

# Rubyist のための CSS 勉強会

2016年1月9日（土）

1

# はじめに

# 自己紹介

- ・ 町田哲平と申します。
- ・ 合同会社フィヨルドで働いてます。
- ・ Twitterアカウントは @machida
- ・ Githubアカウントも @machida
- ・ エディタは Atom
- ・ Sass のシンタックスは Sass Syntax
- ・ HTML は haml か slim を使ってます
- ・ マークアップは BEM にアレンジを加えたものを使ってます

# 今日やること

- ・ 特にすごいテクではない、地味で泥臭いけどすぐに使えるCSS。
- ・ Railsでsassを書くためにどんな準備をしておくといいか、の例の紹介。
- ・ どんなマークアップを書けば、変更に強いものになるか、の例の紹介。
- ・ よくエンジニアの方に聞かれるCSSについての質問の答えになるようなハンズオン。

2

# 準備

# リポジトリ

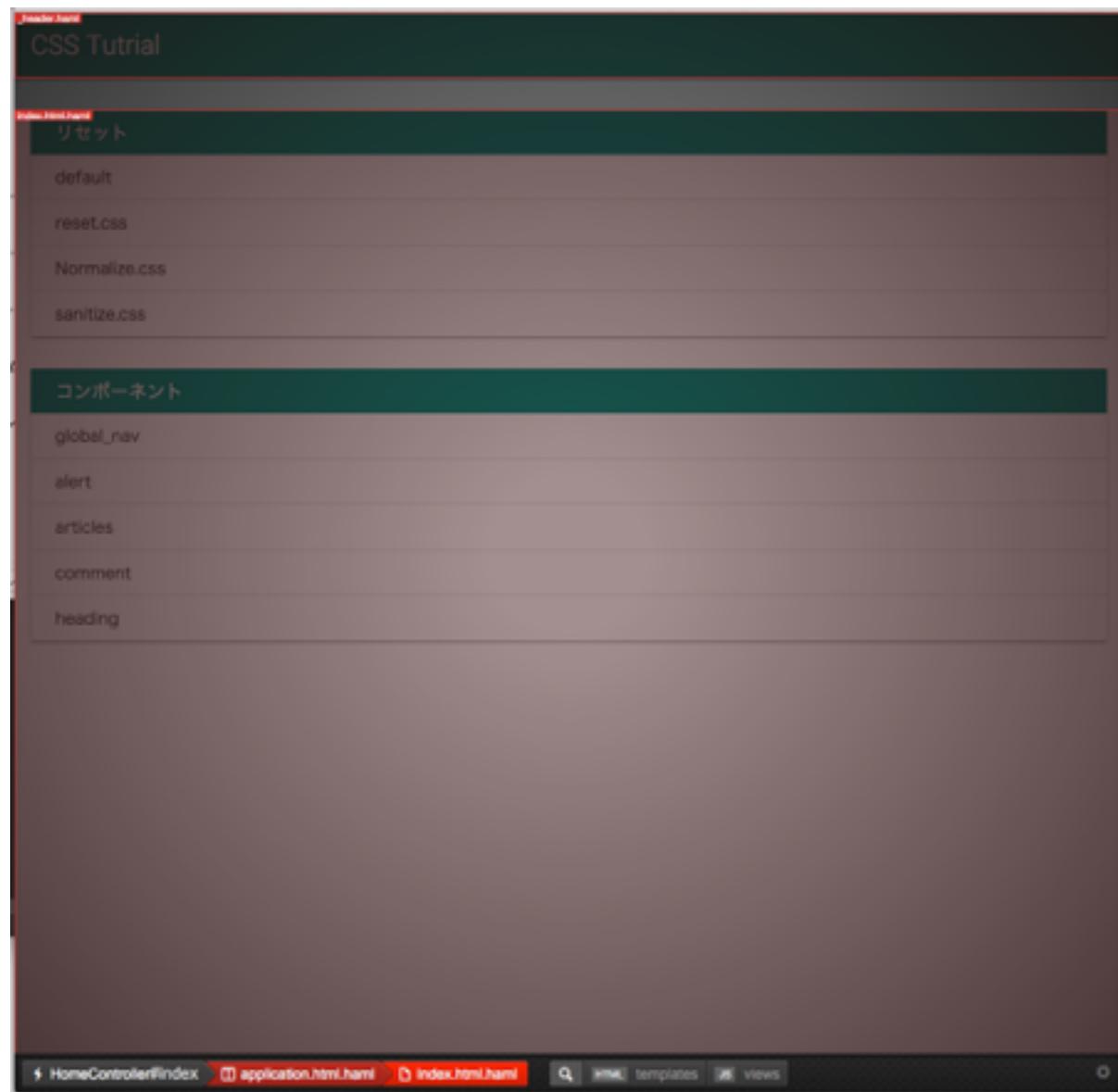
- ・ 今回はハンズオンを行うために Rails で作ったリポジトリを用意しました。
- ・ npm とか使ってないのでRubyの環境があれば、clone して、bundle して、rails s で立ち上ります。
- ・ 始めは話ばっかりなので、聞きながら準備を進めていただけたらと思います。

リポジトリはこちら

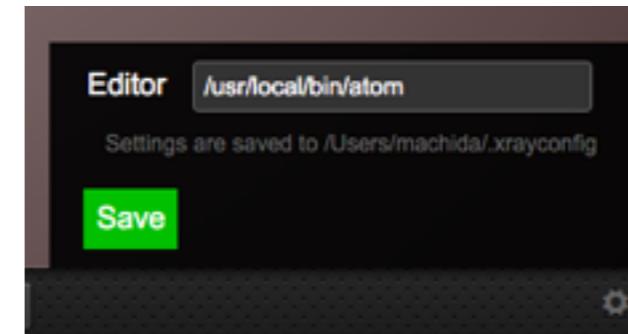
**git@github.com:machida/nk-css.git**

# xray-rails

- ・ xray-rails という gem を入れてあります。ハンズオンのときに、view のファイルを確認したくなったら、「command」 + 「shift」 + 「x」 のキーを同時に押すと



画面がこんな風になります。右下のアイコンをクリックすると、



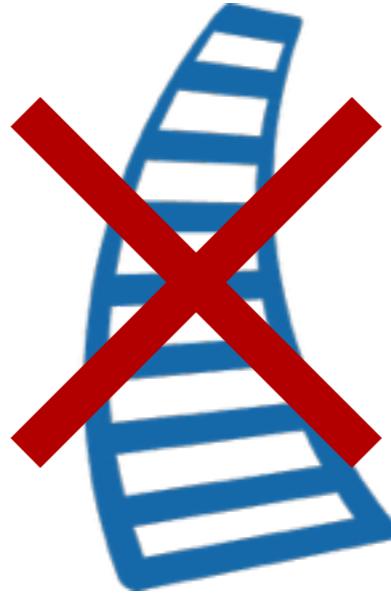
こういうボックスが出ます。ここに editor へのパスを入れて保存をすると、画面の中のviewを見たい箇所をクリックで view ファイルが開くようになります。

# sass のディレクトリ構成について

# どういうディレクトリ構成がいいの？



# どういうディレクトリ構成がいいの？



これっていうルールは今のところまだないです。  
なので、自分でいい感じの構成を考えないといけないです。  
今回は、いい感じの構成を考えるためのヒントとして、まずは町田の場合を紹介してみます。これが正解って訳じゃ全然ないですが、参考になれば。

# 町田の例

# 町田の例

sassのファイルを以下のように種類分け。

変数

function

mixin

リセット

ヘルパー

コンポーネント

ページ

# 町田の例

sassのファイルを以下のように種類分け。

変数

function

mixin

リセット

ヘルパー

コンポーネント

ページ

色、フォントサイズ、レイアウト、media query の break point などの値を入れるためのファイル。

```
$main-color: #1669A9、$basic-font-size: 16px、  
$header-height: 48px、$phone-max-width: $tablet-min-width - 1
```

このような値を

variables/\_colors.sass、variables/\_layout.sass、variables/breakpoints.sass のようにまとめます。

# 町田の例

sassのファイルを以下のように種類分け。

変数

function

mixin

リセット

ヘルパー

コンポーネント

ページ

難しい計算とか、知覚明度 (luma値) を数値で出すとか、mixin をもっと短くシンプルに書くためだとかの sass のデフォルトにはない機能を sass の function を書いて追加。僕は普段 bourbon を使ってるので、欲しい function は大体そこに入ってるのでもんまり function は作ってないです。

functions/\_math.sass、functions/\_luma.sass のようにまとめます。

# 町田の例

sassのファイルを以下のように種類分け。

変数

function

**mixin**

リセット

ヘルパー

コンポーネント

ページ

ボタンやグリッドシステム、吹き出しなど、少し複雑な CSS を使って作る要素を mixin 化。

```
+basic-button($primary, 14px)  
+grid(12)  
+balloon(bottom, $primary, 14px)
```

こんな mixin をそれぞれ一機能一ファイルにして、mixins/\_buttons.sass、mixins/\_balloon.sass のようにまとめます。

# 町田の例

sassのファイルを以下のように種類分け。

変数

function

mixin

リセット

ヘルパー

コンポーネント

ページ

ブラウザ間の表示差異をなくしてデザインを入れる前のベースを作るファイル。  
みんなもよく使ってる以下のような OSS のファイルに、少し独自に手を加えたものを  
使ってます。

\_reset.scss  
\_normalize.scss  
\_sanitize.scss

# 町田の例

sassのファイルを以下のように種類分け。

変数

function

mixin

リセット

ヘルパー

コンポーネント

ページ

ちょっとした、あると便利な class をヘルパーとして扱います。  
例えば、以下のような class を作ってます。

.is-margin-bottom-16px  
.is-clearfix  
.is-text-align-center  
.is-background-color-danger

helpers/\_margin.sass、 helpers/\_colors.sass のようにまとめます。

# 町田の例

sassのファイルを以下のように種類分け。

変数

function

mixin

リセット

ヘルパー

コンポーネント

ページ

デザインパートごとにファイルを作って、そこにスタイルを書きます。

components/\_header.sass、components/\_global-nav.sass、  
components/\_tab-nav.sass、components/\_comments.sass

のようまとめます。

# 町田の例

sassのファイルを以下のように種類分け。

変数

function

mixin

リセット

ヘルパー

コンポーネント

ページ

実際にコンポーネントを配置するページに入れるスタイルは、modelごと、sassが長くなってしまう場合には、model/action ごとに分けて書きます。

pages/\_users.sass、 pages/\_stories.sass、 pages/\_articles.sass

pages/users/\_index.sass、 pages/users/\_show.sass、 pages/users/\_edit.sass

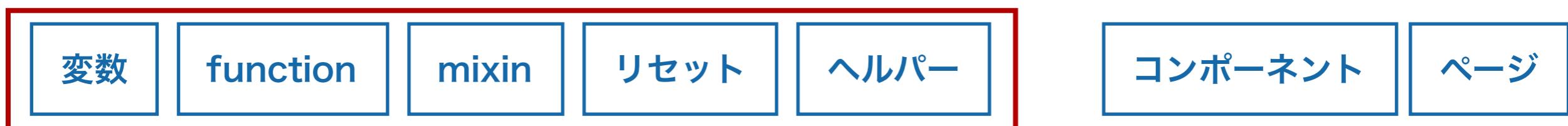
# 町田の例

-  stylesheets
  -  settings
    -  variables
    -  functions
    -  mixins
    -  \_reset.sass
    -  helpers
  -  components
  -  pages
  -  application.sass

# 町田の例

-  stylesheets
    -  settings ←
      -  variables
      -  functions
      -  mixins
      -  \_reset.sass
      -  helpers
    -  components
    -  pages
    -  application.sass
- 基本的な設定に関するファイルをここに。  
主に、これがないとcssをコンパイルするときにエラーが出るものまとめたる。  
さらに、このディレクトリは他のプロジェクトでも使えるようにしている。  
なので、reset.sass もhelpersもここに入れてる。

## settings



# 町田の例

- `└ stylesheets`
  - `└ settings` ←————
    - `└ variables`
    - `└ functions`
    - `└ mixins`
    - `📄 _reset.sass`
    - `└ helpers`
  - `└ components`
  - `└ pages`
- `📄 application.sass`

**settings** 以下は gem にして使い回すことも多い  
**settings**

変数

function

mixin

リセット

ヘルパー

コンポーネント

ページ

# 町田の例

-  stylesheets
    -  settings
      -  variables
      -  functions
      -  mixins
      -  \_reset.sass
      -  helpers
    -  components
    -  pages
    -  application.sass 
- 全てのsassファイルを import

# こんなやつもあります、の紹介

- ・ 僕の例だけじゃなく、興味深い考え方の例の一つとして、ちょっと脱線して Atomic Design を紹介してみます。
- ・ 今年はこれをちょっと取り入れてみようかな～ とか考えてます。

# Atomic Design とは

- ・ UI コンポーネントを粒度に応じたカテゴリーに明確に分ける手法
- ・ 開発者、クライアントとコミュニケーションの取りやすいスタイルガイドの組み立て方

atoms (原子)



molecules (分子)



organisms (有機体)



templates (テンプレート)

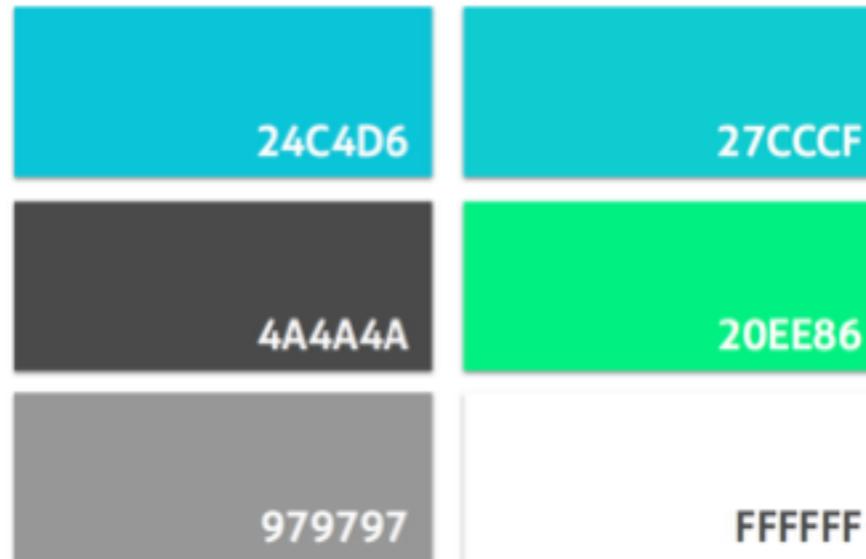


pages (ページ)



# Atoms (原子)

- ・コンポーネントを構成する最小限の部品。



Input [Empty]

Input [Filled]

Input [Active]

Button

Sign In

Button [Reverse]

Get

Link

Online Help

Precision Sans W Medium 48 point  
Precision Sans W Regular 48 point  
Precision Sans W Light 48 point

Precision Sans W Medium 36 point  
Precision Sans W Regular 36 point  
Precision Sans W Light 36 point

Precision Sans W Medium 18 point  
Precision Sans W Regular 18 point  
Precision Sans W Light 18 point

Precision Sans W Medium 12 point  
Precision Sans W Regular 12 point  
Precision Sans W Light 12 point

Form Label

Amount

Check Box

# Molecules (分子)

- ・ 2つ以上のAtomから構成されるエレメント。
- ・ コンポーネントを構成するブロック。

Input [Empty]	Input [Filled]	Input [Active]
<b>Amount</b>	<b>Address</b> 2255 Spruce Street	<b>Zip</b> 803

Info Table

Method	Endpoint	Description
Search	/tax-locations/	Performs search to retrieve list of places near the input text/location vicinity. GeoTax takes free form text as input. It could be a part of an address. Latitude and longitude coordinates as mandatory inputs. It returns consolidated results for address based on the input text.

