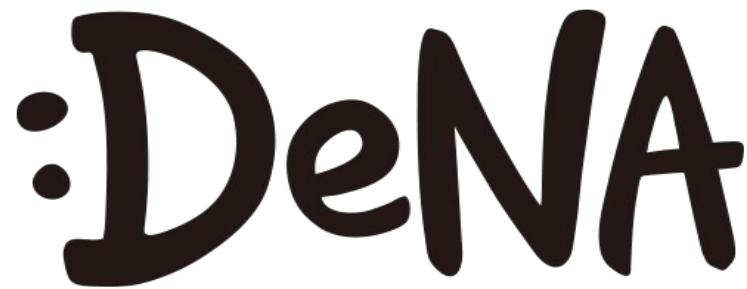


静的サイトジェネレーター(Assemble)をGitHub Pages で公開までしてみよう



April 26, 2016

Chiaki Kasai
Design Dept.
DeNA Co., Ltd.

本題の前に事前に設定をお願いします。

① Node.jsのインストール

<https://nodejs.org/en/>

② Gruntのインストール

```
$ npm install -g grunt-cli
```

③ GitHubにログイン

<https://github.com/>

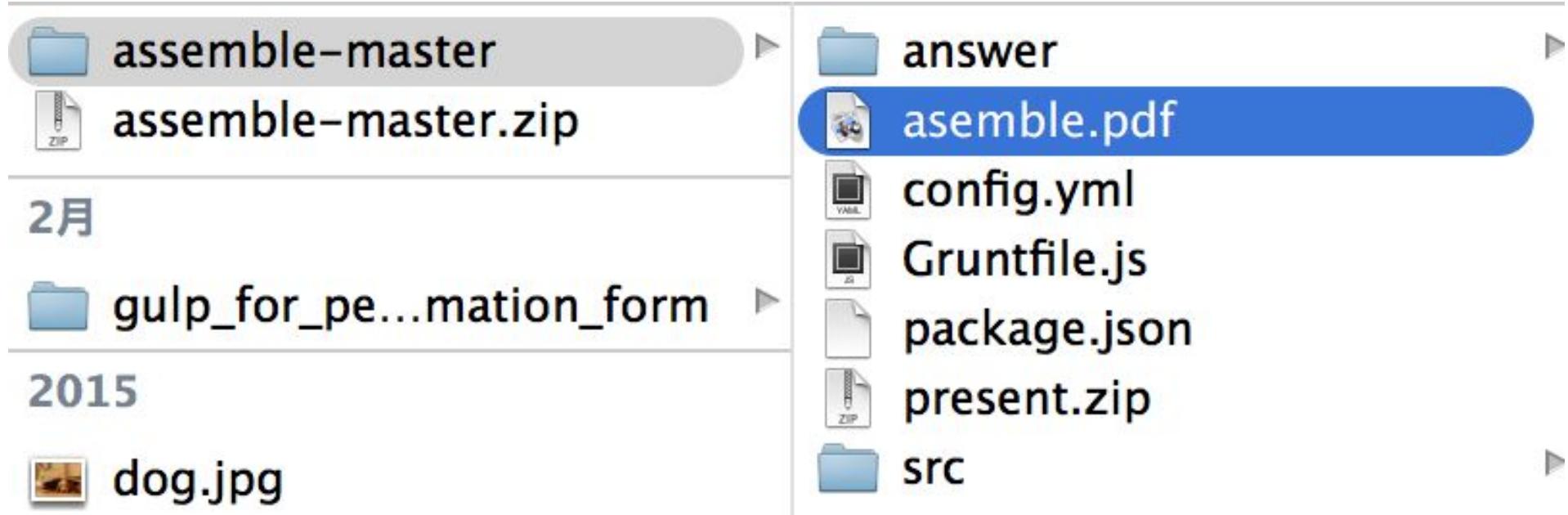
本題の前に事前に準備をお願いします。

④ 今回使用するファイルのダウンロード

<https://github.com/chimaki-kasai/assemble/archive/master.zip>

本題の前に事前に準備をお願いします。

- ⑤ ダウンロードしたassemble-masterフォルダのassemble.pdfを開いてください(今見てる資料になります)。



目次

1. 自己紹介
2. Assmbleの概要
3. **lesson1** : YAMLとhandlebarsを用いてテンプレートをコンパイルしてみよう
4. **lesson2** : partialsを使ってheaderとfooterのファイルを切り分けてみよう
5. **lesson3** : YAML front matterでページ固有の変数を定義してみよう
6. **lesson4** : helperのisを使ってみよう
7. **lesson5** : layoutsで共通のレイアウトを定義してみよう
8. **lesson6** : Github Pagesを使って静的なサイトを公開してみる

自己紹介



笠井 知晶(かさい ちあき)

2015年4月～：DeNA Webデザイナー（新卒2年目）

CSSとJavaScriptでアニメーションを巧みに実装できるフロントエンドエンジニアを目指している

【趣味】

株・音楽・健康的な食事を作る/食べる・運動・将棋・旅行

【その他】

2011年2月：ヒッチハイクで日本一周した旅のエッセイ本「ヒッチハイク女子」を徳間書店から出版



自己紹介



新規事業(UX)

⇒ マンガボックス(デザイン)



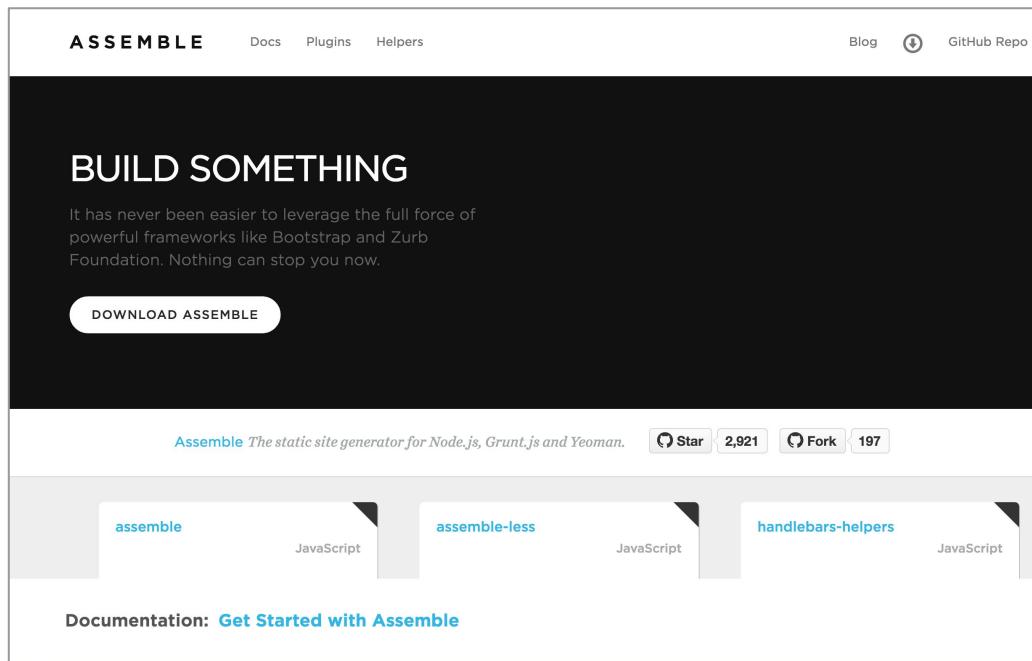
⇒ mobage関連(フロント)



2. Assembleの概要 : Assembleとは

Assemble(アセンブル)とは、Node.jsベース(Gruntで動かせる)静的サイト生成ツールです。コンパイルすることでHTMLを生成するジェネレータツールになります。

<http://assemble.io/>



2. Assmbleの概要：メリット1

- サーバーサイドのプログラムやデータベースがなくてもテンプレートシステムが使えるので、headerやfooterなどを切り分けて管理することができる
- コード量が減る
- 1つのテンプレのみ修正すれば良いのでメンテが楽

2. Assmbleの概要：メリット2：Gruntのプラグイン

AssembleはGruntのプラグインです。既にGruntを触ったことある人は、学習コストをかけずに実装できます。



2. Assmbleの概要：メリット3：Handlebarsが使える

Handlebarsは、JavaScript製のテンプレートエンジンです。

<http://handlebarsjs.com/>

The screenshot shows the official Handlebars.js website. The header features a large orange background with the word "handlebars" in white lowercase letters and a stylized mustache icon below it. To the left, there's a "DEVSWAG" badge with an orange t-shirt icon and a "NEW" badge. A callout box in the center contains text about Handlebars' power for semantic templates and its compatibility with Mustache, along with a link to details. Below the box are "Installation" and "Getting Started" buttons, and a code snippet example.

