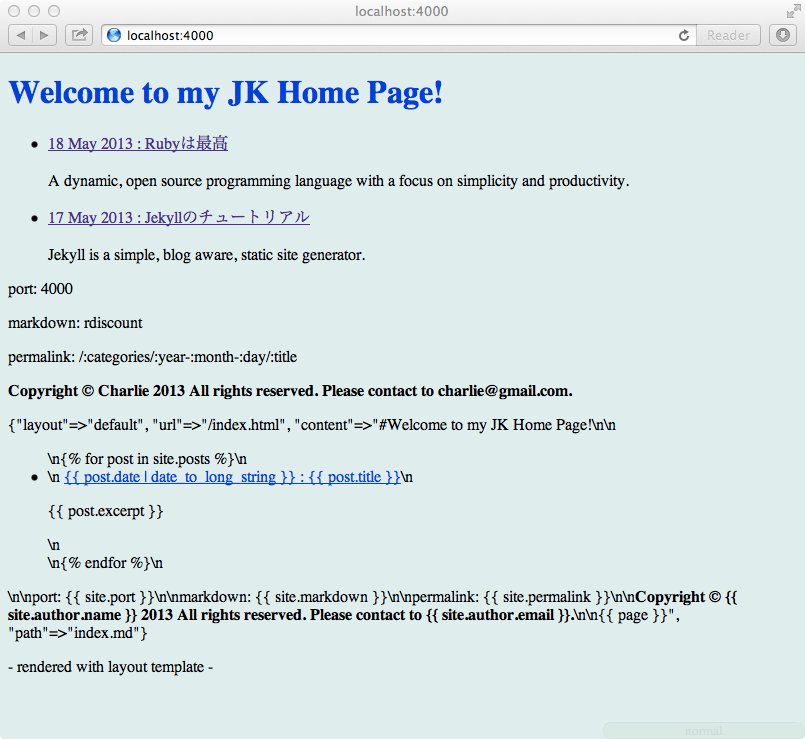
tags: [ruby, language]

excerpt: A dynamic, open source programming language with a focus on simplicity and productivity.

---

##Rubyとは

Rubyは、手軽なオブジェクト指向プログラミングを実現するための種々の機能を持つオブジェクト指向スクリプト言語です。



手動設定した引用が表示されました。

８章　グローバル変数「page」

変数siteはサイト全体の情報を持ったオブジェクトでした。一方で、個々のページの情報はグローバル変数pageが持っています。同様にindex.mdで読みだしてみます。

---

layout: default

title: Oh! My JK

---

#Welcome to my JK Home Page!

port: {{ site.port }}

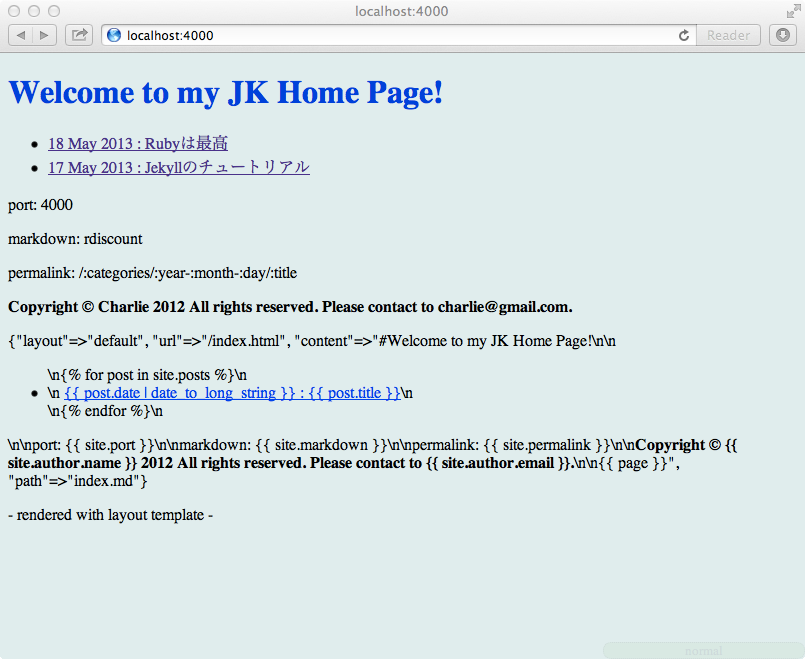
markdown: {{ site.markdown }}

permalink: {{ site.permalink }}

<!-- {{ site }} -->

\*\*Copyright © {{ site.author.name }} 2013 All rights reserved. Please contact to {{ site.author.email }}.\*\*

{{ page }} <!-- ここを追加 -->



トップページにアクセスすると変数pageは、layout, url, content, pathの各要素を持っていることが分かります。

同じことを記事のページ(Post)でもしてみます。\_posts/2013-05-18-Ruby-is-Great.mdに追加します。

---

layout: post

title: Rubyは最高

tags: [ruby, language]