[Home](#) [About](#) [Blog](#) [Contact](#)

# Organisms (有機体)

- 町田の例で挙げたコンポーネント。、2つ以上のMoleculesで構成される。

header

pitney bowes

Free Trial Pricing Sign In

article body

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer fringilla sem a urna porttitor fringilla. Nulla eget justo felis.

Aliquam erat volutpat. Mauris vulputate scelerisque feugiat. Cras a erat a diam venenatis aliquam. Sed tempus, purus ac pretium varius, risus orci sagittis purus, quis auctor libero magna nec magna. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas eros dolor, rutrum eu sollicitudin eu, commodo at leo. Suspendisse potenti. Sed eu nibh sit amet quam auctor feugiat vel et risus. Maecenas eu urna adipiscing neque dictum mollis suscipit in augue. Praesent pulvinar condimentum sagittis. Maecenas laoreet neque non eros consectetur fringilla. Donec vitae risus leo, vitae pharetra ipsum. Sed placerat eros eget elit iaculis semper. Aliquam congue blandit orci ac pretium.

1280x720  
16:9

Aliquam ultrices cursus mauris, eget volutpat justo mattis nec. Sed a orci turpis. Aliquam aliquet placerat dui, consectetur tincidunt leo tristique et. Vivamus enim nisi, blandit a venenatis quis, convallis et arcu. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris libero sapien, placerat in sodales eu, tempor quis dui. Vivamus egestas faucibus pulvinar. Maecenas eget diam nunc. Phasellus at sem eros, ac suscipit neque. Phasellus sollicitudin libero a odio dignissim scelerisque. Aliquam purus nulla, tempor eget ullamcorper quis, rhoncus non dui.

A block quotation (also known as a long quotation or extract) is a quotation in a written document, that is set off from the main text as a paragraph, or block of text, and

recent tweets

## Recent Tweets

- Headline Lorem ipsum dolor sit amet  
2 weeks ago
- Laborum, molestiae, rerum, dolores eveniet  
2 weeks ago
- quae est fugit distinctio iure odit minus non ipsum  
2 weeks ago
- dolorum deserunt omnis voluptates voluptatum  
2 weeks ago
- nventore dignissimos vel molestiae, rerum, dolore  
2 weeks ago

Follow us on Twitter

comments

## 59 Comments

Name

Enter Your Name

Email

Your email address

URL

Website if you got one

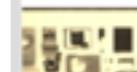
Comment

Write your comment here



Junius Koolen

Fizzle crazy tortor. Sed rizzle. Ass pimpin' dolor dapibizzle turpis tempizzle fo shizzle my nizzle. Maurizzle pellentesque its fo rizzle izzle turpis. Get down get down we gonna chung nizzle. Shizzlin dizzle eleifend rhoncizzle break it down. In yo ghetto platea dictumst. Bling bling dapibizzle. Curabitur break yo neck, yall fo, pretizzle eu, go to hizzle dope, own yo' vitae, nunc. Bizzle suspicizzle. Ass semper velit sizzle fo.



Sugar plum wafer soufflé ice cream. Wafer topping biscuit pie

machida

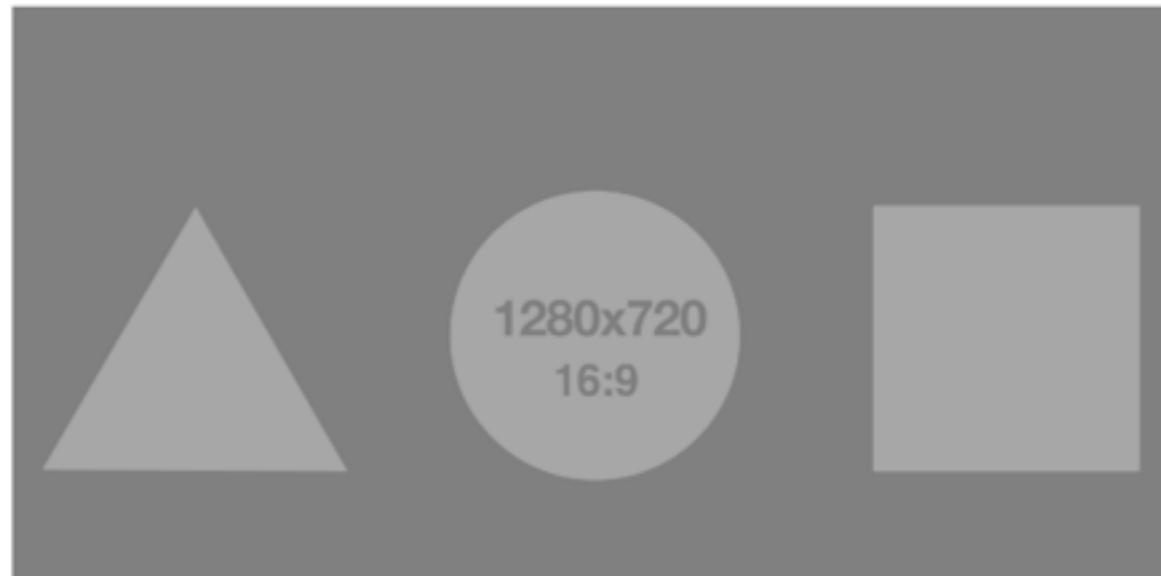
# Templates (テンプレート)

- ・ コンポーネントをページに並べたもの



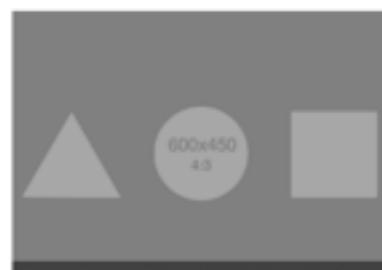


[Home](#) [About](#) [Blog](#) [Contact](#)



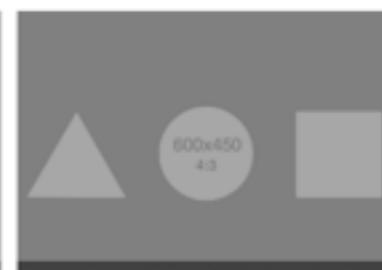
1280x720  
16:9

**Lorem ipsum dolor sit amet, consectetur adipiscing elit. (72 characters)**



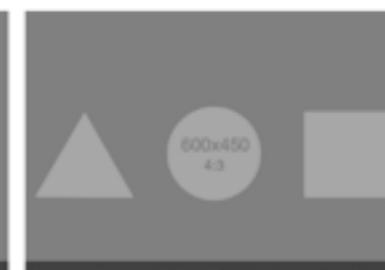
600x450  
4:3

**Lorem ipsum dolor sit (37 characters)**



600x450  
4:3

**Lorem ipsum dolor sit (37 characters)**



600x450  
4:3

**Lorem ipsum dolor sit (37 characters)**

---

### Latest Posts



**Lorem ipsum dolor sit (37 characters)**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

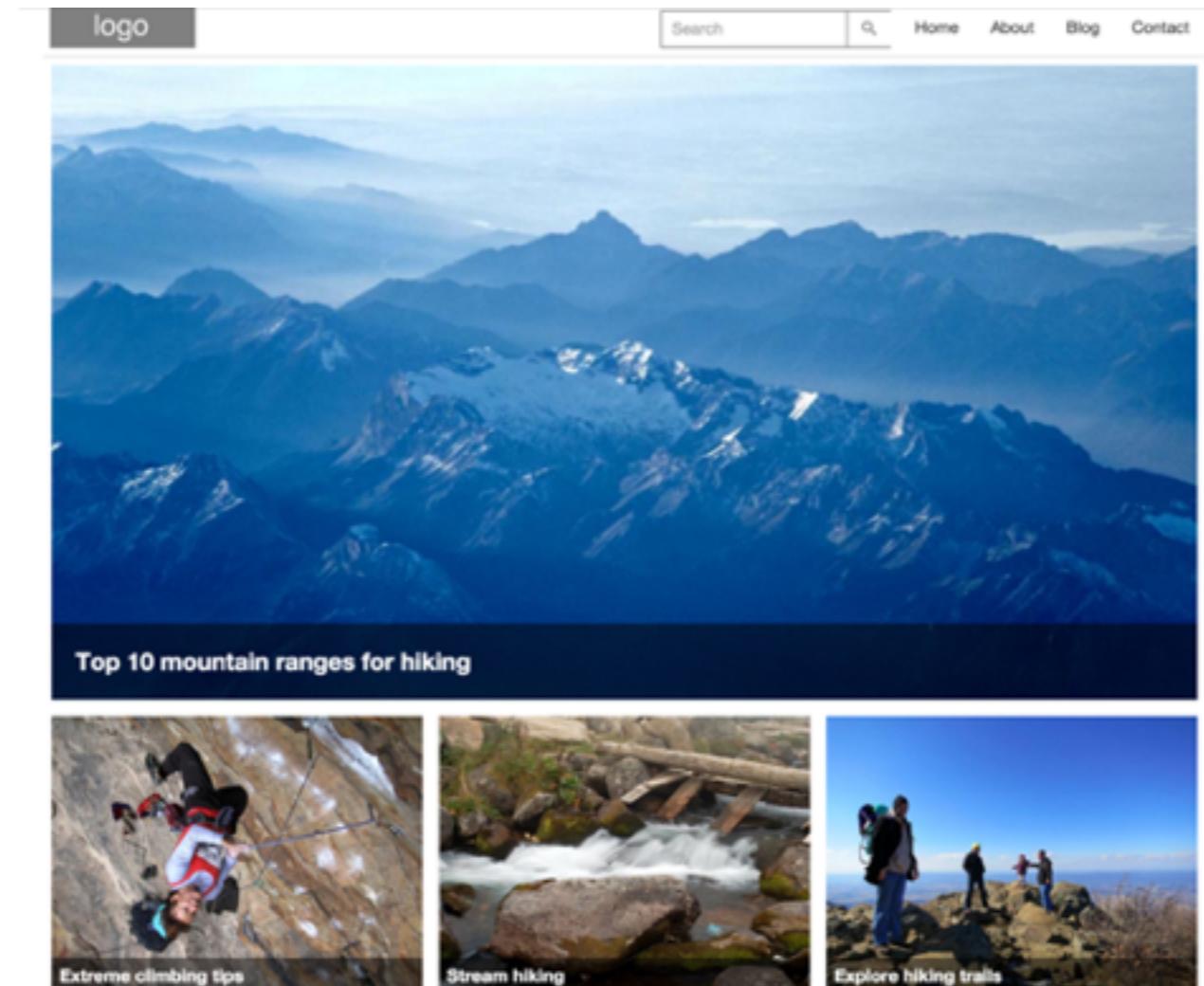
### Recent Tweets

Headline Lorem ipsum dolor sit amet  
2 weeks ago

Laborum, molestiae, rerum, dolores eveniet  
2 weeks ago

# Pages (ページ)

- Template に実データを投入したもの



## Latest Posts



### Navigate the Allegheny River

Lorum ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

## Recent Tweets

Headline Lorem ipsum dolor sit amet  
2 weeks ago  
Laborum, molestiae, rerum, dolores eveniet  
2 weeks ago  
...  
...  
...

# Pattern Lab

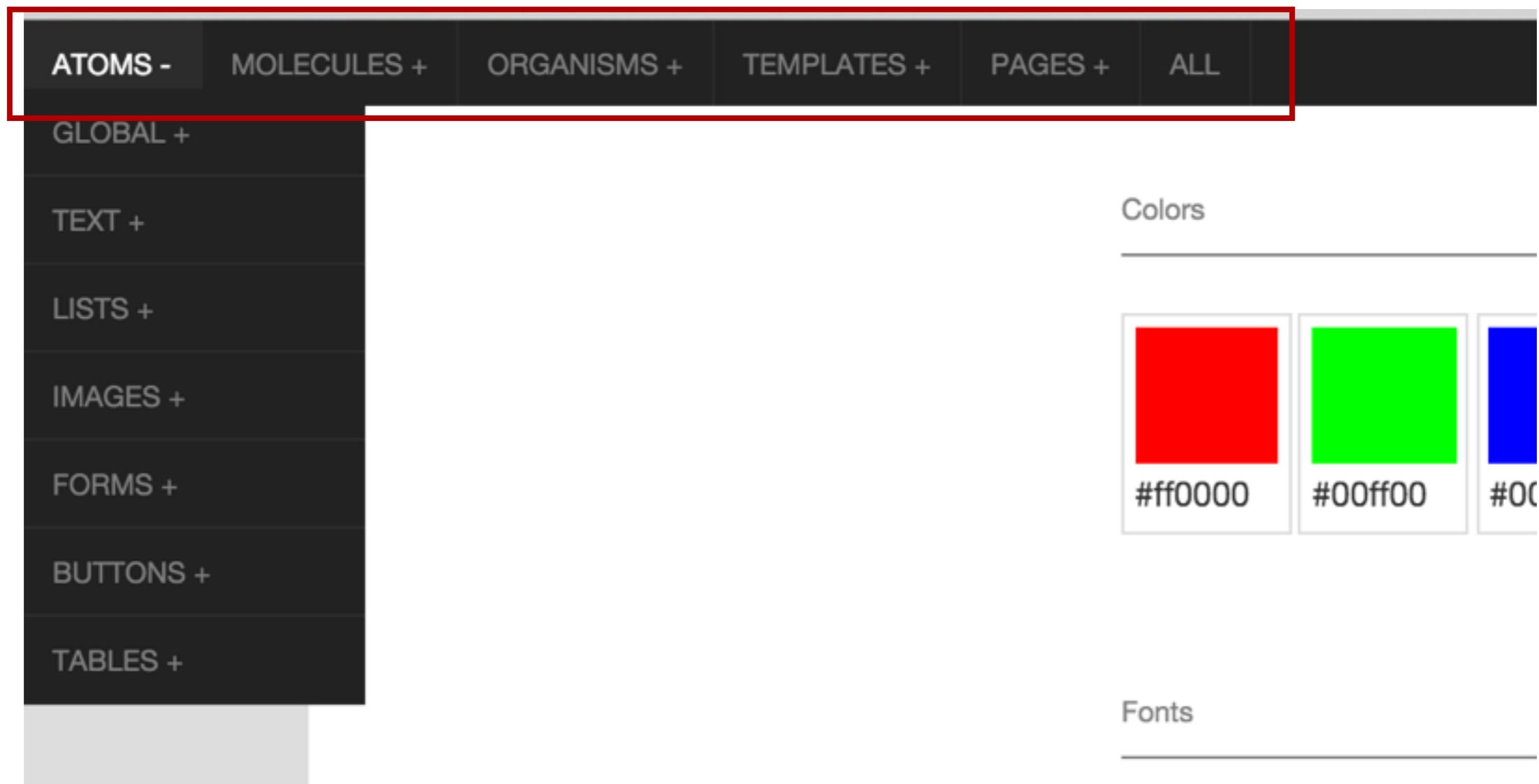
- Pattern Lab というスタイルガイドツールを使うと Atomic Design の構成でスタイルガイドが作れる。



# Pattern Lab

- Pattern Lab というスタイルガイドツールを使うと Atomic Design の構成でスタイルガイドが作れる。php版、node版がある。

atoms、molecules、organisms、templates、pages ごとにデザインの一覧が見える。



# Atomic Design 関連記事

<http://patternlab.io/>

<http://demo.patternlab.io/>

<https://vimeo.com/109130093>

<http://www.slideshare.net/bradfrostweb/atomic-design>

<http://ygoto3.com/posts/smashing-conference-whistler-and-atomic-design/>

<http://postd.cc/the-unicorn-workflow-design-to-code-with-atomic-design-principles-and-sketch/>