Handlebars provides the power necessary to let you build **semantic templates** effectively with no frustration.

Handlebars is largely compatible with Mustache templates. In most cases it is possible to swap out Mustache with Handlebars and continue using your current templates. Complete details can be found [here](#).

Installation

Getting Started

Handlebars templates look like regular HTML, with embedded handlebars expressions.

```
<div class="entry">
  <h1>{{title}}</h1>
  <div class="body">
    {{body}}
  </div>
</div>
```

2. Assmbleの概要：メリット4：YAMLが使える

YAML(ヤムル)とは、XMLやJSONのような、構造化されたデータを表現するテキストのフォーマットです。

jsonより手軽にインデント形式で書くことができます。

```
siteName: assemble with YAML demo
footerLinks:
  - title: assemble
    url: http://assemble.io/
  - title: DeNA
    url: http://dena.com/jp/
  - title: recruit staffing
    url: http://www.r-staffing.co.jp/
```

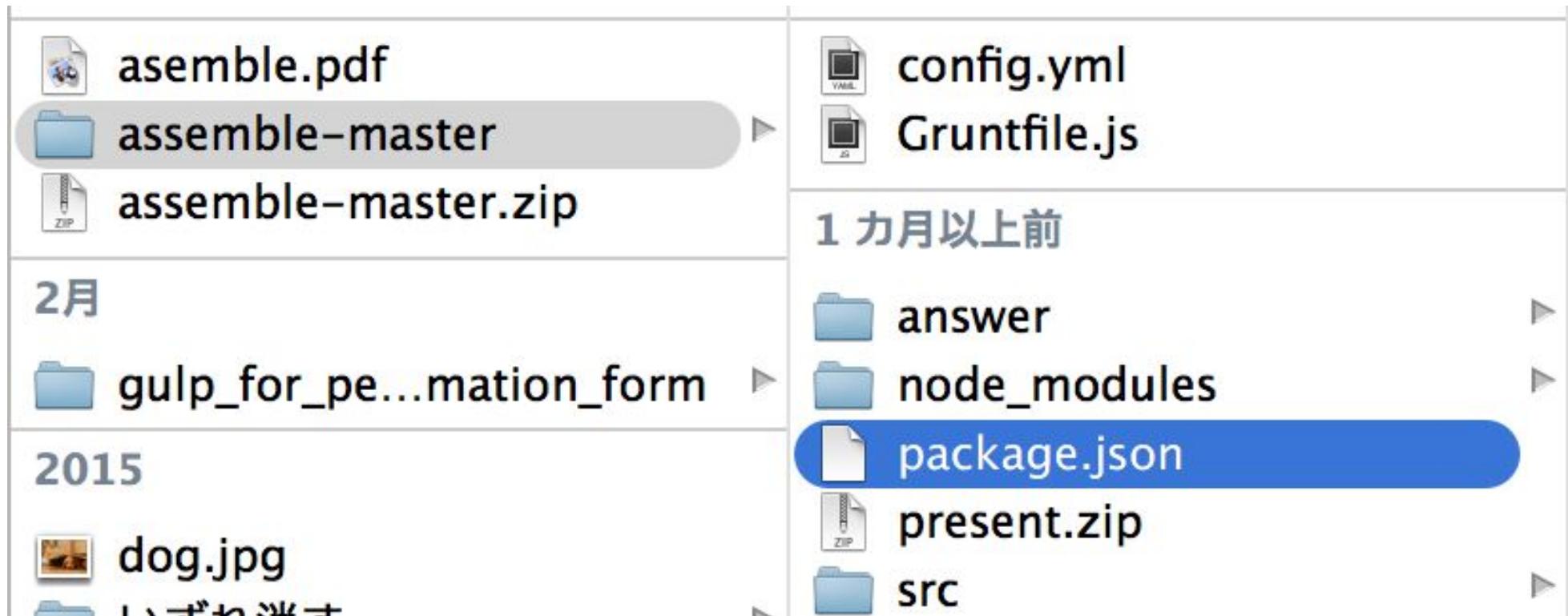
本題の前に事前に準備をお願いします(再掲)

今回使用するファイルのダウンロード

<https://github.com/chimaki-kasai/assemble/archive/master.zip>

実際にやってみましょう

ダウンロードしたファイルにターミナルコマンドで移動します



```
$ cd assemble-master
```

gruntの実行時に必要なファイルをインストールします

ダウンロードしたフォルダまで移動できたら
以下コマンドを実行してください

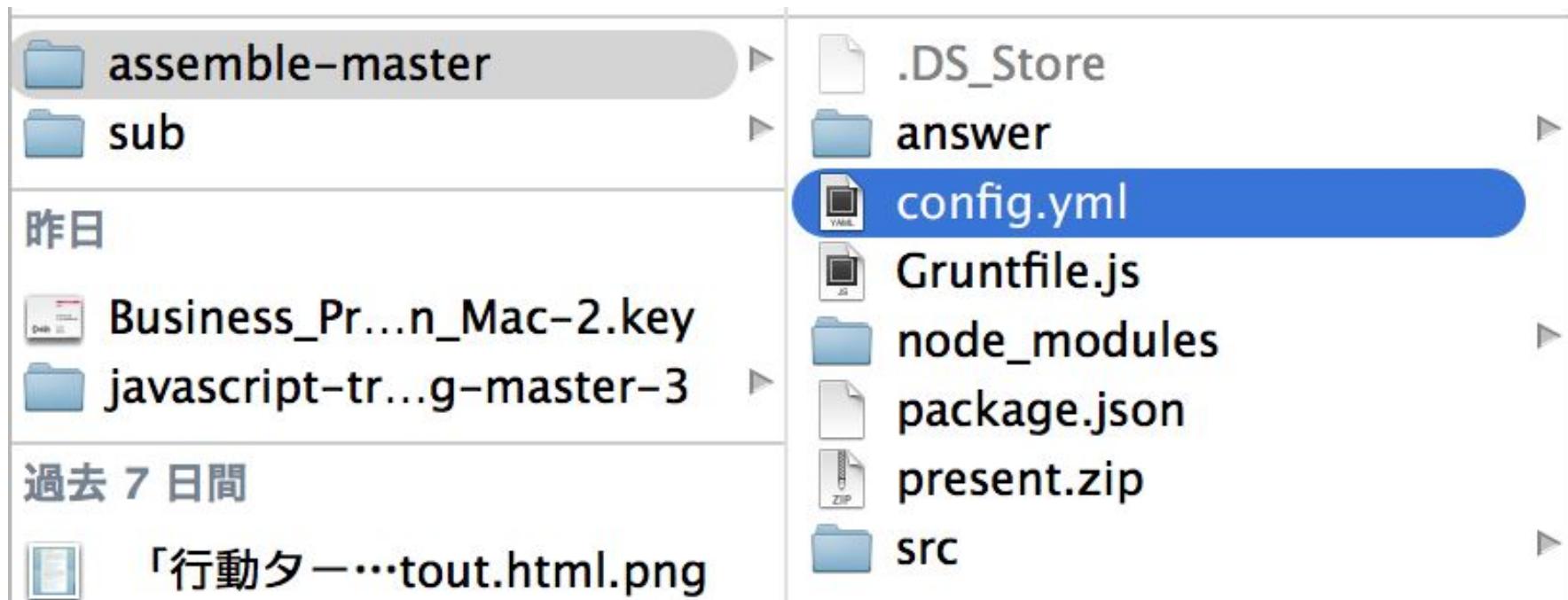
```
$ npm install
```

lesson1

YAMLとhandlebarsを用いて
テンプレートをコンパイルしてみよう

まずはファイルを見てみましょう

ダウンロードしたassemble-masterファイルのconfig.ymlを
エディタで見てみましょう



config.yml(データファイル)を見てみましょう

```
siteName: assemble with YAML demo
footerLinks:
  - title: assemble
    url: http://assemble.io/
  - title: DeNA
    url: http://dena.com/jp/
  - title: recruit staffing
    url: http://www.r-staffing.co.jp/
```