---

##Rubyとは

Rubyは、手軽なオブジェクト指向プログラミングを実現するための種々の機能を持つオブジェクト指向スクリプト言語です。

{{ page }} <!-- ここを追加 -->

アクセスします。



Postでは、layout, title, tags, url, date, id, categories, next, previous, content, pathの各情報にアクセスできることが分かります。

９章　Liquidによるシンタックスハイライト

さて、次にRubyの記事にプログラムコードを書いてみます。JekyllではLiquidタグを利用することによって、[Pygments](http://pygments.org/" \o "Pygments — Python syntax highlighter)によるシンタックスハイライト（Syntax Highlighting）が可能です（version1.0でPygmentsは組み込みとなり別途インストールが不要となったようです）。

次のコマンドを実行してsyntax.cssを生成します。

% pygmentize -S default -f html > assets/css/syntax.css

pygmentizeが利用できない場合は以下のサイトなどから取得出来ます（後述するjekyll newでもsyntax.cssが得られます）。

[tpw/css/syntax.css at master · mojombo/tpw · GitHub](https://github.com/mojombo/tpw/blob/master/css/syntax.css)

\_layouts/default.htmlでこれを読み込めるようにします。

<!DOCTYPE html>

<head>

<meta http-equiv="Content-type" content="text/html; charset=utf-8">

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

<link rel="stylesheet" href="/assets/css/style.css" type="text/css">

<link rel="stylesheet" href="/assets/css/syntax.css" type="text/css"> //ここを追加

</head>

<body>

{{ content }}

<p>- rendered with layout template -</p>

</body>

準備ができました（version1.0ではpygmentsの\_config.ymlへの設定は不要になったようです）。\_posts/2013-05-18-Ruby-is-Great.mdにRubyのコードを書きます。コードは{% highlight ruby %}と{% endhighlight %}というタグで囲みます。

---

layout: post

title: Rubyは最高

tags: [ruby, language]

---

##Rubyとは

Rubyは、手軽なオブジェクト指向プログラミングを実現するための種々の機能を持つオブジェクト指向スクリプト言語です。

##Try Ruby

{% highlight ruby linenos %}

class Person

def initialize(name)

@name = name

end

def hello

"Hello, friend!\nMy name is #{@name}!"

end

end

charlie = Person.new('Charlie')

puts charlie.hello

# >> Hello, friend!

# >> My name is Charlie!

{% endhighlight %}

{{ page }}

サーバーを再起動してブラウザで確認します。



gistタグ

gistタグを使って、シンタックスハイライトされた「[Gists（コードスニペットサービス）](https://gist.github.com/" \o "Gists)」のコードを貼り付けることも可能です（version1.0でgist pluginが組み込みになりました）。

Rubyの記事でgistタグを使ってみます。

---

layout: post

title: Rubyは最高

tags: [ruby, language]

---

##Rubyとは

Rubyは、手軽なオブジェクト指向プログラミングを実現するための種々の機能を持つオブジェクト指向スクリプト言語です。

{% gist 5598133 creature.rb %} <-- ここを追加

##Try Ruby

{% highlight ruby linenos %}

class Person

def initialize(name)

@name = name

end

def hello

"Hello, friend!\nMy name is #{@name}!"

end

end

charlie = Person.new('Charlie')

puts charlie.hello

# >> Hello, friend!

# >> My name is Charlie!

{% endhighlight %}

{{ page }}

結果を見てみます。



Gistsにアップロードしておいたコードが貼り付けられました。

１０章　プラグイン

Jekyllはプラグインによりその変換機能を拡張できます[3](http://melborne.github.io/2013/05/20/now-the-time-to-start-jekyll/" \l "fn:3)。プラグインにはGenerator, Converter, Tagという３種類があります。詳細については次を参照してください。

[Plugins（英語）](http://jekyllrb.com/docs/plugins/)

[Jekyll Wiki Pluginsの説明ページを勝手に翻訳しました（version0.11における日本語訳）](http://melborne.github.io/2012/05/09/translation-of-jekyll-plugins/)

シンタックスハイライトに係るプラグインについて前章で説明しましたが、ここではTEXTEX による数式を貼り付ける[MathJax.rb](https://gist.github.com/834610" \o "MathJax)というTagプラグインを使ってみます（version1.0ではMarukuがサポートするLaTeXをpngにレンダリングする方法を使うこともできるようです。[Extras](http://jekyllrb.com/docs/extras/" \o "Extras)）。

MathJax.rbはブラウザ上でTEXTEX を表示できるようにする[MathJax.js](http://www.mathjax.org/" \o "www.mathjax.org)に対応したプラグインです。