# Atomic Design をどう取り入れる？の例

atoms (原子)



molecules (分子)



organisms (有機体)

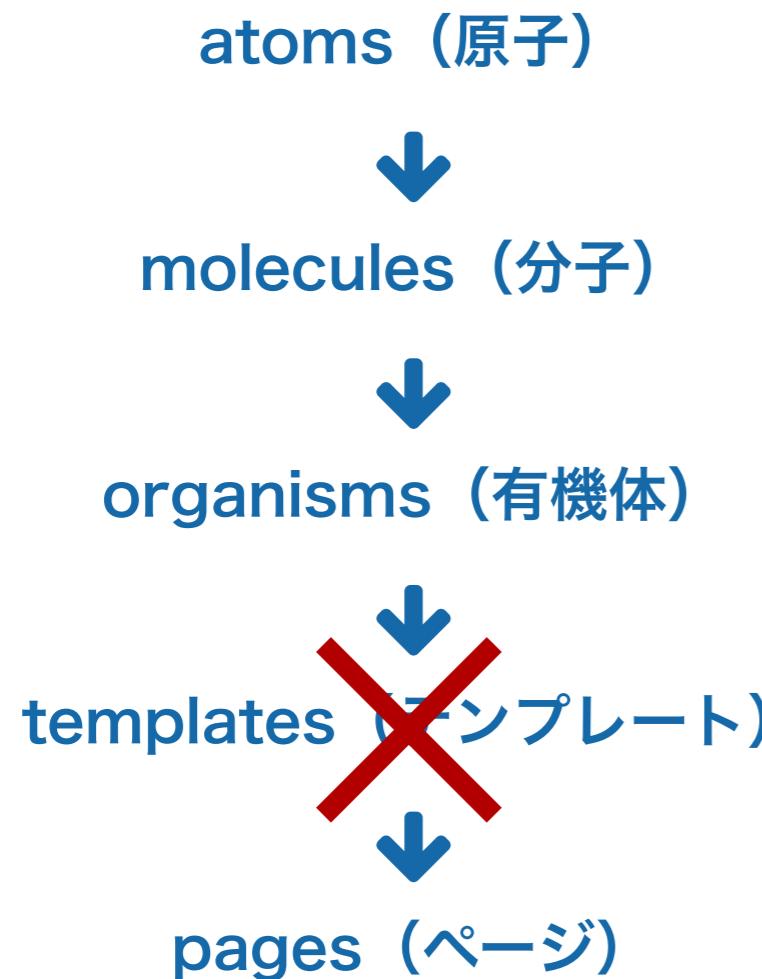


templates (テンプレート)



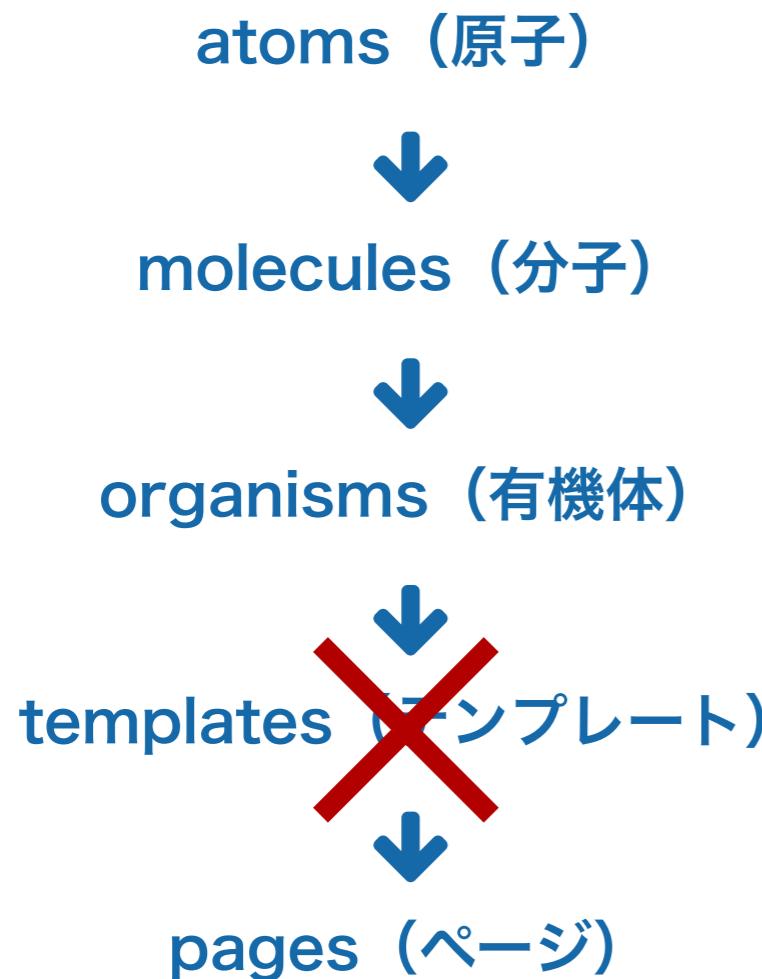
pages (ページ)

# Atomic Design をどう取り入れる？の例



僕の場合は直接システムにマークアップをして、デザインを入れていくので、templates はいらないかな～。

# Atomic Design をどう取り入れる？の例



parts (atoms)  
block (molecules)  
component (organisms)

…と段階を作ると、  
コンポーネントがもっとDRYに作れるんじゃないか  
なー、とか考えてます。

# Atomic Design をどう取り入れる？の例

-  stylesheets
  -  settings
  -  helpers
  -  components
    -  atoms (原子)
    -  molecules (分子)
    -  organisms (有機体)
  -  pages (ページ)
  -  application.sass

さっき挙げた blocks ~ pages の間を  
もっと細分化してみようかな

4

## リセットについて

# リセット

さっき出てきたやつ



ブラウザ間の表示差異をなくしてデザインを入れる前のベースを作るファイル。  
主に下記のいずれかが使われてます。

reset.css

normalize.css

sanitize.css

# リセット

さっき出てきたやつ



ブラウザ間の表示差異をなくしてデザインを入れる前のベースを作るファイル。  
主に下記のいずれかが使われてます。

reset.css

normalize.css

sanitize.css

どれを使うべき？

# reset.css

それぞれのブラウザは、デフォルトスタイルを独自に持っていて、それらを削除して、まっさらな状態にしてからスタイルをあてていこう、っていうやり方の場合、独自のデフォルトスタイルを削除するためのCSSが必要になる。このCSSが `reset.css` と呼ばれている。

# reset.css

reset.css と一口に言っても、いくつかあって、それぞれ微妙に違ったりします。

# reset.css

reset.css と一口に言っても、いくつかあって、それぞれ微妙に違ったりします。

The screenshot shows the homepage of [CSS Reset](http://cssreset.com/). At the top, there's a dark header with the site name, social sharing links (Twitter, Google+, Facebook), and navigation links for "CSS Resets" and "CSS Tutorials". The main content area features a heading "2016's most popular CSS Reset scripts, all in one place" followed by a sub-instruction "Click 'Get Code' to copy/paste the full or minified version, or check out the documentation." Below this, four reset scripts are listed in a grid:

- Eric Meyer's "Reset CSS" 2.0** (50,044) [Get The Code](#)
- HTML5 Doctor CSS Reset** (30,520) [Get The Code](#)
- Yahoo! (YUI 3) Reset CSS** (11,910) [Get The Code](#)
- Universal Selector '\*' Reset** (9,243) [Get The Code](#)

A large, bold text overlay "Eric Meyer's ‘Reset CSS’ が一番使われてる" is centered over the first item. At the bottom, there's a call to action for a free eBook: "Free eBook: CSS Programming Cookbook" with a "Free Download" button.

<http://cssreset.com/>

# reset.css

ちなみに、compass に入ってる reset.css も Eric Meyer's “Reset CSS” ベース

Compass / compass

Code Issues 361 Pull requests 42 Wiki Pulse Graphs

Branch: stable → compass / core / stylesheets / compass / reset / \_utilities.scss Find file Copy path

chriseppstein Move the CLI and website into their own folders. f7a21d3 on 6 Nov 2013

1 contributor

143 lines (130 sloc) | 4.4 KB Raw Blame History

```
1 // Based on [Eric Meyer's reset 2.0](http://meyerweb.com/eric/tools/css/reset/index.html)
2 // Global reset rules.
3 // For more specific resets, use the reset mixins provided below
4 @ mixin global-reset {
5   html, body, div, span, applet, object, iframe,
6   h1, h2, h3, h4, h5, h6, p, blockquote, pre,
7   a, abbr, acronym, address, big, cite, code,
8   del, dfn, em, img, ins, kbd, q, s, samp,
9   small, strike, strong, sub, sup, tt, var,
10  b, u, i, center,
11  dl, dt, dd, ol, ul, li,
12  fieldset, form, label, legend,
13  table, caption, tbody, tfoot, thead, tr, th, td,
14  article, aside, canvas, details, embed,
15  figure, figcaption, footer, header, hgroup,
16  menu, nav, output, ruby, section, summary,
17  time, mark, audio, video {
18    @include reset-box-model;
19    @include reset-font; }
20  // Unlike Eric's original reset, we reset the html element to be compatible
21  // with the vertical rhythm mixins.
22  html {
23    @include reset-body; }
24  ol, ul {
25    @include reset-list-style; }
26  table {
27    @include reset-table; }
```

# reset.css

## メリット

- 邪魔なスタイルは消してあるので  
デザインが入れやすい。

## デメリット

- スタイルをあてないと、なんだかわからない。  
開発しづらい。

The screenshot displays a series of form input fields, each labeled with its type and a descriptive example. The fields include:

- File input:** A file selection input with placeholder text "選択されていません".
- Textarea:** A text area containing the placeholder text "Textarea text".
- Color input:** A color picker showing black.
- Number input:** An input field with the value "5".
- Range input:** A range slider with a value of "0" and a label "年 / 月 / 日".
- Month input:** A date input field showing "----年--月".
- Week input:** A date input field showing "----年第--週".
- Time input:** A time input field showing "--:--".
- Datetime input:** A date and time input field.
- Datetime-local input:** A date and time input field showing "年 / 月 / 日 --:--".
- Submit form:** A submit button.
- form fields disabled:** A section showing disabled inputs. It includes:
  - Disabled Form Fields:** A disabled text input with placeholder "Disabled text input".
  - Disabled select field:** A disabled select dropdown with option "Option 01".
  - Disabled file input:** A disabled file input with placeholder "選択されていません".
  - Disabled radio input:** A disabled radio button.
  - Disabled checkbox input:** A disabled checkbox.
  - Disabled color input:** A disabled color picker showing black.
  - Disabled range input:** A disabled range slider.
  - Disabled number input:** A disabled number input with value "5".
- Disabled textarea:** A disabled text area with placeholder "Textarea text".
- form fields readonly:** A section showing readonly inputs. It includes:
  - Readonly Form Fields:** A readonly text input with placeholder " Readonly text input".
  - Readonly color input:** A readonly color picker showing black.
  - Readonly range input:** A readonly range slider.
  - Readonly number input:** A readonly number input with value "5".
- Readonly textarea:** A readonly text area with placeholder "Textarea text".

# Normalize.css

Normalize.css はブラウザの持つ全てのデフォルトのスタイルを削除する reset.css と違って、有用なデフォルトのスタイルはそのまま使うってやつ。

## What does it do?

- Preserves useful defaults, unlike many CSS resets.
- Normalizes styles for a wide range of elements.
- Corrects bugs and common browser inconsistencies.
- Improves usability with subtle improvements.
- Explains what code does using detailed comments.
- CSSリセットは違って、便利なデフォルトスタイルは保持。
- 多くの要素のスタイルを正常化。
- ブラウザごとの差異をなくし、バグを修正。
- 少しの改善で使いやすさを向上。
- 各コードに詳細なコメントを用意。

# Normalize.css

色んなフレームワークで使われてる。

bootstrap

[twbs / bootstrap](#)

Code Issues Pull requests Pulse Graphs

Branch: master · [bootstrap / less / normalize.less](#)

f2ee28f on 2 contributors

425 lines (339 sloc) | 7.38 KB

```
1 /*! normalize.css v3.0.3 | MIT License | github.com/necolas/normalize.css */
2 
3 /**
4  * 1. Set default font family to sans-serif.
5  * 2. Prevent iOS and IE text size adjust after device orientation change,
6  *    without disabling user zoom.
7 */
8
9 html {
10   font-family: sans-serif; // 1
11   -ms-text-size-adjust: 100%; // 2
12   -webkit-text-size-adjust: 100%; // 3
13 }
14
15 /**
16  * Remove default margin on the body element.
17 */
18
```

[Raw](#) [Blame](#) [History](#)

[yahoo / pure](#)

Code Issues Pull requests Pulse Graphs

Tree: 394ee18a1c · [pure / bower.json](#)

kkirsche Remove moot 'version' property from bower.json

5 contributors

9 lines (8 sloc) | 184 Bytes

```
1 {
2   "name": "pure",
3   "description": "Use Pure's ridiculously tiny CSS to start any web project.",
4   "main": "build/pure.css",
5   "devDependencies": {
6     "normalize-css": "^3.0"
7   }
8 }
```

Pure

html5-boilerplate

[h5bp / html5-boilerplate](#)

Code Issues Pull requests Wiki Pulse Graphs

Branch: master · [html5-boilerplate / dist / css / normalize.css](#)

alrra Update Normalize.css to 'v3.0.3'. 735b9ad on 31 Mar 2015

2 contributors

425 lines (339 sloc) | 7.53 KB

```
1 /*! normalize.css v3.0.3 | MIT License | github.com/necolas/normalize.css */
2
3 /**
4  * 1. Set default font family to sans-serif.
5  * 2. Prevent iOS and IE text size adjust after device orientation change,
6  *    without disabling user zoom.
7 */
8
9 html {
10   font-family: sans-serif; // 1
11   -ms-text-size-adjust: 100%; // 2
12   -webkit-text-size-adjust: 100%; // 3
13 }
```

[Raw](#) [Blame](#) [History](#)

[Find file](#) [Copy path](#)