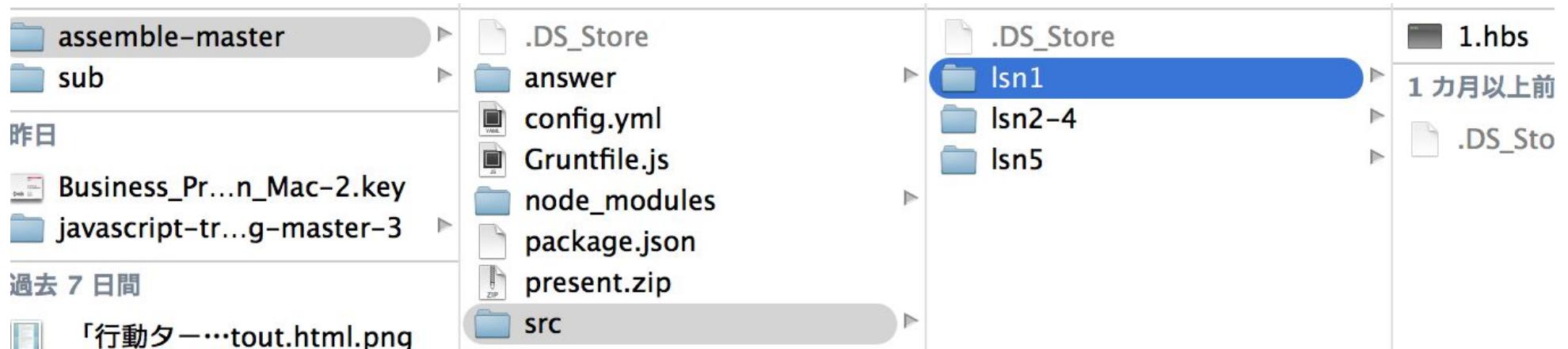
jsonより手軽にインデント形式で書けるのが特徴です

config.ymlはこのJSONとまったく同じ内容を表現しています

```
{  
    "siteName": "assemble with YAML demo",  
    "footerLinks": [  
        {  
            "title": "assemble",  
            "url": "http://assemble.io/"  
        },  
        {  
            "title": "DeNA",  
            "url": "http://dena.com/jp/"  
        },  
        {  
            "title": "recruit staffing",  
            "url": "http://www.r-staffing.co.jp/"  
        }  
    ]  
}
```

Handlebarsのファイルを見てみましょう

ダウンロードしたassemble-masterファイルの
src / lsn1 / 1.hbsをエディタで見てみましょう



Handlebarsを見てみましょう

```
<!doctype html>
<html lang="ja">
<head>
<meta charset="utf-8">
<title>{{config.siteName}}</title>
</head>
<body>
  <h1>{{config.siteName}}</h1>
  <p>Hello assmble!</p>
  <ul>
    {{#each config.footerLinks}}
      <li><a href="{{url}}">{{title}}</a></li>
    {{/each}}
  </ul>
</body>
</html>
```

Handlebarsを見てみましょう

▼ 1.hbs

```
<title>{{config.siteName}}</title>
```

▼ config.yml

```
siteName: assemble with YAML demo
```

Handlebarsを見てみましょう

▼ 1.hbs

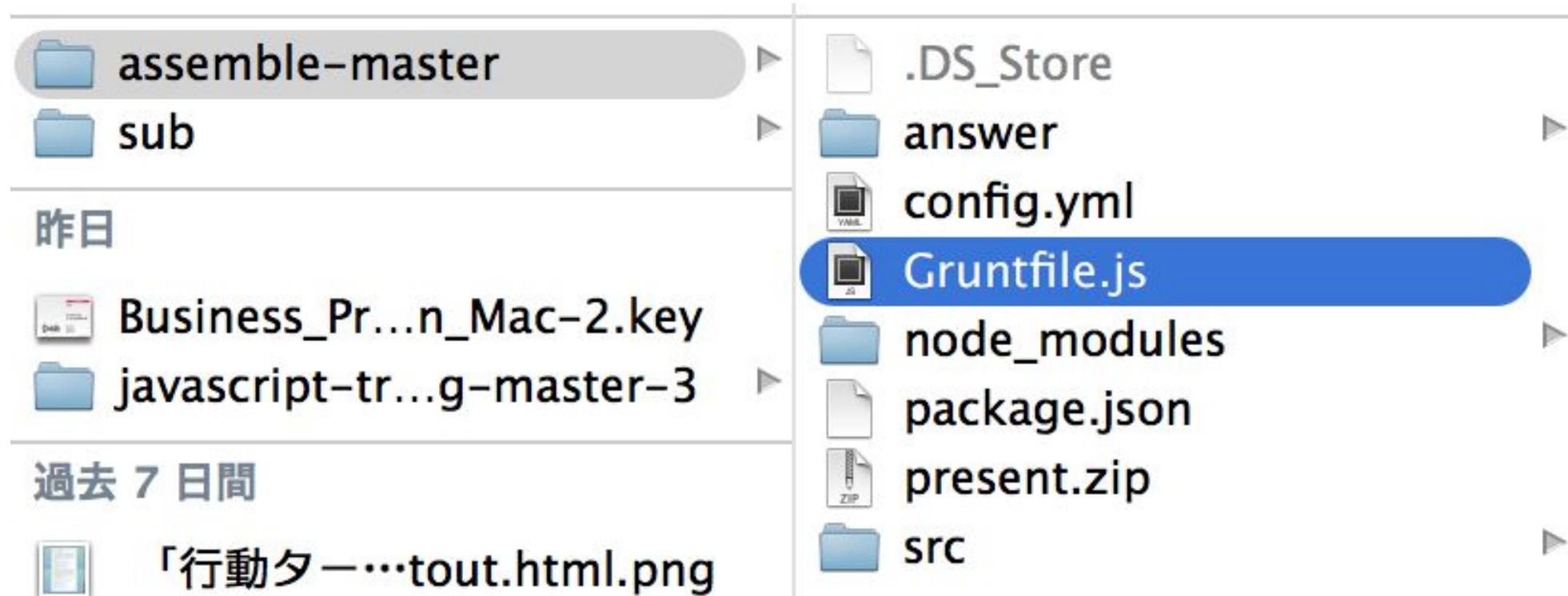
```
{{{#each config.footerLinks}}}
  <li><a href="{{url}}">{{title}}</a></li>
{{/each}}
```

▼ config.yml

```
footerLinks:
  - title: assemble
    url: http://assemble.io/
  - title: DeNA
    url: http://dena.com/jp/
  - title: recruit staffing
    url: http://www.r-staffing.co.jp/
```

実際にコードを書いてみよう

ダウンロードしたassemble-masterファイルの
Gruntfile.jsをエディタで開いてください



Gruntfile.jsを見てみましょう

```
module.exports = function(grunt) {  
  
  grunt.task.loadNpmTasks('assemble');  
  
  /* =====  
   * タスク一覧  
   * ===== */  
  
  grunt.initConfig({  
    // ここに書き足していく  
  }); // grunt.initConfig  
  
  /* =====  
   * タスク定義  
   * ===== */  
  
  // ↓ ('ターミナルで実行するコマンド名', ['実行したいgruntのタスク名'])  
  grunt.registerTask('', ['']);  
  
};
```