ルートに\_plugins/ディレクトリを生成し、リンク先からMathJax.rbをダウンロードして配置します。

[MathJax.rb](https://gist.github.com/834610)

% mkdir \_plugins

% mv ~/Downloads/MathJax.rb \_plugins/mathjax.rb

\_layouts/default.htmlでMAthJax.jsを読み込めるようにします。

<!DOCTYPE html>

<head>

<meta http-equiv="Content-type" content="text/html; charset=utf-8">

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

<link rel="stylesheet" href="/assets/css/style.css" type="text/css">

<link rel="stylesheet" href="/assets/css/syntax.css" type="text/css">

<script type="text/javascript" src="http://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS-MML\_HTMLorMML"></script> <!-- ここを追加 -->

</head>

<body>

{{ content }}

<p>- rendered with layout template -</p>

</body>

準備ができたので、\_posts/2013-05-16-Jekyll-Tutorial.mdに数式を追加します。

{% math %}

Formular\ from Wikipedia\\

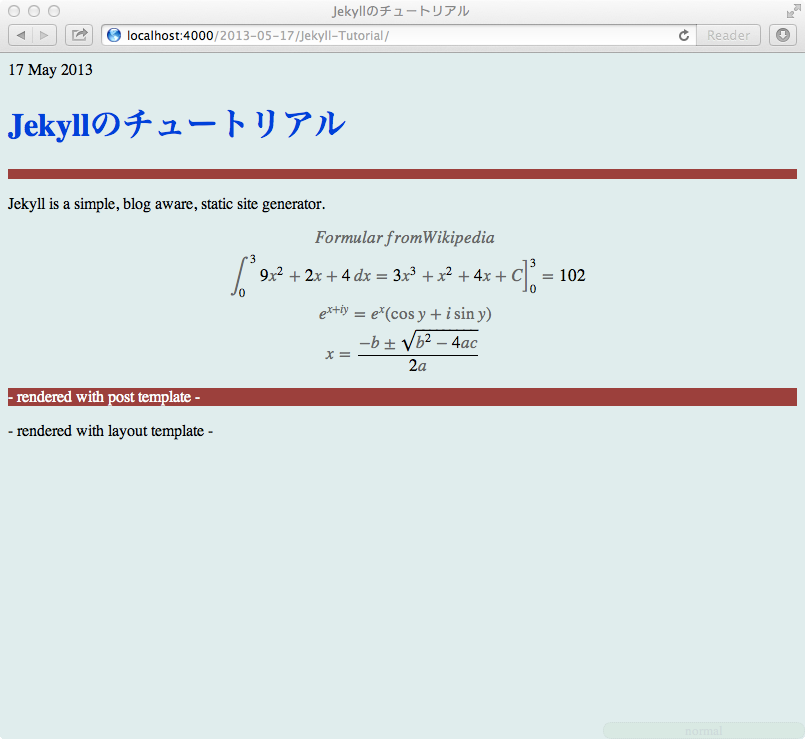
\int\_0^3 9x^2 + 2x + 4\, dx = 3x^3 + x^2 + 4x + C \Big\rbrack\_0^3 = 102\\

e^{x+iy} = e^x(\cos y + i\sin y)\\

x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}

{% endmath %}

サーバーを再起動してブラウザで確認します。



きれいな数式が表示されました。

１１章　その他の機能

バージョン1.0で新たに追加された機能のいくつかについて簡単に書いておきます。

ブログのひな形

jekyll newサブコマンドにより、簡単にブログのひな形が作れるようになりました。

ブログを構築する別のフォルダに移動して試してみます。

% jekyll new myblog

New jekyll site installed in /myblog.

% cd myblog

% tree

.