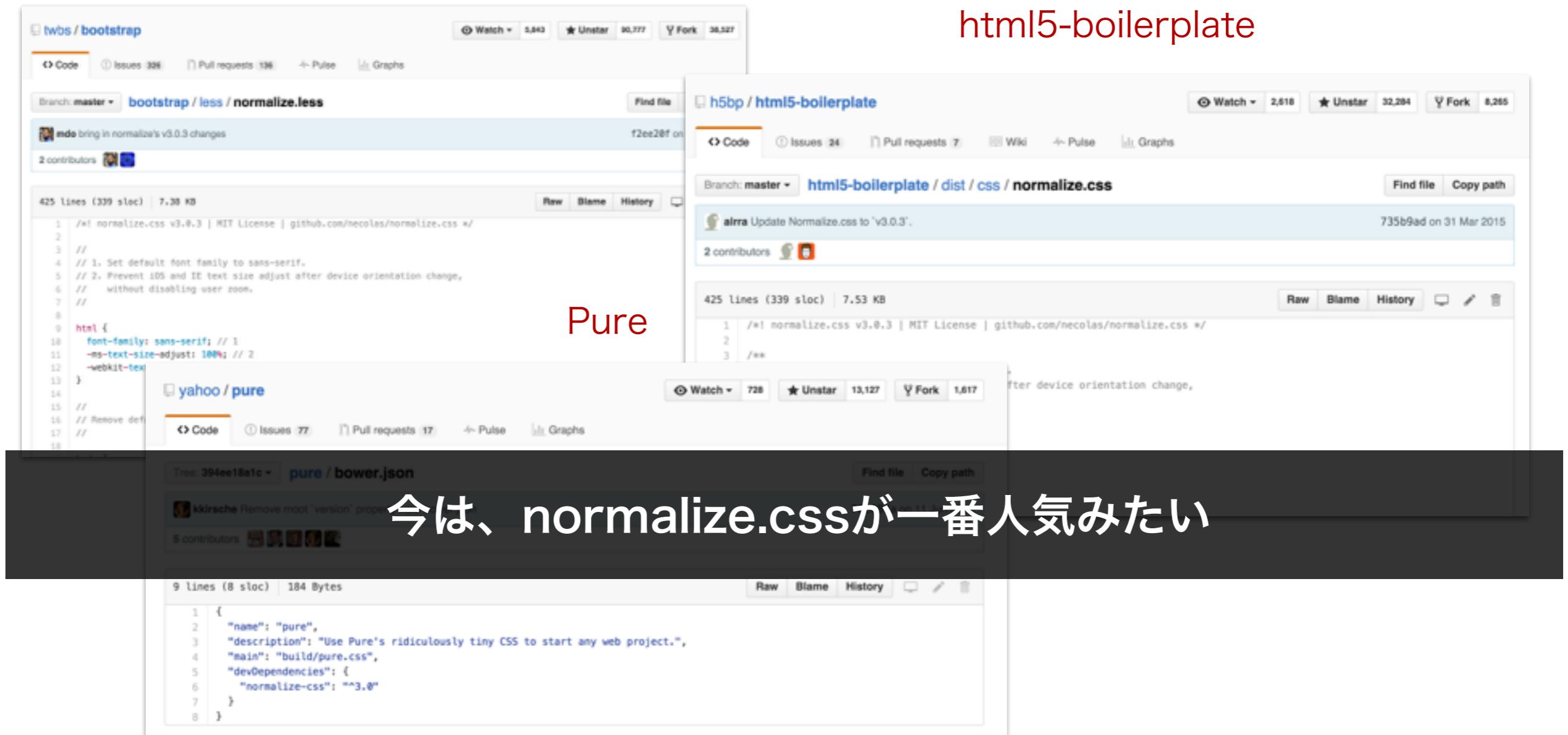
735b9ad on 31 Mar 2015

after device orientation change,

# Normalize.css

色んなフレームワークで使われてる。

## bootstrap



# Normalize.css

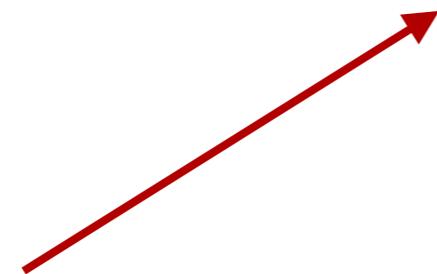
## メリット

- 元々あるスタイルを消してから足す  
という手間がなくなってエコな感じ。

## デメリット

- 不要なスタイルを毎回消すの面倒。

hタグ、上下にマージン付いてるの?  
邪魔だな～



## Base HTML

### address

#### Example

*Company Name, Inc.  
811 7th Ave, Suite 700  
Manhattan, NY 10019  
USA*

### blockquote

#### Example

*Some sort of famous witty quote marked up with a  
and a child  
element.*

*Even better philosophical quote marked up with just a  
element.*

# sanitize.css

デザインをしやすいように、欲しいところは残し、リセットしたいところはリセット。  
それに追加して、デザインを入れやすいようにデフォルトの状態を調整。

# sanitize.css

デザインをしやすいように、欲しいところは残し、リセットしたいところはリセット。  
それに追加して、デザインを入れやすいようにデフォルトの状態を調整。

reset.css を使っても、Normalize.css を使っても、もっとデザインを入れやすいように、設定を消したり、独自の設定を追加することが多いのですが、だんだん sanitize.css っぽくなってくる。

# sanitize.css

例えば、

**box-sizing: border-box** が始めから設定されてたり

```
:root {  
  box-sizing: border-box;  
}
```

code,

kbd,

pre,

samp {

font-family: monospace, monospace;

}

**code** 関連は **font-family** には **monospace** が指定さ  
れてたり

# sanitize.css

例えば、

```
*,  
::before,  
::after {  
    border-style: solid;  
    border-width: 0;  
}
```

**border-style** のデフォルトが **solid** に設定されてるので、  
**border-style** の指定を忘れても **border** はちゃんと表  
示されるようになってたり。

# sanitize.css

例えば、

```
*,  
::before,  
::after {  
    border-style: solid;  
    border-width: 0;  
}
```

**border-style** のデフォルトが **solid** に設定されてるので、  
**border-style** の指定を忘れても **border** はちゃんと表  
示されるようになってたり。

こんな設定がいくつか含まれてます

参考 : <http://coliss.com/articles/build-websites/operation/css/about-sanitize-css.html>

# sanitize.css

## メリット

- ・ デザインを入れるにあたって、必要なものは入ってるし、いらないものはなくなってるし、デザインを入れやすい。

## デメリット

- ・ 場合によっては reset 以上にスタイルをあてないと、なんだかわからない。

デザインを入れないと開発しづらい。

フォームに見えない！

Textarea  
Color input ■  
Number input 5  
Range input 0 年 /月/日  
Month input 一年--月  
Week input 第一週  
Time input --:--  
Datetime input  
Datetime-local input 年 /月/日 --:--  
Submit form  
**form fields disabled**

**Example**  
Disabled Form Fields  
Disabled text input Disabled text input  
Disabled select field Option 01  
Disabled file input ファイルを選択 選択されていません  
○ Disabled radio input  
□ Disabled checkbox input  
Disabled color input ■  
Disabled range input  
Disabled number input 5  
Textarea text

Disabled textarea

# どれを使うべき？

reset.css

normalize.css

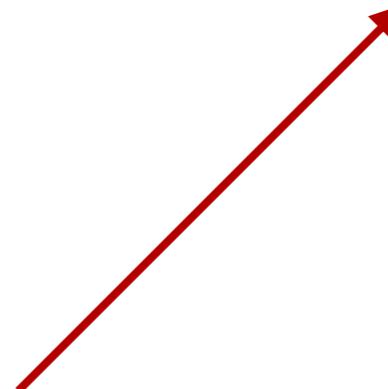
sanitize.css

# どれを使うべき？

reset.css

normalize.css

sanitize.css



個人的には sanitize.css に一票！

# 見比べてみよう

今回用意したハンズオン用のRailsアプリを立ち上げると、トップページに「リセット」というナビゲーションが表示されます。

同じHTMLに下記のcssを読ませたページを作りました。見比べてみてください。

- default（何もcssを読ませないブラウザのデフォルト）
- reset.css
- normalize.css
- sanitize.css

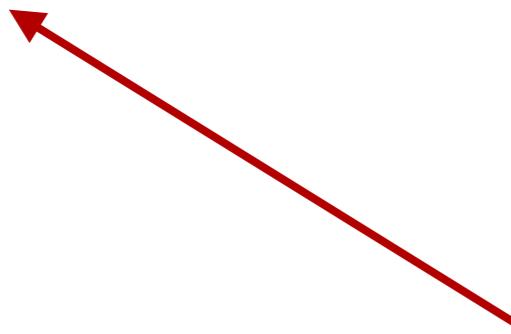
5

## 今回のマークアップについて

# 今回のマークアップについて

今回はCSSがテーマなので、マークアップを丁寧にやってるとCSSのハンズオンの時間がなくなってしまうので、今回は例として町田のマークアップを紹介しておきます。

- OOCSS
- BEM
- SMACSS
- MCSS
- FLOCSS



色々あるのでググったりしてください

# 町田のマークアップ

僕はちょっと変則的な BEM を使ってます。  
まずは、簡単にBEMをおさらいしてみます。

# BEMとは？

- ・ BEMとは、Block（ブロック） 、 Element（要素） 、 Modifier（修飾） の略。

# BEMとは？

- ・ BEMとは、Block（ブロック）、Element（要素）、Modifier（修飾）の略。

って何？

# BEMとは？

- ・ BEMとは、Block（ブロック）、Element（要素）、Modifier（修飾）の略。

って何？

今回だけいつもと違う見た目にする、ってときに使う。

例えば、

- ・ ログインした状態だとヘッダのナビのサイズが変わる
- ・ 今はクリスマス時期なのでロゴの横にサンタの絵がくっついてる
- ・ トップページだけヘッダのサイズが大きい
- ・ サイドナビゲーションメニューの「使い方」、「よくある質問」、「お申し込み」のうち、「お申し込み」だけ背景色が赤い

# BEMとは？

- ・ BEMとは、Block（ブロック） 、 Element（要素） 、 Modifier（修飾） の略。

**.header \_\_ logo-image -- christmas**

**Block**

**Element**

**Modifier**

header という block の中の logo-image という要素の christmas 版

# BEMとは？

- ・ BEMとは、Block（ブロック） 、 Element（要素） 、 Modifier（修飾） の略。

CSS Tutorial

リセット

- default
- reset.css
- Normalize.css
- sanitize.css

コンポーネント

- global\_nav
- alert
- headding
- comment
- articles

ブロック “home-nav”

# BEMとは？

- ・ BEMとは、Block（ブロック） 、 Element（要素） 、 Modifier（修飾） の略。

The screenshot shows a user interface for a CSS tutorial. On the left, there's a sidebar with a teal header labeled "CSS Tutorial". Below it is a red button labeled "リセット" with a white border. To the right of the button is a list of CSS files: "default", "reset.css", "Normalize.css", and "sanitize.css". On the right, there's a larger panel with a teal header labeled "コンポーネント". Inside this panel, there's a list of components: "global\_nav", "alert", "headding", "comment", and "articles". A red arrow points from the text "要素 “title”" to the word "title" in the "headding" component list.

CSS Tutorial

リセット

default  
reset.css  
Normalize.css  
sanitize.css

コンポーネント

global\_nav  
alert  
headding  
comment  
articles

要素 “title”

# BEMとは？

- ・ BEMとは、Block（ブロック） 、 Element（要素） 、 Modifier（修飾） の略。

CSS Tutorial

リセット

default

reset.css

Normalize.css

sanitize.css

コンポーネント

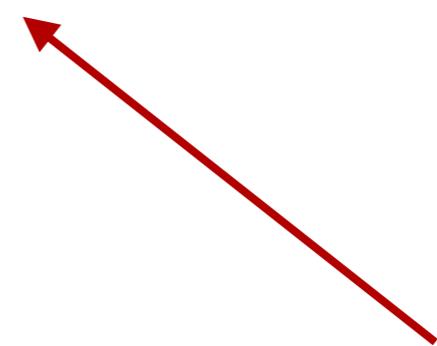
global\_nav

alert

headding

comment

articles



修飾 “reset”

(今回の場合、背景色が赤いことが修飾)

# BEMとは？

[Block]\_\_[Element]--[Modifier]

The screenshot shows a user interface for a CSS tutorial. At the top, there is a green header bar with the text "CSS Turtial". Below it is a red navigation bar with the text "リセット" (Reset) in white. To the right of this bar is a red arrow pointing towards the text ".home-nav\_\_title—reset". Below the red bar is a vertical sidebar with the following items:

- default
- reset.css
- Normalize.css
- sanitize.css

At the bottom, there is another green header bar with the text "コンポーネント" (Components). Below it is a white sidebar with the following items:

- global\_nav
- alert
- headding
- comment
- articles

.home-nav\_\_title—reset

# BEMとは？

[Block]\_\_[Element]--[Modifier]

The screenshot shows a user interface for a CSS tutorial. At the top, there's a teal header bar with the text "CSS Turtial". Below it is a red header bar with the text "リセット". Underneath the red bar is a white sidebar with a red border containing the following list:

- default
- reset.css
- Normalize.css
- sanitize.css

Below this is another teal header bar with the text "コンポーネント". Underneath it is a white sidebar containing the following list:

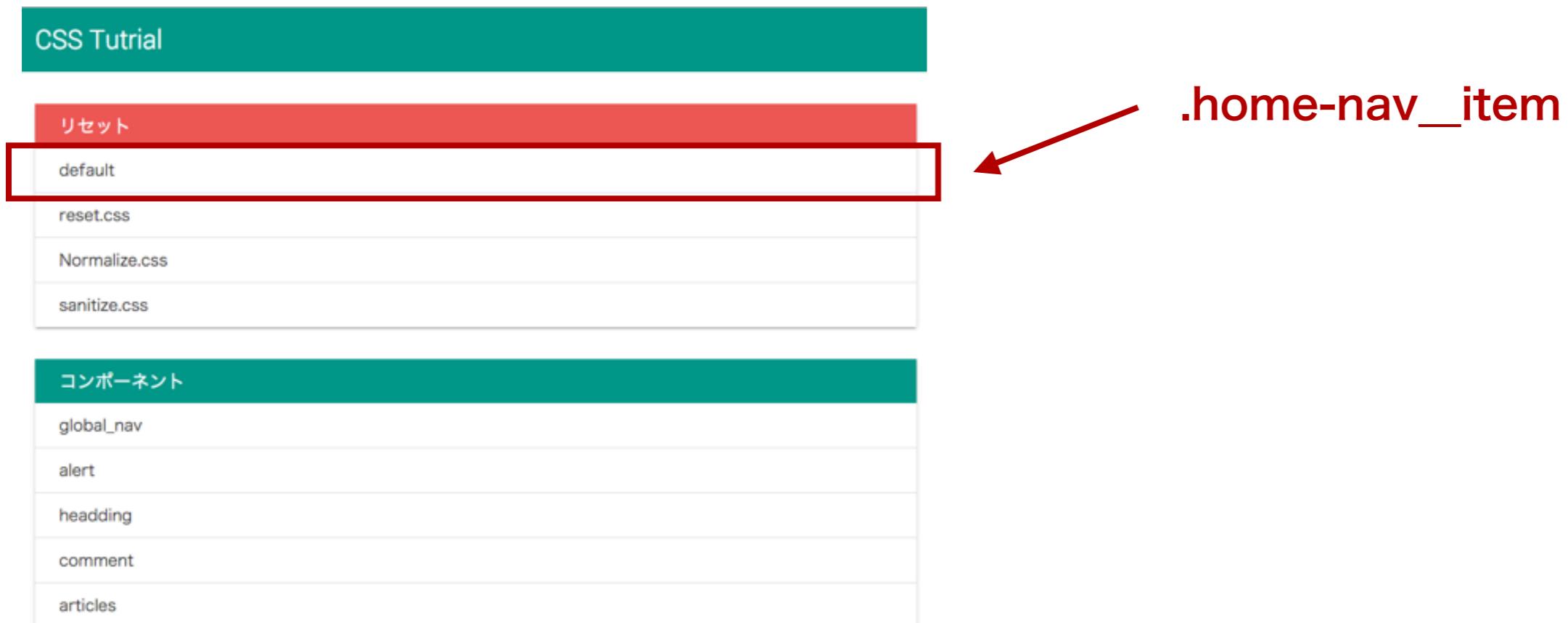
- global\_nav
- alert
- headding
- comment
- articles

.home-nav\_items

(BEだけ、 BMだけってこともある)

# BEMとは？

[Block]\_\_[Element]--[Modifier]



# BEMとは？

[Block]\_\_[Element]--[Modifier]

CSS Turiail

リセット

- default
- reset.css
- Normalize.css
- sanitize.css

コンポーネント

- global\_nav
- alert
- headding
- comment
- articles

.home-nav\_\_item-link

# BEMとは？

[Block]\_\_[Element]--[Modifier]