実際にGruntfile.jsにタスクを書いてassembleを使ってみましょう

```
assemble: {  
  
  lsn1 :{  
    options: {  
      data: ['config.yml']  
    },  
    files: [  
      {  
        src: 'src/lsn1/1.hbs',  
        dest: 'dest1/1.html'  
      }  
    ]  
  }  
}  
  
} // assemble
```

タスクの定義もしましょう

```
/* =====  
 | タスク定義  
 | =====  
 // ↓ ('ターミナルで実行するコマンド名', ['実行したいgruntのタスク名'])  
 grunt.registerTask('', []);
```

```
grunt.registerTask('lsn1', ['assemble:lsn1']);
```

実際にコンパイルしてみよう

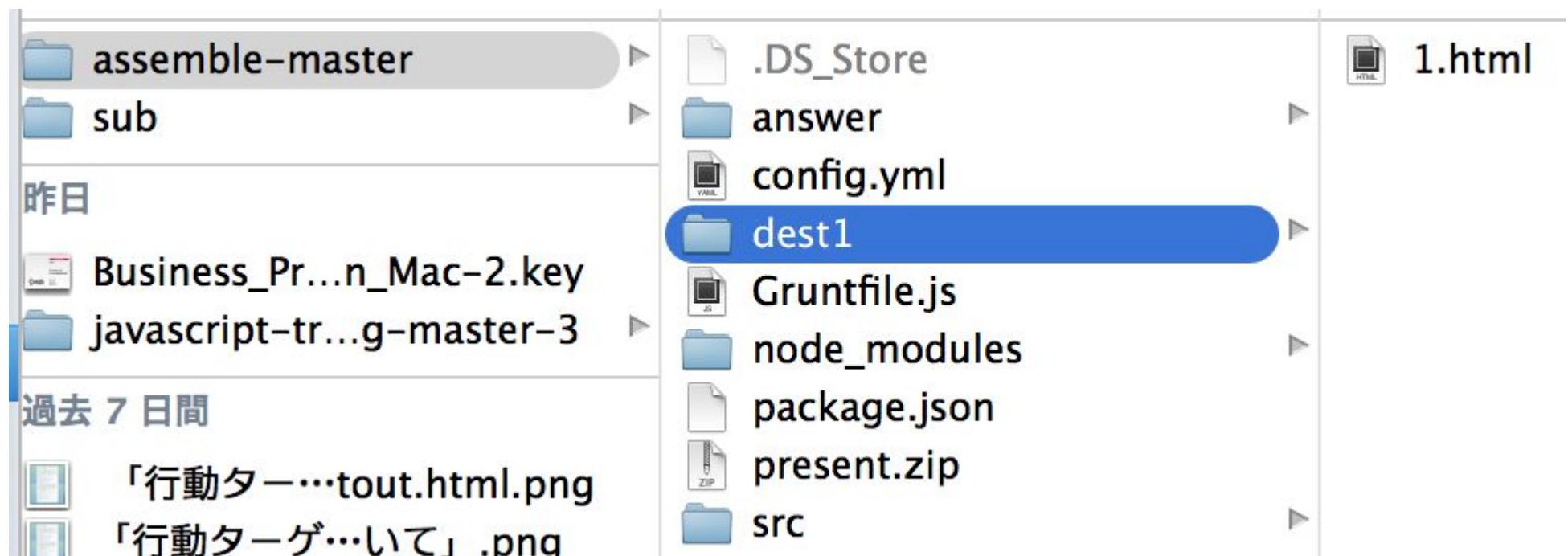
- ① Gruntfile.jsのあるディレクトリまでターミナルで移動してコマンド実行

```
$ grunt lsn1
```

実際に見てみましょう

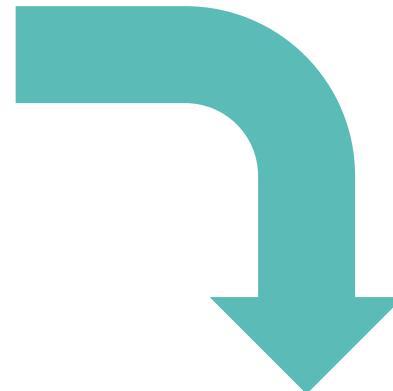
新しくdest1フォルダができると思います。

そこに先ほどのコンパイルしたテンプレートが出力されているので見てみましょう。



```
<title>{{config.siteName}}</title>
</head>
<body>
  <h1>{{config.siteName}}</h1>
  <p>Hello assmble!</p>
  <ul>
    {{#each config.footerLinks}}
      <li><a href="{{url}}">{{title}}</a></li>
    {{/each}}
  </ul>
</body>
```

コンパイル結果



```
<title>assemble with YAML demo</title>
</head>
<body>
  <h1>assemble with YAML demo</h1>
  <p>Hello assmble!</p>
  <ul>

    <li><a href="http://assemble.io/">assemble</a></li>

    <li><a href="http://dena.com/jp/">DeNA</a></li>

    <li><a href="http://www.r-staffing.co.jp/">recruit staffing</a></li>

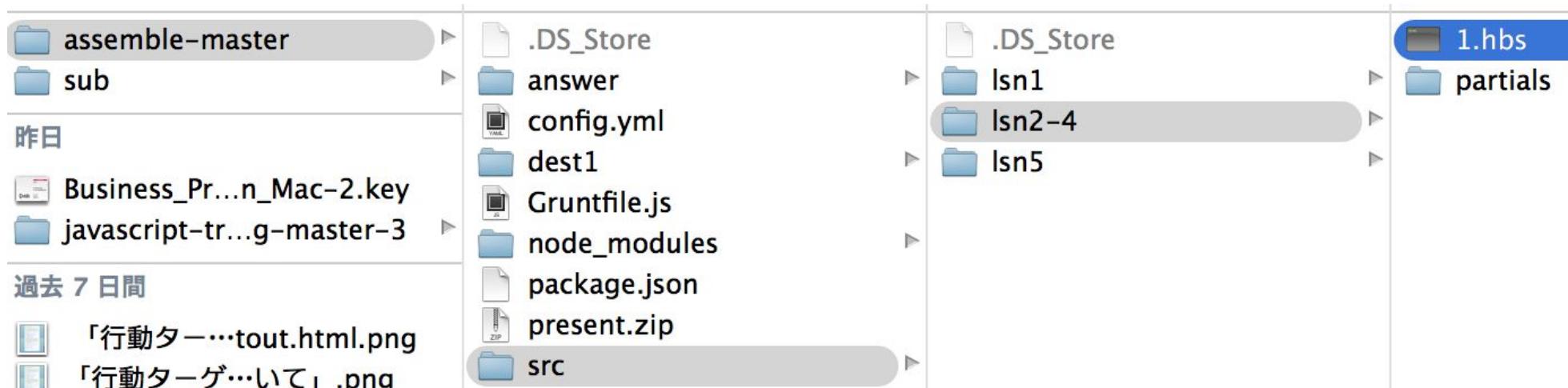
  </ul>
</body>
```

lesson2

partialsを使って
headerとfooterのファイルを切り分けてみよう

メインテンプレートとなる

src/lsn2-4/1.hbsをエディタで開いてみましょう



メインテンプレート(1.hbs)でheaderとfooterを読み込むために 黄色枠のところを変更していきます

```
<body>
  <h1>{{config.siteName}}</h1>
  <p>Hello assmble!</p>
  <ul>
    {{#each config.footerLinks}}
      <li><a href="{{url}}">{{title}}</a></li>
    {{/each}}
  </ul>
</body>
```

h1をheaderに、ulをfooterに書き換えましょう

```
<body>
{{> header}}
<p>Hello assemble!</p>
{{> footer}}
</body>
```