├── \_config.yml

├── \_layouts

│   ├── default.html

│   └── post.html

├── \_posts

│   └── 2013-05-17-welcome-to-jekyll.markdown

├── css

│   ├── main.css

│   └── syntax.css

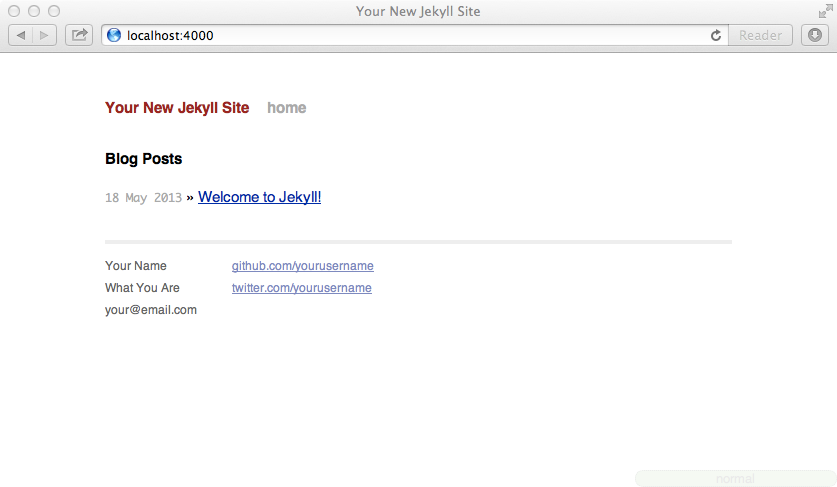
└── index.html

3 directories, 7 files

サーバーを起動してアクセスしてみます。

% jekyll serve

トップページです。



１件記事が既に書かれているので開いてみます。



シンプルですがこのまま使えるくらいのひな形が出力されました。

下書き

\_draftsフォルダ内に配置したファイルはjekyll buildによるコンパイル対象にはなりませんが、jekyll serveに--dratfsオプションを付けることでプレビューできるようになります。

チュートリアルのフォルダに戻って試してみます。

\_draftsフォルダを作り、下書きファイルをタイトルのみのファイル名で用意します。

% mkdir \_drafts

% touch \_drafts/ruby\_trivias.md

ファイルを開き下書きをします。

---

layout: post

title: Rubyのトリビア

---

Rubyのトリビアを紹介します。

jekyll serveコマンドに–draftsオプションを付けてサーバーを起動します。

% jekyll serve --drafts

本日が2013/5/17だとして、http://localhost:4000/2013-05-17/ruby\_trivias/にアクセスしてみます。



投稿日未定の記事を書くときなど便利に使えそうです。

[draftについてのドキュメント](http://jekyllrb.com/docs/upgrading/)

ページネーション

\_config.ymlでpaginate属性をセットすることで、グローバル変数「paginator」が有効になりページネーションが利用できるようになります。トップページで記事のリストを複数頁に分けるなどの目的で活用できるでしょう。ただ現状、paginatorへのアクセスはhtmlファイル内限定、ナビゲーションのためのコードが冗長などの制約があります。

[Paginationについてのドキュメント](http://jekyllrb.com/docs/pagination/)

他のブログからの移管

WordPressなどの他のブログシステムからJekyllへのデータの移管を支援する機能が提供されています。ブログシステム毎に異なるこの機能はjekyll-importというgemにまとめられていて、旧データベースにアクセスしてJekyllのためのmarkdown形式の記事を生成します。詳細は以下を参照して下さい。

[Blog migrationsについてのドキュメント](http://jekyllrb.com/docs/migrations/)

１２章　まとめ

チュートリアルは以上です。

最後にここに記述したJekyllの機能をまとめておきます。

jekyll buildコマンドは\_siteディレクトリ以下に変換ファイルを生成する。

YAML Front-Matter付きのファイルはその設定に従い変換され、YAML Front-Matter無しのファイルはそのままコピーされる。

ページごとの設定はそのYAML Front-Matterで行い、サイト全体の設定は\_config.ymlで行う。

記事ファイルは:year-:month-:day-:titleのフォーマットで、\_postsディレクトリ以下に配置する。

レイアウトファイルは\_layoutディレクトリ以下に配置する。

サイト情報にはグローバル変数siteでアクセスできる。

ページ情報にはグローバル変数pageでアクセスできる。

Liquidテンプレート{{ }}または{% %}を使ってサイトまたはページオブジェクトの埋め込みができる。

\_pluginsディレクトリ以下にPlugin Scriptを配置することにより、機能拡張が可能である。

jekyll newコマンドでブログのひな形が生成できる。



[電子書籍「30分のチュートリアルでJekyllを理解する」EPUB/MOBI版](http://gum.co/xfJY)

このリンクはGumroadにおける商品購入リンクになっています。クリックすると、オーバーレイ・ウインドウが立ち上がって、この場でクレジットカード決済による購入が可能です。購入にはクレジット情報およびメールアドレスの入力が必要になります。購入すると、入力したメールアドレスにコンテンツのDLリンクが送られてきます。

=== Ruby関連電子書籍100円で好評発売中！ ===

[M’ELBORNE BOOKS](http://melborne.github.io/books/)

[](http://melborne.github.io/books/)

Github Page上でpluginを利用するためローカルでビルドしています。 [↩](http://melborne.github.io/2013/05/20/now-the-time-to-start-jekyll/#fnref:1)

Gumroadでの最低販売価格が99¢であり、円安により100円での販売ができなくなってしまいました。 [↩](http://melborne.github.io/2013/05/20/now-the-time-to-start-jekyll/#fnref:2)

Github Pageでpluginを利用するためにはちょっとした工夫が必要になります。 [↩](http://melborne.github.io/2013/05/20/now-the-time-to-start-jekyll/#fnref:3)