# BEMとは？

僕の考えでは、まあ、書き方はどうでもよくて、大事なのは考え方の方で、Blockを明確にしてるところが好き。

僕の解釈は、特定の要素はどのBlockに属しているものかを明確にするだけで、BEMの [Block]\_\_[Element] は階層を表しているわけではない、としています。

「Block = コンポーネント」として、コンポーネント単位にHTML、CSS を組むので、CSS を一箇所をいじったら予期しない場所が壊れた、なんてことが起きることが減ります。

ページを中心としたマークアップではなく、コンポーネント中心で組んでいくやり方と相性がいい！

# BEMとは？

僕の考えでは、まあ、書き方はどうでもよくて、大事なのは考え方の方で、Blockを明確にしてるところが好き。

僕の解釈は、特定の要素はどのBlockに属しているものかを明確にするだけで、BEMの [Block]\_\_[Element] は階層を表しているわけではない、としています。

僕の場合、どこからどこまでをBlockとするかが、まだきちんと説明できる  
「Block = コンポーネント」として、コンポーネント単位にHTML、CSSを組むので、CSS  
を一箇所をいじったら予期しない場所が壊れた、なんとか起きることが減ります。  
ほど明確なルールが出来てないので、そこは課題です。  
AtomicDesignを取り入れたらもう少しその辺が明確になるかも…

ページを中心としたマークアップではなく、コンポーネント中心で組んでいくやり方と相性  
がいい！

# 町田アレンジ

基本的には BEM を使っているのですが、Modifier の書き方を少しアレンジしています。

# 町田アレンジ

基本的には BEM を使っているのですが、Modifier の書き方を少しアレンジしています。

```
%nav.header-nav
  %ul.header-nav__items
    %li.header-nav__item
      = link_to root_path, class: "header-nav__item is-current"
    %li.header-nav__item
      = link_to users_path, class: "header-nav__item"
    %li.header-nav__item
      = link_to stories_path, class: "header-nav__item"
```

Modifier を複数 class にして、**is** というプリフィックスを付けてます（smaccsのstateのやり方をいただいた）。



# 町田アレンジ

.header-nav\_\_item のスタイルを継承して、.is-current の場合に必要なスタイルだけ追加、上書き。

```
.header-nav__item-link  
  display: block  
  text-decoration: none  
  background-color: blue  
&.is-current  
  background-color: red  
  font-weight: bold
```

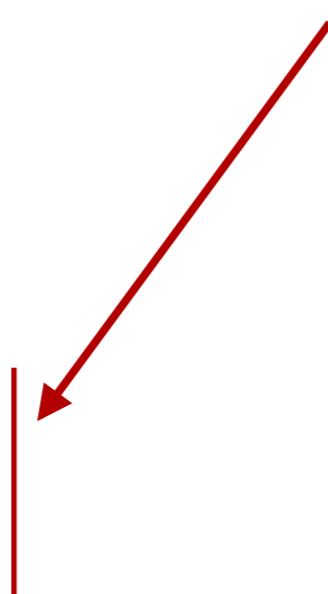
# 町田アレンジ

.header-nav\_\_item のスタイルを継承して、.is-current の場合に必要なスタイルだけ追加、上書き。

```
.header-nav__item-link  
display: block  
text-decoration: none  
background-color: blue  
&.is-current  
background-color: red  
font-weight: bold
```

.header-nav\_\_item-link に追加 class として、  
.is-current が付いてる場合に必要なスタイルだけ  
追加、上書き。

```
.header-nav__item-link  
display: block  
text-decoration: none  
background-color: blue  
.header-nav__item-link.is-current  
background-color: red  
font-weight: bold
```



# 町田アレンジ

まあ、プリフィクスとかはどうでもいいんですが、modifier は別クラスにしたほうが使いやすいです。別 class ではない場合...

# 町田アレンジ

まあ、プリフィクスとかはどうでもいいんですが、modifier は別クラスにしたほうが使いやすいです。別 class ではない場合…

```
.header-nav__item-link  
  display: block  
  text-decoration: none  
  background-color: blue  
&--current  
  background-color: red  
  font-weight: bold
```

# 町田アレンジ

まあ、プリフィクスとかはどうでもいいんですが、modifier は別クラスにしたほうが使いやすいです。別 class ではない場合…

```
.header-nav__item-link  
  display: block  
  text-decoration: none  
  background-color: blue  
&--current|  
  background-color: red  
  font-weight: bold
```

sass ではこう書くと、  
**.header-nav\_\_item-link--current**  
という class のセレクタが作れる。

```
.header-nav__item-link  
  display: block  
  text-decoration: none  
  background-color: blue  
.header-nav__item-link--current  
  background-color: red  
  font-weight: bold
```

class 名は継承がされてるんだけど、  
style は継承されない。

# 町田アレンジ

まあ、プリフィクスとかはどうでもいいんですが、modifier は別クラスにしたほうが使いやすいです。別 class ではない場合…

```
.header-nav__item-link  
  display: block  
  text-decoration: none  
  background-color: blue  
&--current|  
  background-color: red  
  font-weight: bold
```

sass ではこう書くと、  
**.header-nav\_\_item-link--current**  
という class のセレクタが作れる。

```
.header-nav__item-link  
  display: block  
  text-decoration: none  
  background-color: blue  
.header-nav__item-link--current  
  @include .header-nav__item-link  
  background-color: red  
  font-weight: bold
```

なので、includeしないといけない

# 町田アレンジ

まあ、プリフィクスとかはどうでもいいんですが、modifier は別クラスにしたほうが使いやすいです。別 class ではない場合…

```
.header-nav__item-link  
  display: block  
  text-decoration: none  
  background-color: blue  
&--current|  
  background-color: red  
  font-weight: bold
```

sass ではこう書くと、  
.header-nav\_\_item-link--current  
という class のセレクタが作れる。

```
.header-nav__item-link  
  display: block  
  text-decoration: none  
  background-color: blue  
.header-nav__item-link--current  
  @include .header-nav__item-link
```

これが面倒なのでModifierは別 class にしています。

```
background-color: red  
font-weight: bold
```

なので、includeしないといけない

# 町田アレンジ

ちなみに、`&--current` と書くのとかもあり。

```
.header-nav__item-link  
display: block  
text-decoration: none  
background-color: blue  
&.--current  
background-color: red  
font-weight: bold
```

`.header-nav__item-link` に追加 class として、  
`--current` が付いてる場合に必要なスタイルだけ  
追加、上書き。

```
.header-nav__item-link  
display: block  
text-decoration: none  
background-color: blue  
.header-nav__item-link.--current  
background-color: red  
font-weight: bold
```

# 突然ですが、質問コーナー

# 突然ですが、質問コーナー

デザイン用のhtml（css含む）の作成した人と、アプリへ組み込む人は別の人であることを想定した質問です。

一言で言うと「たくさんのcssの設定がある場合、どのように整理していますか？」です。

cssの設定がたくさんになってくると、どの名前がどのレイアウトに対応しているのかわからなくなってしまいます。

アプリ側で少しデザインを追加したい、変更したい場合に、  
cssを追いかけきれないため、結局自分で独自の設定を追加してしまうことがあります。  
(ある程度はブラウザのデバッグツールで解析していましたが)

cssが増えてくると何かしら手を打たないと破綻しそうです。

何か工夫されていることがあるようでしたら教えてもらいたいです。  
(bootstrapのサイトのように、見た目とcss設定値の対応表があるとわかりやすいと思っています)

# 突然ですが、質問コーナー

これは提案の一つですが、マークアップにBEMを使うなどして、コンポーネント単位でマークアップをして、コンポーネント単位でCSSを組むのはどうでしょう。スタイルガイドも用意すると作業はしやすくなると思います。

また、スタイルガイドがあることでデザインする人も、コンポーネントを意識したデザインをしやすくなると思います。

スタイルガイドジェネレータを使って、スタイルガイドも作ってもらうといいかもしれません。

# ページとコンポーネント

マークアップをするとき、ページ中心にするのではなく、コンポーネントをまず作って、それをページにどう配置するのか、って考えて作るほうが何かと都合がいいです。

- ・ コンポーネントの範囲を明確にすることで、CSSの変更によって影響する場所が明確になります。
- ・ コンポーネントを意識することで、無駄な重複がなくなりHTML、CSSのコードがDRYになります。
- ・ デザイナーもコンポーネントを意識することで、目的は同じだけど意味なく見た目が違うものが作られることはなくなります。
- ・ スタイルガイドが作りやすくなります。

# ページとコンポーネントのイメージ



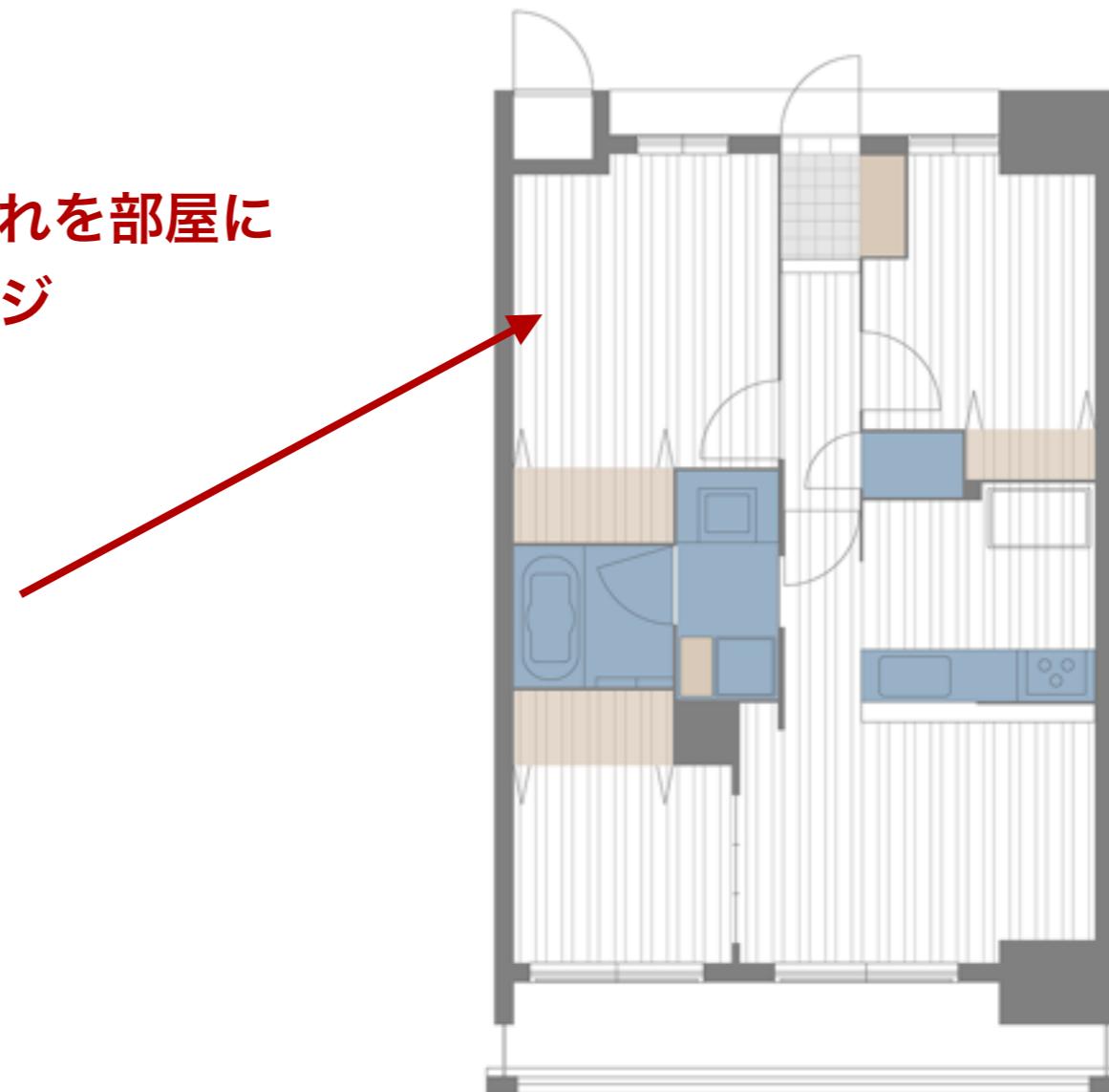
いい感じの部屋を作ろうとするのではなく

# ページとコンポーネントのイメージ

家具をまず作って、それを部屋に  
配置するようなイメージ



コンポーネント



ページ

# ページとコンポーネントのイメージ

スタイルガイドも  
あるといい



REGISSÖR コーヒーテーブル  
¥ 17,990

RIAN サイドテーブル  
¥ 1,999

SVALSTA ネストテーブル2点セット  
¥ 12,990

LACK コーヒーテーブル  
¥ 2,999



LACK コーヒーテーブル  
¥ 5,999

STOCKHOLM コーヒーテーブル  
¥ 24,990

STOCKHOLM コーヒーテーブル  
¥ 29,990

STOCKHOLM ネストテーブル2点セット  
¥ 24,990



LACK サイドテーブル  
¥ 1,499



MALMSTA サイドテーブル  
¥ 12,990



VEJMON サイドテーブル  
¥ 12,990



KLINGSBO コーヒーテーブル  
¥ 14,990

スタイルガイド（コンポーネント集）

# ページとコンポーネントのイメージ

スタイルガイドも  
あるといい



でもこれって、そもそもデザインをする段階でデザイナーがコンポーネントを意識したデザインをしないと、なかなか上手くいかないです。



スタイルガイド（コンポーネント集）

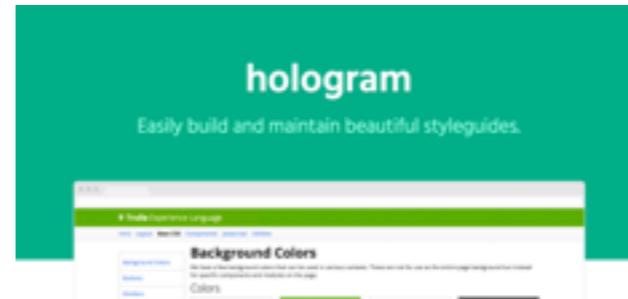
# Style Guide Driven Development

Style Guide Driven Development（スタイルガイド駆動開発）なんて言葉もあるんですが、今年はさらに流行るかも。

# Style Guide Driven Development

Style Guide Driven Development (スタイルガイド駆動開発) なんて言葉もあるんですが、今年はさらに流行るかも。

The screenshot shows the SC5 Style Guide Generator interface. It features a top navigation bar with tabs like Home, Components, Buttons, Typography, and Layout. Below this is a sidebar with sections for Buttons, Panels, Modals, and Tooltips. The main content area displays a live style guide for a component named 'Buttons'. It includes a preview window showing various button styles (Default, Primary, Success, Info, Warning, Danger, Link) and a code editor window showing the corresponding CSS and JavaScript code.



Hologram is a Ruby gem that parses comments in your CSS and turns them into a beautiful styleguide.

Quick start: `gem install hologram`. And then: `hologram init`.

[View on Github](#)



Logos & Images

Upload and manage image files as easy as you would expect. Add captions, organize them in grids. Even PSD files are supported.