切り分けたheaderとfooterを作っていくましょう

partialsフォルダにheader.hbsとfooter.hbsがあるので
エディタで開いて好きなように編集しましょう



Gruntfile.jsにlsn2のタスクを追加しましょう

```
,lsn2: {  
  options: {  
    data: ['config.yml'],  
    partials: 'src/lsn2-4/partials/*.hbs'  
  },  
  files: [  
    {  
      src: 'src/lsn2-4/1.hbs',  
      dest: 'dest2/1.html'  
    }  
  ]  
}
```

タスクの定義もしましょう

```
/* =====  
 | タスク定義  
 | =====  
 // ↓ ('ターミナルで実行するコマンド名', ['実行したいgruntのタスク名'])  
 grunt.registerTask('', []);
```

```
grunt.registerTask('lsn2', ['assemble:lsn2']);
```

実際にコンパイルしてみよう

- ① Gruntfile.jsのあるディレクトリでコマンド実行

```
$ grunt lsn2
```

実際に見てみましょう

新しくdest2フォルダができると思います。

1.htmlをエディタで開いてみると

headerとfooterがちゃんと読み込んでいます。

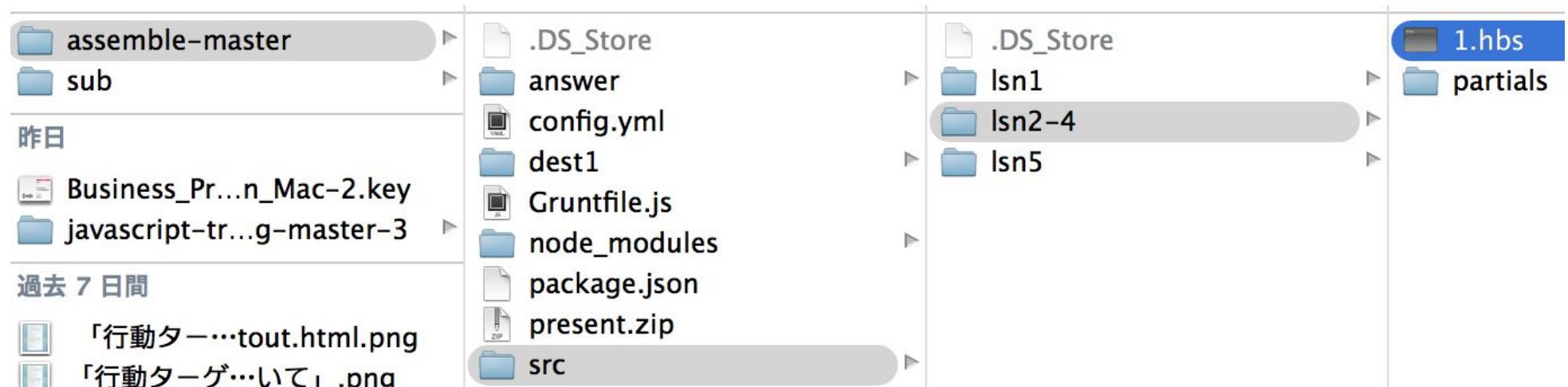
```
<body>
  <header>
    <h1>ここは切り分けたheader部分です</h1>
  </header>
  <p>Hello assmble!</p>
  <footer>
    <p>ここはfooterです</p>
  </footer>
</body>
```

lesson3

YAML front matterでページ固有の変数を定義してみよう

使うテンプレートは先ほどと同じです

※ 開いてない人はsrc/lxn2-4/1.hbsをエディタで開いてみましょう



YAML front matterは そのページ内だけに有効な変数を利用することが可能です

```
---
```

```
pageTitle: タイトルだよ
pageDescription: ページの説明だよ
---
```

```
<!doctype html>
<html lang="ja">
<head>
<meta charset="utf-8">
<title>{{pageTitle}} - {{config.siteName}}</title>
</head>
<body>
  {{> header}}
  <p>
    {{pageDescription}}
  </p>
  {{> footer}}
</body>
</html>
```

1.hbsに好きな変数を書いてみよう

pageTitle: タイトルだよ

pageDescription: ページの説明だよ

```
<title>{{pageTitle}} - {{config.siteName}}</title>
```

```
<p>
```

```
    {{pageDescription}}
```

```
</p>
```

Gruntfile.jsにlsn3のタスクを追加しましょう(dest3だけが変更)

```
,lsn3: {  
  options: {  
    data: ['config.yml'],  
    partials: 'src/lsn2-4/partials/*.hbs'  
  },  
  files: [  
    {  
      src: 'src/lsn2-4/1.hbs',  
      dest: 'dest3/1.html'  
    }  
  ]  
}
```

タスクの定義もしましょう

```
/* =====  
 | タスク定義  
 | =====  
 // ↓ ('ターミナルで実行するコマンド名', ['実行したいgruntのタスク名'])  
 grunt.registerTask('', []);
```

```
grunt.registerTask('lsn3', ['assemble:lsn3']);
```

実際にコンパイルしてみよう

- ① Gruntfile.jsのあるディレクトリでコマンド実行

```
$ grunt lsn3
```

実際に見てみましょう

新しくdest3フォルダができると思います。

YAML front matterがちゃんと読み込めてます。

```
<title>タイトルだよ - assemble with YAML demo</title>
</head>
<body>
  <header>
    <h1>ここは切り分けたheader部分です</h1>
  </header>
  <p>
    ページの説明だよ
  </p>
  <footer>
    <p>ここはfooterです</p>
  </footer>
</body>
```

lesson4

helperのisを使ってみよう

isはassembleのhelperです。

簡単に言うと「 A is B : A = B 」で、

イコール(=)の場合trueになります。

使い方はif文とほぼ同じです。

```
{#{is game "FFRK"}}

<p>
  ファイナルファンタジー レコードキーパーは、
  スクウェア・エニックスとDeNAの共同制作によるスマートフォン用ゲーム。
  基本プレイ無料。
</p>
{else}
<p>
  DeNAと任天堂が共同開発したアプリ
  「Miitomo(ミートモ)」是非使ってみてね。
</p>
{{/is}}
```

YAML front matterの変数がFFRKなら、上のpタグが表示される

```
---
```

```
pageTitle: 会社情報
pageDescription: 会社情報を掲載しています。
game: FFRK
```

```
---
```

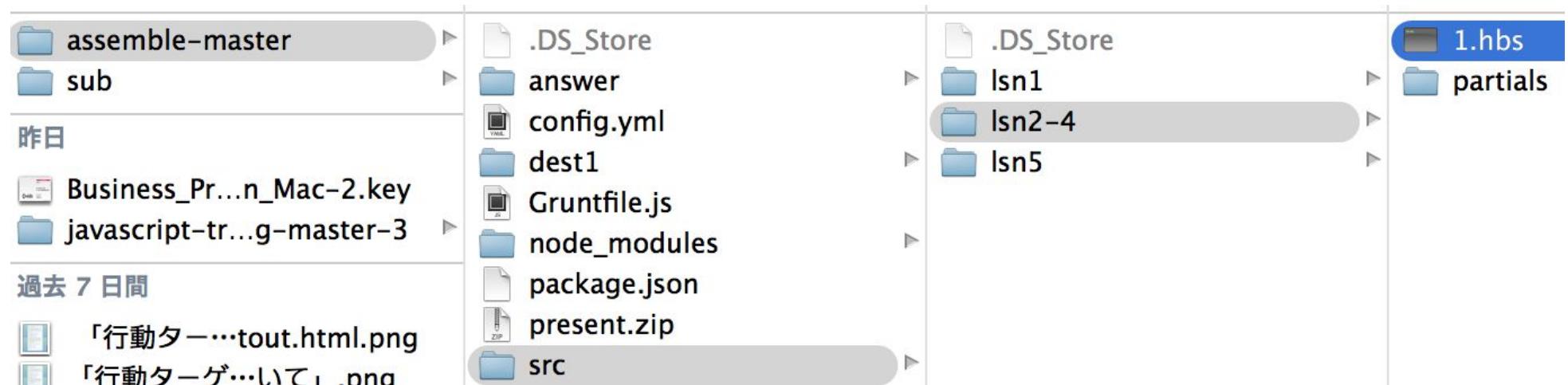
```
<!doctype html>
<html lang="ja">
<head>
<meta charset="utf-8">
<title>{{pageTitle}} - {{config.siteName}}</title>
</head>
<body>
{{> header}}
```

```
<p>{{pageDescription}}</p>
```

```
  {{#is game "FFRK"}}
    <p>
      ファイナルファンタジー レコードキーパーは、
      スクウェア・エニックスとDeNAの共同制作によるスマートフォン用ゲーム。
      基本プレイ無料。
    </p>
  {{else}}
    <p>
      DeNAと任天堂が共同開発したアプリ
      「Miitomo(ミートモ)」是非使ってみてね。
    </p>
  {{/is}}
```