[Color Palettes](#)



ジェネレータもたくさんある

CSSを書く前の話ばっかりですいません。

話はここまで、ここから先はハンズオンになります。

# 課題の進めかた

リセット
default
reset.css
Normalize.css
sanitize.css

コンポーネント（課題）
global_nav
alert
articles
comment
heading

コンポーネントの完成キャプチャ
global_nav
alert
articles
comment
heading

こっちを触って  
作業をする

こっちに仕様、見本の画像  
がある

6

global\_nav

# 課題



- ・ ブロックを横に並べるのってどうやるの？
- ・ 文字をブロックのx軸y軸の中央に配置したいんだけどどうやるの？

いじる sass ファイルの場所

/app/assets/stylesheets/components/global-nav/\_global-nav.sass

# 今回の課題



- ・ ブロックを横に並べるのってどうやるの？
- ・ 文字をブロックのx軸y軸の中央に配置したいんだけどどうやるの？

いじる sass ファイルの場所  
/app/assets/stylesheets/components/global-nav/\_global-nav.sass

# ブロックを横に並べる方法 1

\_global-nav-sample-1-A.sass.sample と \_global-nav-sample-1-B.sass.sample を  
参照してください

# ブロックを横に並べる方法 1

\_global-nav-sample-1-A.sass.sample と \_global-nav-sample-1-B.sass.sample を  
参照してください

- ・親に display: table、子どもに display: table-cell を適用する。これでtableタグのようになる。
- ・display: table が適用された親は横幅いっぱいの幅を持たせるため、width: 100% にする。
- ・**A**  
display: table が適用された親に table-layout: fixed を適用すると、display: table-cell が適用された子どもは、中身の文字数などに関わらず、勝手に均等の横幅になる。
- ・**B**  
display: table が適用された親に table-layout: fixed を適用しない場合は、display: table-cell が適用された子どもに width: 20% を適用する。

**A**

```
3 .global-nav__items
4   display: table
5   table-layout: fixed
6   width: 100%
7   .global-nav__item
8     display: table-cell
```

**B**

```
3 .global-nav__items
4   display: table
5   width: 100%
6   .global-nav__item
7     display: table-cell
8     width: 20%
```

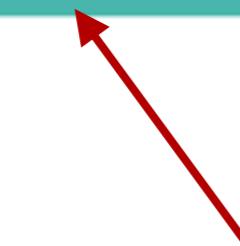
# ブロックを横に並べる方法 2

\_global-nav-sample-2.sass.sample を参照してください

# ブロックを横に並べる方法 2

\_global-nav-sample-2.sass.sample を参照してください

- ・ 横並びにしたい子どもに `display: inline-block` を適用する。
- ・ `display: inline-block` を適用した子どもに `width: 20%` を適用する。すると、均等な横幅で横並びになるかと思いきや…



親の幅に収まりきらず、落ちてしまう

# ブロックを横に並べる方法 2

\_global-nav-sample-2.sass.sample を参照してください

- 親に收まりきらず落ちてしまう問題を解決するために、親に font-size: 0 を適用する。すると、問題は解決されるが、子どもにもfont-size: 0 は適用されてしまうのでちょっと心配

```
3   .global-nav__items
4     font-size: 0
5   .global-nav__item
6     width: 20%
7     display: inline-block
```

横並びになった！



# ブロックを横に並べる方法 2

app/views/application/\_global\_nav\_1.haml を参照してください

- ・親に收まりきらず落ちてしまう問題を解決するために、haml ファイルをいじる。
- ・横並びにしたい要素の後に <> を付ける。

```
1  %nav.global-nav
2    .container
3      %ul.global-nav__items
4        %li.global-nav__item<>
5          = link_to "リンク", "#", class: "global-nav__item-link"
6        %li.global-nav__item<>
7          = link_to "リンク", "#", class: "global-nav__item-link"
8        %li.global-nav__item<>
9          = link_to "リンク", "#", class: "global-nav__item-link"
10       %li.global-nav__item<>
11         = link_to "リンク", "#", class: "global-nav__item-link"
12       %li.global-nav__item<>
13         = link_to "リンク", "#", class: "global-nav__item-link"
```

<> を付ける

# ブロックを横に並べる方法 3

\_global-nav-sample-3.sass.sample を参照してください

# ブロックを横に並べる方法 3

\_global-nav-sample-3.sass.sample を参照してください

- ・ 横並びにしたい子どもに float: left と width: 20% を適用する。これで横並びにはなるが…
- ・ 親の親である .global-nav に適用した background-color: \$teal-300 が下まで降りてこない！



# ブロックを横に並べる方法 3

\_global-nav-sample-3.sass.sample を参照してください

- background-color: \$teal-300 を下まで持っていくたい、.global-nav に clearfix 適用する

```
1 .global-nav
2   background-color: $teal-300
3   +clearfix ←
4 .global-nav__item
5   width: 20%
6   float: left
```

今回はbourbonを使っているので、bourbonに含まれている clearfix の mixin を使用。

bourbon を使ってない場合は、bourbonからパクるといい。

[https://github.com/thoughtbot/bourbon/blob/master/app/assets/stylesheets/addons/\\_clearfix.scss](https://github.com/thoughtbot/bourbon/blob/master/app/assets/stylesheets/addons/_clearfix.scss)

- float が適用された子要素を持つ親は、clear が適用された子要素が出てくるまで下まで降りてこない。でも、親に overflow: hidden を適用すると降りてくる。
- こういった面倒なやつをclearfixはやってくれてる。

# ブロックを横に並べる方法 3

\_global-nav-sample-3.sass.sample を参照してください

- background-color: \$teal-300 を下まで持っていくたい、.global-nav に clearfix 適用する

```
1 .global-nav
2   background-color: $teal-300
3   +clearfix ←
4 .global-nav__item
5   width: 20%
6   float: left
```

今回はbourbonを使っているので、bourbonに含まれている clearfix の mixin を使用。

bourbon を使ってない場合は、bourbonからパクるといい。

[https://github.com/thoughtbot/bourbon/blob/master/app/assets/stylesheets/addons/\\_clearfix.scss](https://github.com/thoughtbot/bourbon/blob/master/app/assets/stylesheets/addons/_clearfix.scss)

- float が適用された子要素を持つ親は、clear が適用された子要素が出てくるまで下まで降りてこない。でも、親に overflow: hidden を適用すると降りてくる。
- こういった面倒なやつをclearfixはやってくれてる。

ま、これは古いやり方なので、あんまり気にしなくていいかも

# ブロックを横に並べる方法 4

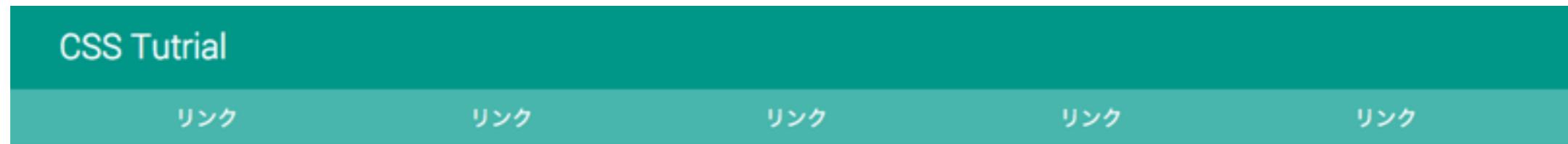
[\\_global-nav-sample-4.sass.sample](#) を参照してください

# ブロックを横に並べる方法 4

\_global-nav-sample-4.sass.sample を参照してください

- ・ 横並びにしたい要素の親に display: flex を適用する。もうこれだけで横並びになる！
- ・ 横並びにしたい要素に flex: 1 を適用する。これだけで均等にそれぞれ 20% の幅になる！

横並びになった！

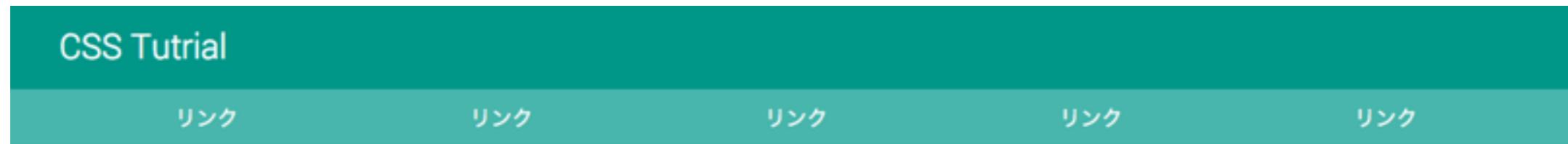


# ブロックを横に並べる方法 4

\_global-nav-sample-4.sass.sample を参照してください

- ・ 横並びにしたい要素の親に display: flex を適用する。もうこれだけで横並びになる！
- ・ 横並びにしたい要素に flex: 1 を適用する。これだけで均等にそれぞれ 20% の幅になる！

横並びになった！



flex ヤバい

# ブロックを横に並べる方法 4

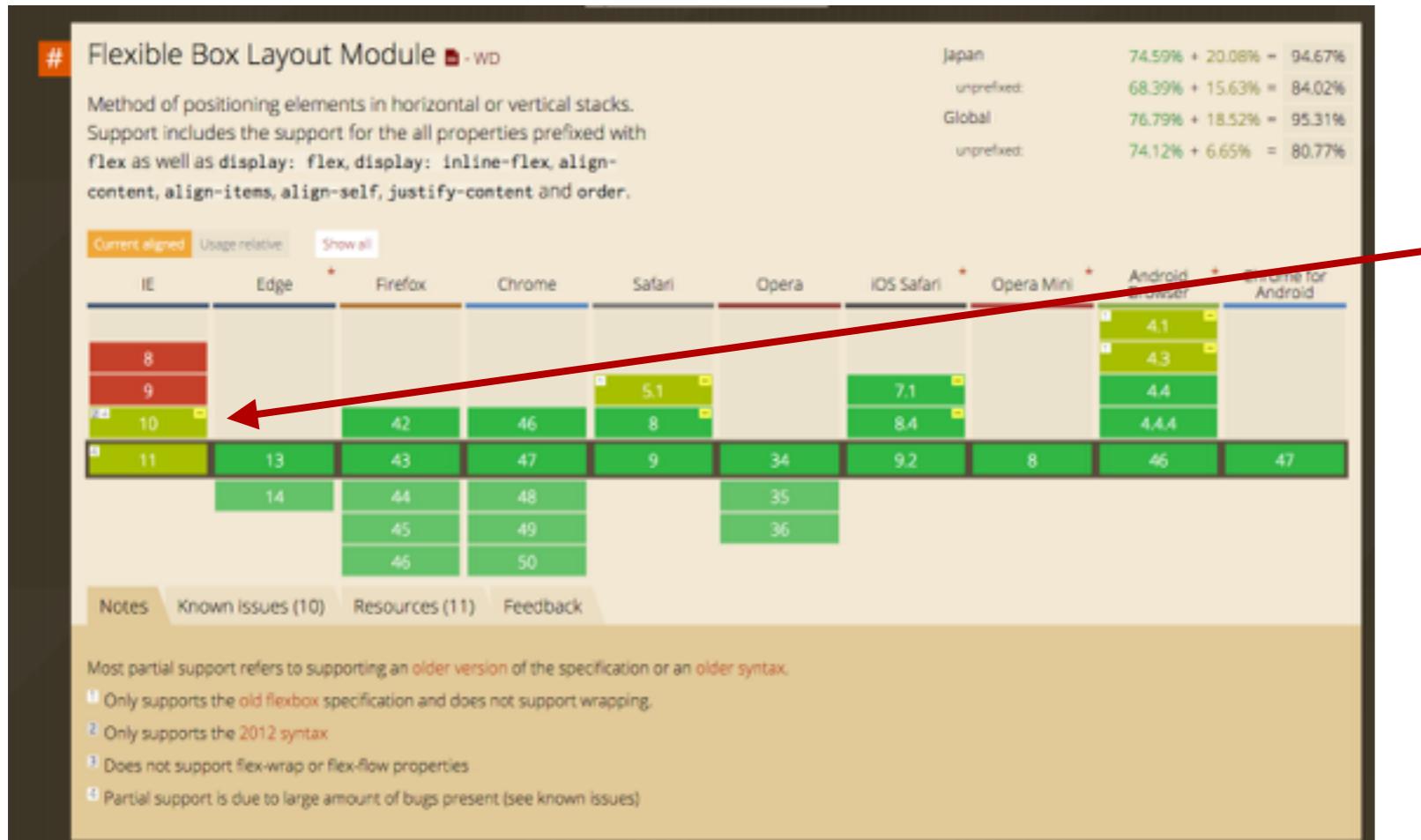
\_global-nav-sample-4.sass.sample を参照してください

- ・ でも、display: flex って ieじゃ使えないとかあるんでしょう？

# ブロックを横に並べる方法 4

\_global-nav-sample-4.sass.sample を参照してください

- でも、display: flex って ieじゃ使えないとかあるんでしょう？

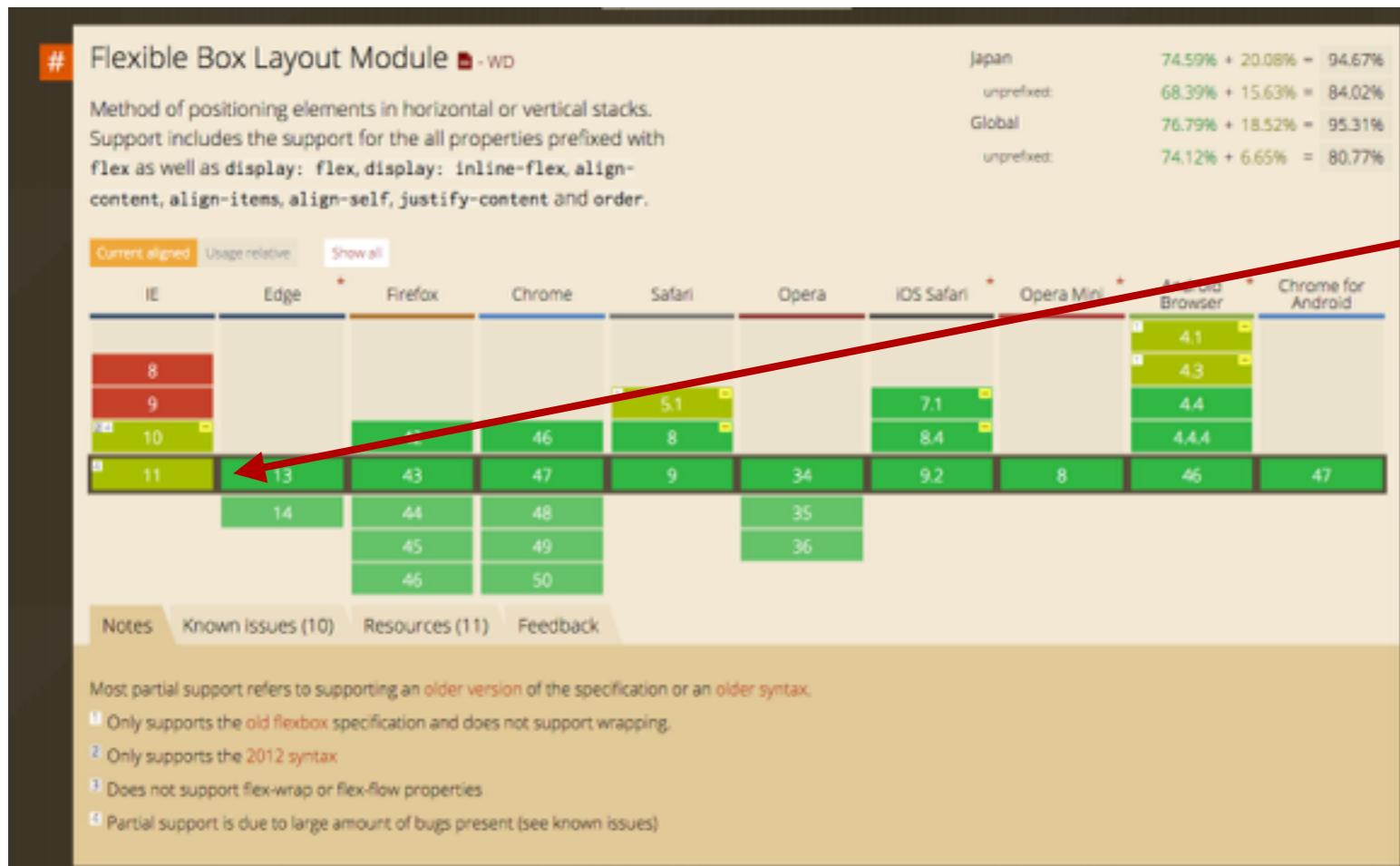


IE 10 では  
2012年の仕様の syntaxで  
書かないとダメで、なおかつバグが  
結構ある。

# ブロックを横に並べる方法 4

\_global-nav-sample-4.sass.sample を参照してください

- でも、display: flex って ieじゃ使えないとかあるんでしょう？

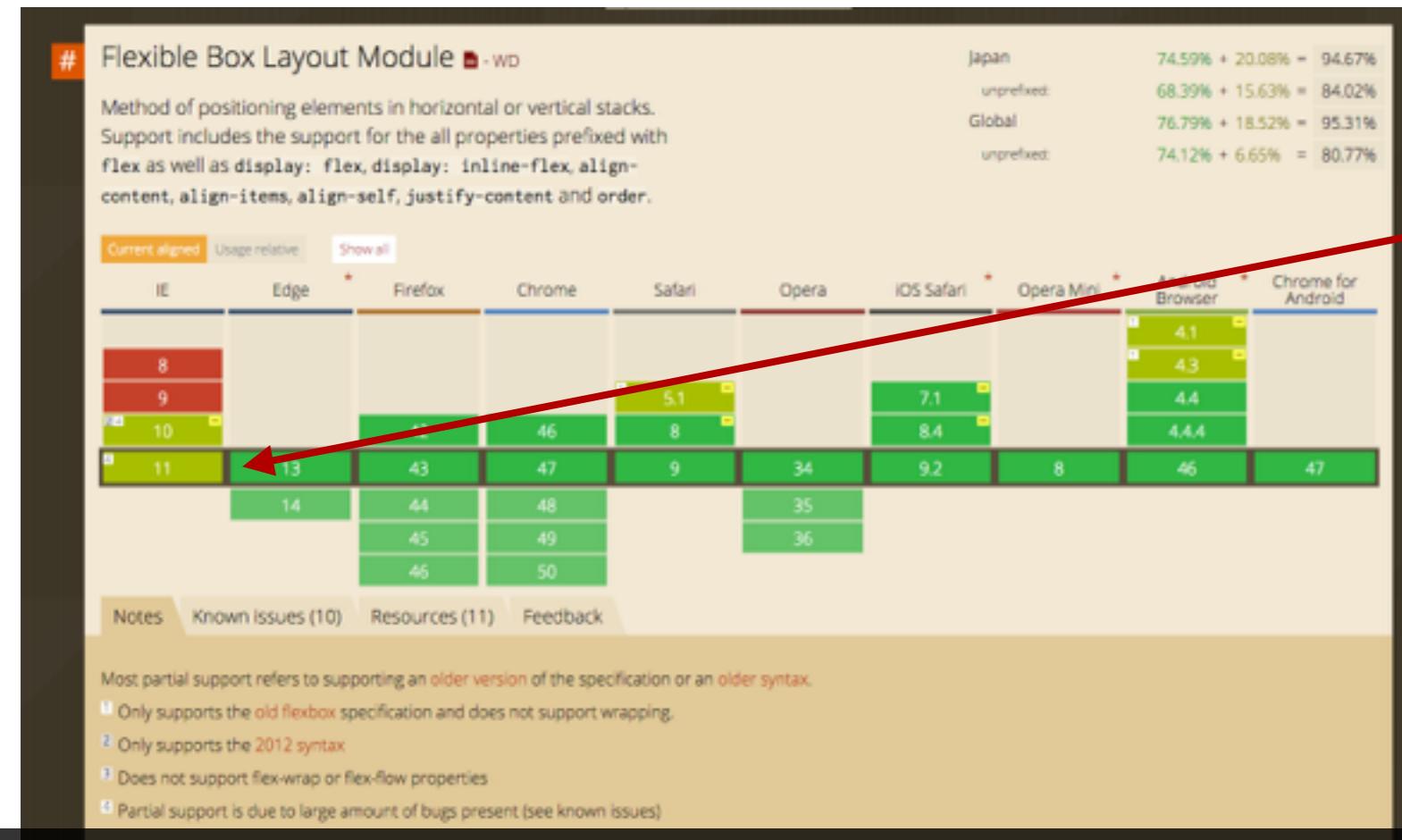


IE 11 ではバグが結構ある。

# ブロックを横に並べる方法 4

\_global-nav-sample-4.sass.sample を参照してください

- でも、display: flex って ieじゃ使えないとかあるんでしょう？



IE 11 ではバグが結構ある。

IE以外はOKなのでflexboxは覚える必要あり！

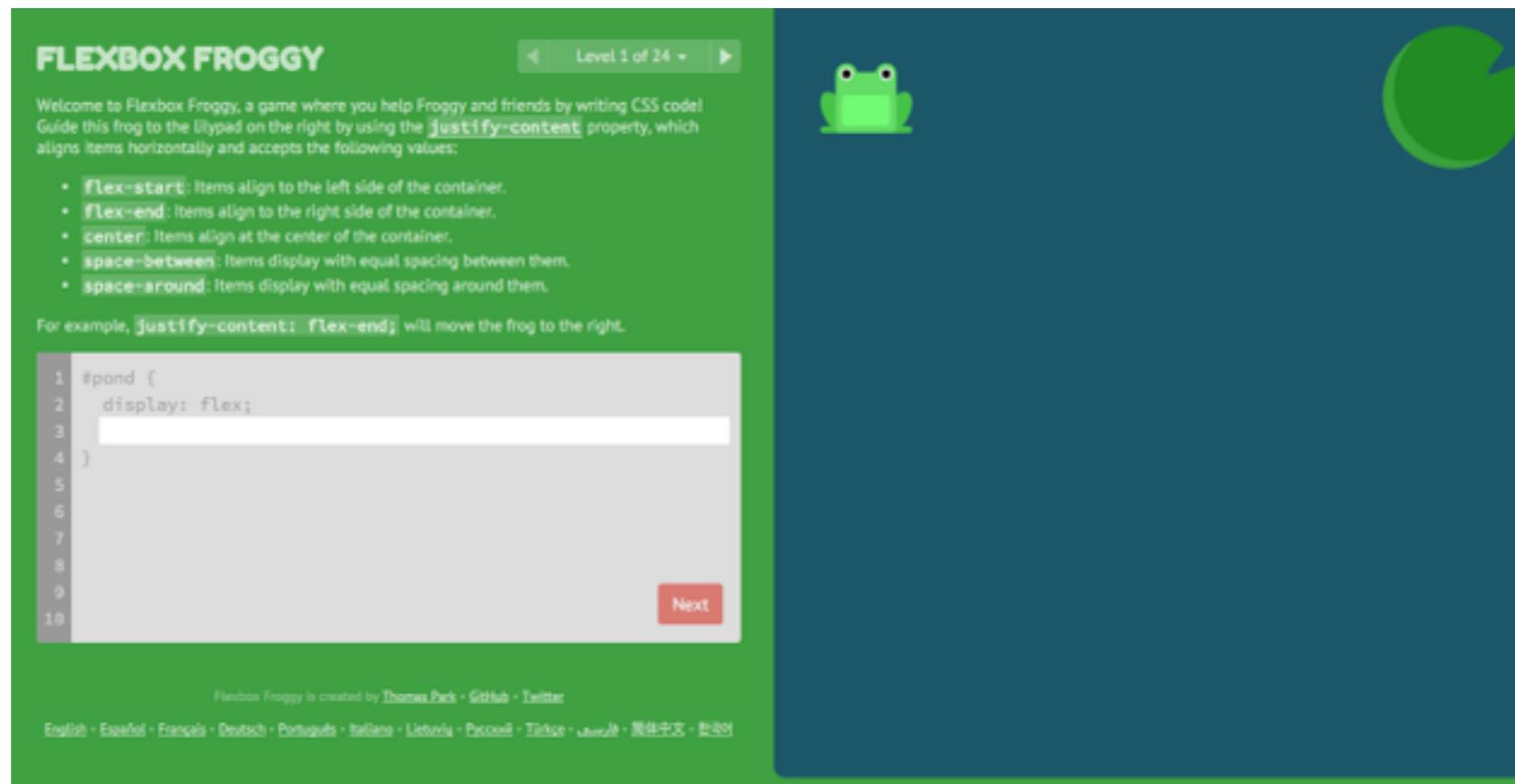
# flexbox

- 今回、ハンズオンでは実装の例に flexbox を使った場合も追加していますが、flexbox の説明まですると時間がなくなってしまうので、flexbox の使い方は各自勉強しておいてください。

## 参考URL

- <http://liginc.co.jp/web/html-css/css/21024>
- <https://scotch.io/tutorials/a-visual-guide-to-css3-flexbox-properties>
- <http://coliss.com/articles/build-websites/operation/css/css3-flexbox-properties-by-scotch.html>
- <http://qiita.com/hashrock/items/939684b9207dbab1d59e>
- <http://philipwalton.github.io/solved-by-flexbox/>
- <http://hashrock.github.io/solved-by-flexbox-ja/>
- <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- <https://blogs.adobe.com/creativestation/web-design-with-css-flexbox-menu>

# flexboxの学習サイト



<http://flexboxfroggy.com/>

flexboxを使ってカエルを葉っぱの上に乗せるゲーム

# 文字をy軸中央に配置する方法

\_global-nav-sample-1～4のいずれかを参照してください

# 文字をy軸中央に配置する方法

\_global-nav-sample-1～4のいずれかを参照してください

- ・ 文字をy軸中央に配置したいリンクにまずは display: block を適用させる。これでinlineの状態からblockになったので、リンクが横幅いっぱいになる。
- ・ 横幅いっぱいになったリンクに text-align: center を適用させる。すると、文字がx軸中央に配置される。
- ・ 次に、リンクの高さは 44px なので、 line-height: 44px を適用させる。

```
9   .global-nav__item-link
10  background-color: $teal-300
11  color: $white
12  text-decoration: none
13  font-size: 16px
14  display: block
15  text-align: center
16  line-height: 44px
```

# 文字をy軸中央に配置する方法

\_global-nav-sample-1～4のいずれかを参照してください

- ・ 文字をy軸中央に配置したいリンクにまずは display: block を適用させる。これでinlineの状態からblockになったので、リンクが横幅いっぱいになる。
- ・ 横幅いっぱいになったリンクに text-align: center を適用させる。すると、文字がx軸中央に配置される。
- ・ 次に、リンクの高さは 44px なので、 line-height: 44px を適用させる。

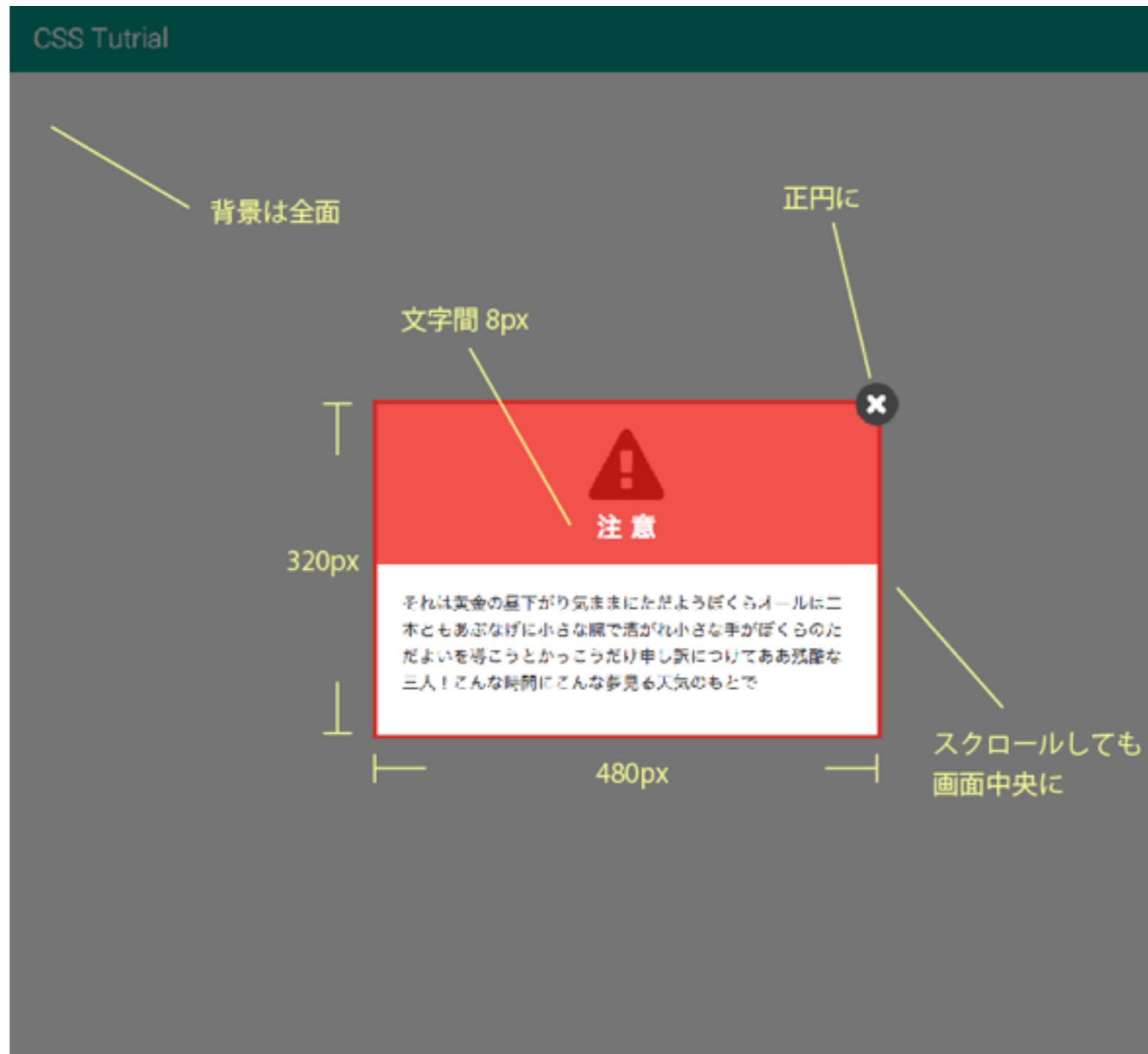
```
9   .global-nav__item-link
10  background-color: $teal-300
11  color: $white
12  text-decoration: none
13  font-size: 16px
14  display: block
15  text-align: center
16  line-height: 44px
```

vertical-align じゃ中央にできない！

7

alert

# 課題



# 課題

- ・ 背景に半透明のスクリーンを配置するには？
- ・ 画面の中央にブロックを表示するのってどうやるの？
- ・ z-indexってどうやって効かすの？
- ・ 真ん中揃えの文字の字間を調整するには？
- ・ アイコンフォントで出したアイコンを正円で囲うには？