使うテンプレートは先ほどと同じです

※ 開いてない人はsrc/lxn2-4/1.hbsをエディタで開いてみましょう



1.hbsに好きな変数と条件分岐を追加しましょう

game: FFRK

```
{#is game 'FFRK'}
```

<p>FFRK面白いよ</p>

```
{else}
```

<p>Miitomo面白いよ</p>

```
{/is}
```

Gruntfile.jsにlsn4のタスクを追加しましょう(dest4だけが変更)

```
,lsn4: {  
  options: {  
    data: ['config.yml'],  
    partials: 'src/lsn2-4/partials/*.hbs'  
  },  
  files: [  
    {  
      src: 'src/lsn2-4/1.hbs',  
      dest: 'dest4/1.html'  
    }  
  ]  
}
```

タスクの定義もしましょう

```
/* =====  
 | タスク定義  
 | =====  
 // ↓ ('ターミナルで実行するコマンド名', ['実行したいgruntのタスク名'])  
 grunt.registerTask('', []);
```

```
grunt.registerTask('lsn4', ['assemble:lsn4']);
```

実際にコンパイルしてみよう

- ① Gruntfile.jsのあるディレクトリでコマンド実行

```
$ grunt lsn4
```

実際に見てみましょう

新しくdest4フォルダに1.htmlができると思います。

FFRKの変数がある場合のpタグが表示されています。

```
<p>  
    ページの説明だよ  
</p>
```

```
<p>FFRK面白いよ</p>
```

```
<footer>  
    <p>ここはfooterです</p>  
</footer>
```

elseの場合も見てみましょう

1.hbsのgameの変数を変えてみましょう

game: Miitomo

実際にコンパイルしてみよう

- ① Gruntfile.jsのあるディレクトリでコマンド実行

```
$ grunt lsn4
```

実際に見てみましょう

新しくdest4フォルダに1.htmlができると思います。

FFRKの変数がない場合のpタグが表示されています。

```
<p>
    ページの説明だよ
</p>
```

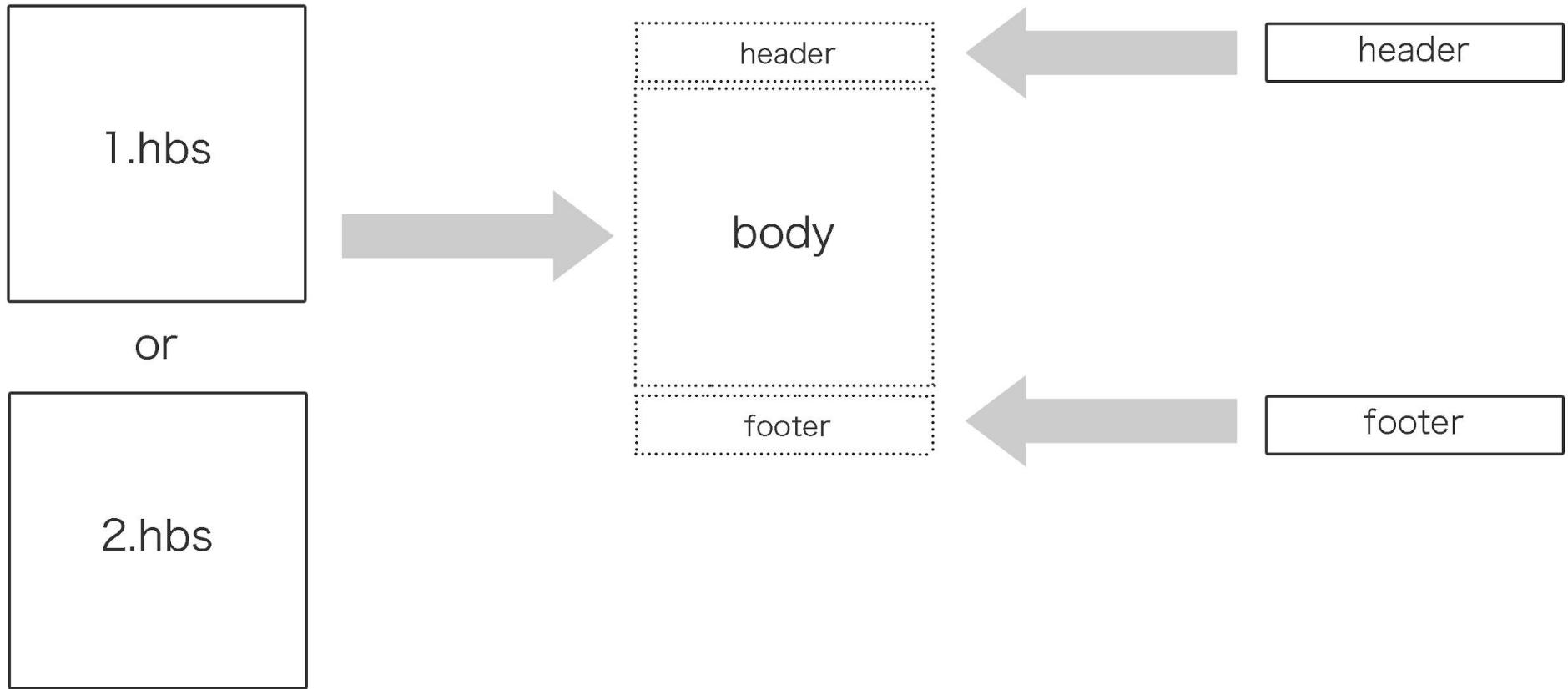
```
<p>Miitomo面白いよ</p>
```

```
<footer>
    <p>ここはfooterです</p>
</footer>
```

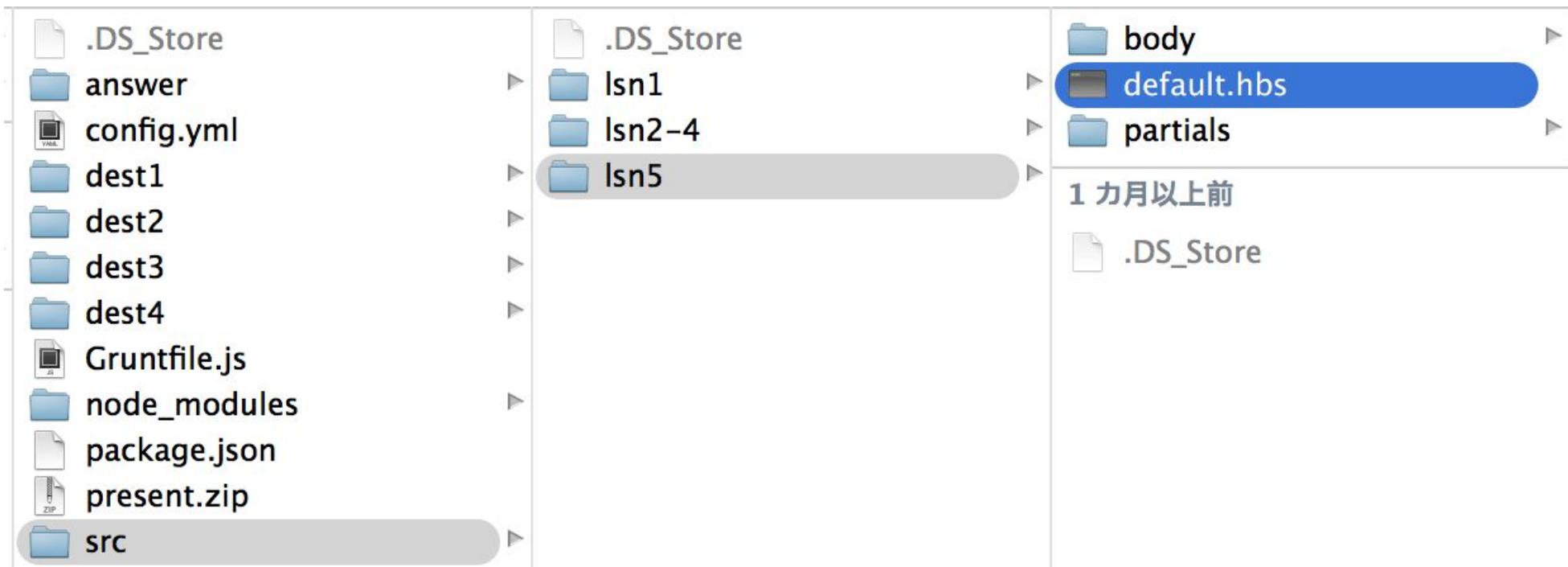
lesson5

layoutsで共通のレイアウトを定義してみよう

layoutsを使ってbodyも切り出し任意のファイルを読み込む



Isn5/default.hbsをエディタで開いてみましょう



default.hbsに新しくbody変数を追加しましょう

```
{{> body}}
```

bodyフォルダにある

body1.hbsとbody2.hbsをエディタで開いてみましょう



body1.hbs

body2.hbs

を自由に編集しましょう

中身はそれぞれ違うと良いですね

Gruntfile.jsにlsn5のタスクを追加しましょう

```
lsn5: {  
  options: {  
    data: ['config.yml'],  
    partials: 'src/lsn5/partials/*.hbs',  
    layout: 'src/lsn5/default.hbs',  
    flatten: true  
  },  
  files: [  
    {  
      src: 'src/lsn5/body/*.hbs',  
      dest: 'dest5/'  
    }  
  ]  
}
```

タスクの定義もしましょう

```
/* =====  
 | タスク定義  
 | =====  
 // ↓ ('ターミナルで実行するコマンド名', ['実行したいgruntのタスク名'])  
 grunt.registerTask('', []);
```

```
grunt.registerTask('lsn5', ['assemble:lsn5']);
```

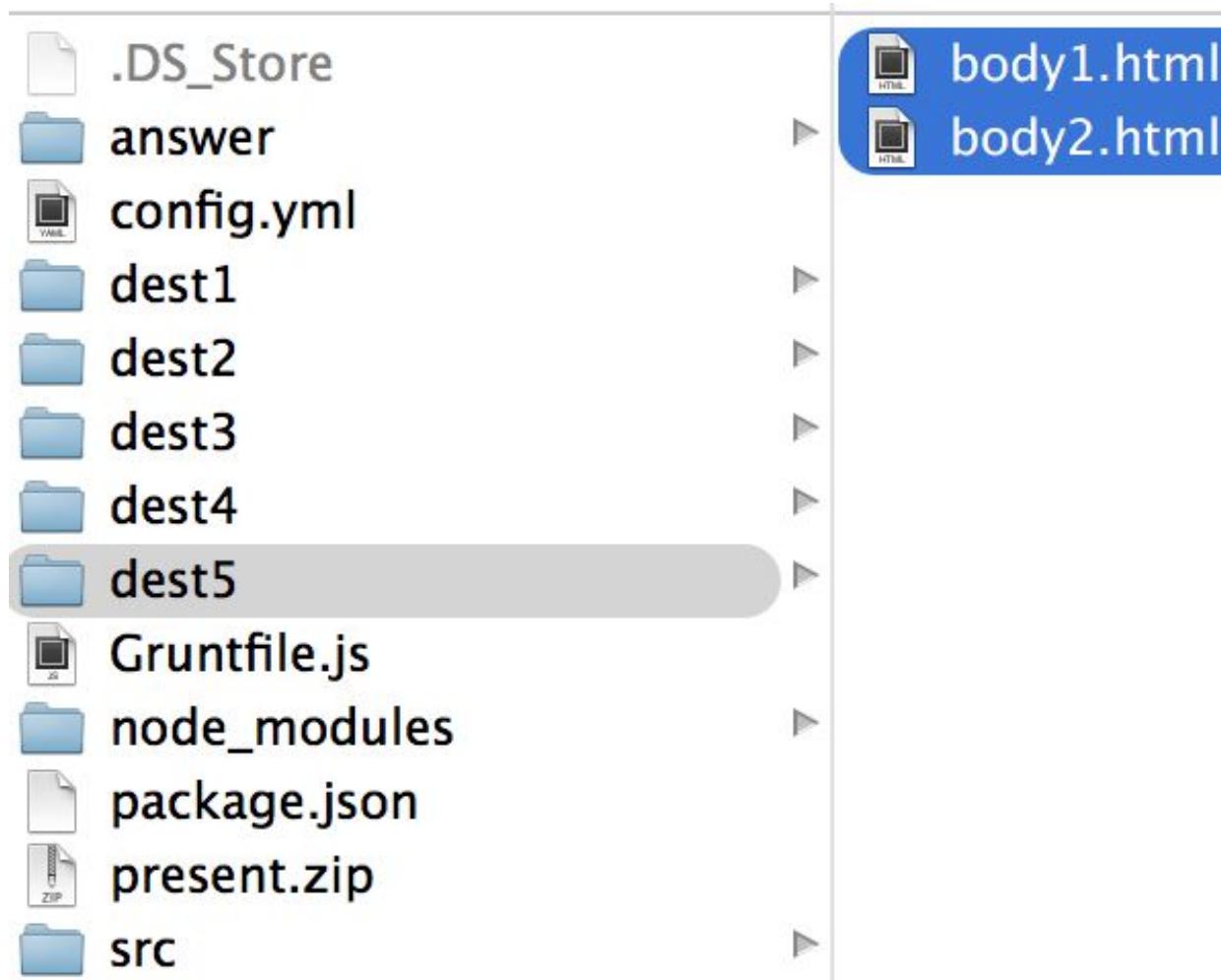
実際にコンパイルしてみよう

- ① Gruntfile.jsのあるディレクトリでコマンド実行

```
$ grunt lsn5
```

実際に見てみましょう

新しくdest5フォルダができると思います。



lesson6（最後）

Github Pagesを使って静的なサイトを公開してみる

GitHub Pagesとは

静的サイトのファイルをGithubで管理して、Githubのリポジトリへプッシュすれば公開できるというものです。

※ データベースを用いるような動的なウェブページは公開できません。
そして、リポジトリは公開されてしまうので注意しましょう。

<https://pages.github.com/>

実際にやってみよう

① GitHubにログインしましょう

<https://github.com/>

New repositoryを作ります

Your repository "chimaki-kasai/assemble" was successfully deleted.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Let's get started!](#)

chimaki-kasai ▾

yutakam80 forked [chimaki-kasai/assemble_20160426](#) to [yutakam80/assemble_20160426](#) 3 days ago

Welcome to GitHub! What's next? (on 5 Jul 2015)

Create a repository
Tell us about yourself
Browse interesting repositories
Follow @github on Twitter

💡 ProTip! Edit your feed by updating the users you [follow](#) and repositories you [watch](#).

(悰) Expanded webhook events

You can now subscribe to additional webhook events

View 20 new broadcasts

Your repositories 0

[New repository](#)

You don't have any repositories yet!
Create your first repository or learn more about Git and GitHub.

Subscribe to your news feed

Repository name(assemble_test)を決めて Create repositoryを押します。

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name

assemble_test



Great repository names are short and memorable. Need inspiration? How about [scaling-computing-machine](#).

Description (optional)



Public
 Anyone can see this repository. You choose who can commit.



Private
 You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

git cloneする前に任意のフォルダを作ってcdで移動してください

▼ desktopに作った場合

```
$ cd desktop/任意のファイル名
```

▼ その他に作った場合(以下の方法で簡単に移動)

```
$ cd フォルダをターミナルにドラック&ドロップ
```

※ cd とだけターミナルにうって、任意のファイルをドラック&ドロップするだけでフォルダ名が出るのでenterを押してください

任意のファイルがある場所で コマンドを使ってローカルにクローンします

The screenshot shows a GitHub repository page for 'chimaki-kasai/assemble_test'. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Wiki, Pulse, Graphs, and Settings. Below the navigation, there's a 'Quick setup' section with a note about previous experience. It shows two cloning options: 'Set up in Desktop' and 'HTTPS' (which is selected and highlighted with a red box). The HTTPS URL is https://github.com/chimaki-kasai/assemble_test.git. A red box also highlights the copy icon in the top right corner of this section. Below this, there's a section for creating a new repository on the command line, with a code block containing the following commands:

```
echo "# assemble_test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/chimaki-kasai/assemble_test.git
git push -u origin master
```

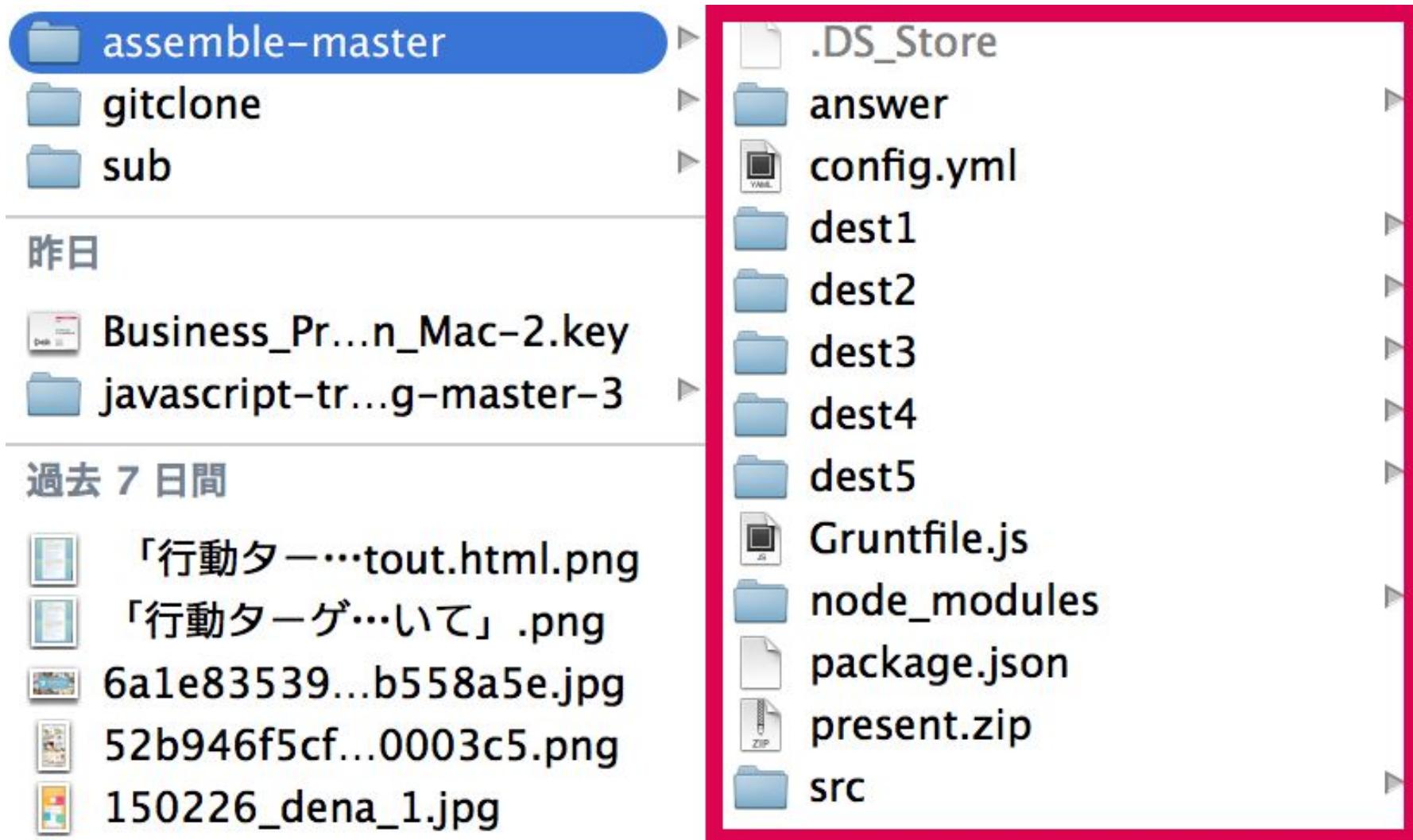
A red box highlights the copy icon in the top right corner of this command block.

\$ git clone コピペ

cloneしたフォルダへ移動してください

```
$ cd assemble_test
```

assemble_masterのファイルを git cloneしたフォルダassembleに移動orコピペしましょう



masterブランチにadd / commit / push

① assemble_20160426のあるディレクトリでコマンド実行

```
$ git add -A
```

```
$ git commit -m "first commit"
```

```
$ git push origin master
```

GitHubに反映されました

1 commit 1 branch 0 releases 0 contributors

Branch: master [New pull request](#) [New file](#) [Upload files](#) [Find file](#) [HTTPS](#) <https://github.com/chimak> [Raw](#) [Download ZIP](#)

Chiaki Kasai first commit	Latest commit e1cf3d3 38 seconds ago
answer	first commit 38 seconds ago
dest1	first commit 38 seconds ago
dest2	first commit 38 seconds ago
dest3	first commit 38 seconds ago
dest4	first commit 38 seconds ago
dest5	first commit 38 seconds ago
node_modules	first commit 38 seconds ago
src	first commit 38 seconds ago
Gruntfile.js	first commit 38 seconds ago
config.yml	first commit 38 seconds ago
package.json	first commit 38 seconds ago
present.zip	first commit 38 seconds ago

Help people interested in this repository understand your project by adding a README.

[Add a README](#)

次にgh-pages(表示用ブランチ)にもpush

```
$ git subtree push --prefix dest1/ origin gh-pages
```

※ 最初のlesson1で作ったファイル(dest1)のみを
今回は表示確認用にpushしたいと思います

画面確認

chimaki-kasai / **assemble_test**

Code Issues 0 Pull requests 0 Wiki Pulse Graphs **Settings**

No description or website provided. — Edit

1 commit 1 branch 0 releases 0 contributors

Branch: master New pull request New file Upload files Find file HTTPS https://github.com/chimak   Download ZIP

 Chiaki Kasai first commit	Latest commit 90c8021 31 seconds ago	
 answer	first commit	30 seconds ago
 dest1	first commit	30 seconds ago

画面確認

GitHub Pages

✓ Your site is published at http://chimaki-kasai.github.io/assemble_test.

Update your site
To update your site, push your HTML or Jekyll updates to your gh-pages branch. Read the [Pages help article](#) for more information.

Overwrite site
Replace your existing site by using our automatic page generator. Author your content in our Markdown editor, select a theme, then publish.

[Launch automatic page generator](#)

Danger Zone

Make this repository private
Please [upgrade your plan](#) to make this repository private.

[Make private](#)

Transfer ownership
Transfer this repository to another user or to an organization where you have admin rights.

[Transfer](#)

Delete this repository
Once you delete a repository, there is no going back. Please be certain.

[Delete this repository](#)

画面確認 /1.html を追加してください



404

File not found

The site configured at this address does not contain the requested file.

If this is your site, make sure that the filename case matches the URL.
For root URLs (like `http://example.com/`) you must provide an `index.html` file.

[Read the full documentation](#) for more information about using **GitHub Pages**.

[GitHub Status](#) — [@githubstatus](#)





ご清聴有難うございました！