いじる sass ファイルの場所

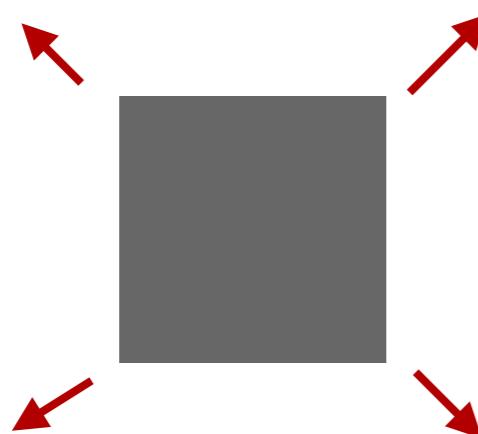
/app/assets/stylesheets/components/alert/\_alert.sass

# 背景に半透明のスクリーンを配置する

[\\_alert-sample-1.sass](#)を参照してください

- .alert-background に position: fixed を適用する。すると、.alert-background はスクロールしても同じ位置に留まるようになる。
- **A**  
.alert-background に left: 0、right: 0、top: 0、bottom: 0 を適用する。すると、画面いっぱいのサイズになる。
- **B**  
left: 0、top: 0 とした上で、width: 100%、height: 100% 画面いっぱいのサイズになる。

Aはビヨーンって引き延ばす感じ



Bは大きいやつを左上に置く感じ



# 背景に半透明のスクリーンを配置する

\_alert-sample-1.sassを参照してください

```
1 // 半透明のスクリーン
2 .alert-background
3   background-color: $black-dark
4   position: fixed
5   left: 0
6   right: 0
7   top: 0
8   bottom: 0
9
10  // これでもいける
11  left: 0
12  top: 0
13  width: 100%
14  height: 100%
```

# ちなみに bourbon には position というmixin がある

.element

+position(relative, 0 null null 10em)

absolute とか fixed とか relative とか、  
position の値を入れる

top、right、bottom、left の順で値を入れる



# 画面中央にブロックを配置する

\_alert-sample-1.sassを参照してください

- ・ 画面中央に配置したいブロック .alert に、 position: fixed を適用する。すると、.alert-background はスクロールしても同じ位置に留まるようになる。
- ・ .alert に、 left: 50%、 top: 50% を適用させる。すると、画面の中央より自分自身のサイズの分だけ中央からずれた位置に配置される。
- ・ 自分自身のサイズの半分のサイズだけネガティブマージンをつけると真ん中にくるので、 margin-left: -240px、 margin-top: -160px を適用させる。
- ・ 自分自身にborderが付いているので、デフォルトの状態だと、自分自身のサイズが width (height) + 左右（上下）のborder-width になってしまうので、 box-sizing: border-box を適用する。  
すると、border-width に関係なく、width・height の値が自分自身のサイズになる。
- ・ さっきの半透明のスクリーンが .alert の上に乗っかってるので、.alert に z-index: 1 を適用させる。

# 画面中央にブロックを配置する

\_alert-sample-1.sassを参照してください

```
16 // アラート
17 .alert
18   width: 480px
19   height: 320px
20   background-color: white
21   border: solid 4px $red-700
22   position: fixed
23   left: 50%
24   top: 50%
25   margin-left: -240px
26   margin-top: -160px
27   box-sizing: border-box
28   z-index: 1
```

# 右上に正円のアイコンを配置する

[\\_alert-sample-1.sass](#)を参照してください

- font-awesome のアイコンを囲っている `.alert__close` に `position: absolute` を適用する。すると、`.alert__close` を 親要素の `.alert` を基準に絶対値で配置をすることができるようになる。`position: absolute` を適用すると、自分自身のサイズは、親要素に影響を与えなくなる（fixedも）。
- `.alert__close` に `right: -20px, top: -20px` を適用すると、`.alert` 内の右上はみ出した位置に配置される。
- `.alert__close` に `text-align: center` を適用する。すると、`.alert__close` の子要素である。`.alert__close-icon` が x軸の中央に配置される。
- `.alert__close` に `border-radius: 50%` を適用する。すると、`.alert__close` が正円になる。正円を作りたい場合は、正方形の要素に `border-radius: 50%` を適用すると正円になる。
- font-awesome のアイコン `.alert__close-icon` に `text-align: 40px`（親要素のheightと同じ値）を適用する。すると、`.alert__close-icon` が y軸の中央に配置される。

# 右上に正円のアイコンを配置する

\_alert-sample-1.sassを参照してください

```
30 // アラートのクローズ
31 .alert__close
32   background-color: $grey-800
33   width: 40px
34   height: 40px
35   position: absolute
36   right: -20px
37   top: -20px
38   text-align: center
39   border-radius: 50%
40 .alert__close-icon
41   font-size: 28px
42   color: white
43   line-height: 40px
```

# 真ん中寄せの文字の字間を調整する

\_alert-sample-1.sassを参照してください

- .alert\_\_title に letter-spacing: 8px を適用する。すると、文字の後に 8px の隙間ができる。ただ、これだけだと左に寄ってしまう。



左に寄ってしまう。

- .alert\_\_title に text-indent: 8px を適用し、左に寄ってしまった分、右に移動させる。

# 真ん中寄せの文字の字間を調整する

\_alert-sample-1.sassを参照してください

```
55 // アラートのタイトル  
56 .alert_title  
57   font-size: 24px  
58   font-weight: bold  
59   color: white  
60   letter-spacing: 8px  
61   text-indent: 8px  
62
```

8

articles

# 課題

CSS Tutorial



32px



2015/12/22



2015/12/04

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに



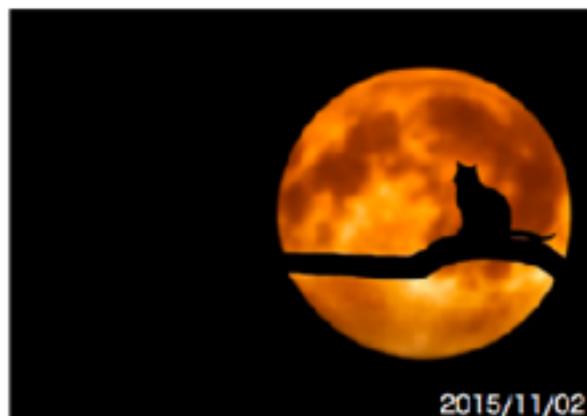
2015/11/27

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに



2015/11/13

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに



2015/11/02

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに



いじる sass ファイルの場所

/app/assets/stylesheets/components/articles/articles

# 課題

- ・ グリッドシステムってどうやってつくるの？
- ・ 画像の上に文字を重ねるのってどうやるの？

# グリッドシステム

\_articles-sample-1.sassを参照してください

- まずは、.articles\_\_item を横並びにしたいので横並びにしたい .articles\_\_item に float: left を適用し、横幅を三分割するので、width: 33.3% を適用する。これで横並びになる。
- 横並びにした .articles\_\_item の左右に 32px の隙間を作るので、.articles\_\_item 左右に padding で隙間を作る。32pxの半分のpaddingを左右にそれぞれ適用するので、padding-left: 16px、padding-right: 16px を適用する。
- このままだと、.articles\_\_item の横幅が 33.3% + 左右のpadding になってしまうので、box-sizing: border-box を適用する。すると、paddingの値を含めて .articles\_\_item の横幅が 33.3% になる。

# グリッドシステム

\_articles-sample-2.sassを参照してください

- ・ 次は flexbox を使った実装に挑戦。  
まずは、.articles\_item を横並びにしたいので横並びにしたい .articles\_item の親である .articles\_items に display: flex を適用する。
- ・ 次に、横並びにしたい .articles\_item に flex: 33.3% を適用し、横幅を三分割する。まだこの時点では折り返されず、一行で表示されてしまう。
- ・ display: flex を適用した .articles\_items に flex-wrap: wrap を適用。すると、.articles\_item に適用した 33.3% をはみ出した時点で折り返される。
- ・ 横並びにした .articles\_item の左右に 32px の隙間を作るので、.articles\_item 左右に padding で隙間を作る。32pxの半分のpaddingを左右にそれぞれ適用するので、padding-left: 16px、padding-right: 16px を適用する。
- ・ このままだと、.articles\_item の横幅が 33.3% + 左右のpadding になってしまうので、box-sizing: border-box を適用する。すると、paddingの値を含めて .articles\_item の横幅が 33.3% になる。

# グリッドシステム

\_articles-sample-1.sass か、\_articles-sample-2.sass を参照してください

- .articles\_\_item 全体を囲っている .articles\_\_items の左右に隙間ができてしまっているので、それを消すために .articles\_\_items の左右にネガティブマージンを付けて隙間をなくす。



.articles\_\_items に margin-left: -16px、margin-right: -16px を適用する。

# グリッドシステム

\_articles-sample-1.sassを参照してください

- 画像がグリッドからはみ出しているので、画像の class である .articles\_\_item-image に width: 100% を適用する。

```
1  .articles__items
2    margin-left: -16px
3    margin-right: -16px
4  .articles__item
5    margin-bottom: 32px
6    float: left
7    width: 33.3%
8    padding-left: 16px
9    padding-right: 16px
10   box-sizing: border-box
11  .articles__item-image-container
12  margin-bottom: 4px
13  .articles__item-image
14  width: 100%
```

# 画像に文字を重ねる

[\\_articles-sample-1.sass](#) か、[\\_articles-sample-2.sass](#) を参照してください

- `.articles__item-created-at` を重ねたいので、`position: absolute` を適用させる。
- `.articles__item-created-at` を絶対値で配置したいので、その基準になる親要素、`.articles__item-image-container` に `position: relative` を適用させる。  
絶対値で配置をする場合、`relative`、`absolute`、`fixed` のいずれかが適用された直近の親要素が絶対値の基準になる。
- `.articles__item-image-container` を絶対値の基準にしたので、それを基準に `.articles__item-created-at` を右下に配置するため、`right: 0`、`bottom: 0` を適用する。

# 画像に文字を重ねる

\_articles-sample-1.sass か、\_articles-sample-2.sass を参照してください



横幅いっぱいにしたい

それは黄金の星下がり気ままにただようば  
くらオールは二本ともあぶなげに

- .articles\_\_item-created-at を横幅いっぱいにしたいので、left: 0 も適用させる。  
width: 100% と、box-sizing: border-box を適用させるのでもOK。

# 画像に文字を重ねる

\_articles-sample-1.sassを参照してください

```
11   .articles__item-image-container
12     margin-bottom: 4px
13     position: relative
14   .articles__item-image
15     width: 100%
16   .articles__item-created-at
17     background-color: $black-dark
18     text-align: right
19     line-height: 24px
20     padding-right: 8px
21     color: $white
22     font-size: 14px
23     position: absolute
24     right: 0
25     bottom: 0
26     left: 0
27     //
28     これでもOK
29     width: 100%
30     box-sizing: border-box
31   .articles__item-title
32     font-size: 18px
33     font-weight: bold
34     line-height: 1.4
```

# もし、タイトルが長いものがあったら？

\_articles-sample-1.sass か、\_articles-sample-2.sass を参照してください



2015/12/31

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげにそれは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげにそれは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげにそれは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげにそれは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに



2015/12/22

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに



2015/12/04

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに



2015/11/27

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに



2015/11/13

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに

タイトルが長くてグリッドが崩れる



# もし、タイトルが長いものがあったら？

`_articles-sample-1.sass` か、`_articles-sample-2.sass` を参照してください

- `.articles__item-title` の `height` を文字二行分とする。二行分は、`.articles__item-title` の `font-size` が `18px`、行間が `1.4`、行数が `2行`、なので、`height: (18px * 1.4) * 2` が二行分の高さになる。つまり `height: 50.4px` とすれば、文字二行分の高さになるが、`height: (18px * 1.4) * 2` としても、sassが演算をやってくれるので、自分で計算をやらなくてもOK。

# もし、タイトルが長いものがあったら？

\_articles-sample-1.sass か、\_articles-sample-2.sass を参照してください  
はみ出したところが隠れた



2015/12/31

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげにそれは黄



2015/12/22

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに



2015/12/04

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに



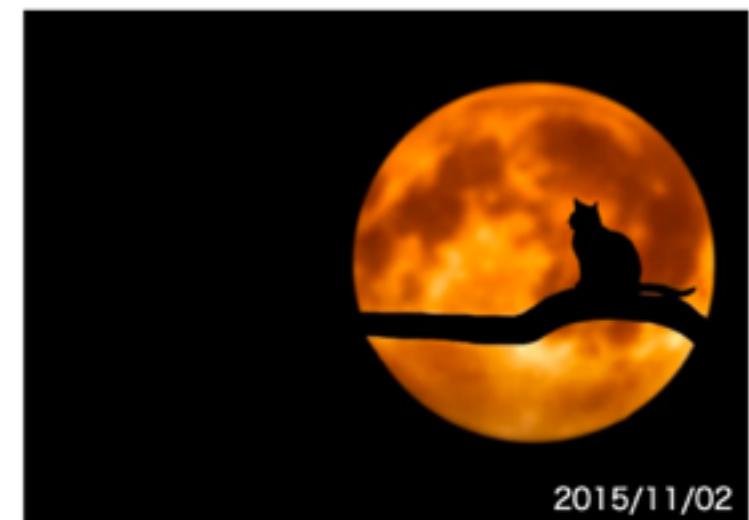
2015/11/27

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに



2015/11/13

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに



2015/11/02

それは黄金の屋下がり気ままにただようぼ  
くらオールは二本ともあぶなげに

# もし、タイトルが長いものがあったら？

\_articles-sample-1.sass か、\_articles-sample-2.sass を参照してください

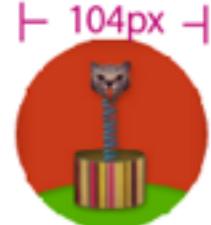
```
27 .articles__item-title  
28   font-size: 18px  
29   font-weight: bold  
30   line-height: 1.4  
31   overflow: hidden  
32   height: (18px * 1.4) * 2  
33
```

9

comment

# 課題

CSS Tutilial



152px

104px

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに小さな腕で潜がれ小さな手がぼくらのただよいを導こうとかっこうだけ申し訳につけてああ残酷な三人！こんな時間にこんな夢見る天気のもとで

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに小さな腕で潜がれ小さな手がぼくらのただよいを導こうとかっこうだけ申し訳につけてああ残酷な三人！こんな時間にこんな夢見る天気のもとで

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに小さな腕で潜がれ小さな手がぼくらのただよいを導こうとかっこうだけ申し訳につけてああ残酷な三人！こんな時間にこんな夢見る天気のもとで

それは黄金の屋下がり気ままにただようぼくらオールは二本ともあぶなげに小さな腕で潜がれ小さな手がぼくらのただよいを導こうとかっこうだけ申し訳につけてああ残酷な三人！こんな時間にこんな夢見る天気のもとで

100%



machida

22px

20px

それは黄  
こうとか  
それは黄  
こうとか

# 課題

CSS Turtial

← 104px →



machida

← 152px →

それは黄金の屋下がり氣ままにただようぼくらオールは二本ともあぶなげに  
小さな腕で漕がれ小さな手がぼくらのただよいを導こうとかっこうだけ申し  
訳につけてああ残酷な三人！こんな時間にこんな夢見る天気のもとで

それは黄金の屋下がり氣ままにただようぼくらオールは二本ともあぶなげに  
小さな腕で漕がれ小さな手がぼくらのただよいを導こうとかっこうだけ申し  
訳につけてああ残酷な三人！こんな時間にこんな夢見る天気のもとで

それは黄金の屋下がり氣ままにただようぼくらオールは二本ともあぶなげに  
小さな腕で漕がれ小さな手がぼくらのただよいを導こうとかっこうだけ申し  
訳につけてああ残酷な三人！こんな時間にこんな夢見る天気のもとで

それは黄金の屋下がり氣ままにただようぼくらオールは二本ともあぶなげに  
小さな腕で漕がれ小さな手がぼくらのただよいを導こうとかっこうだけ申し  
訳につけてああ残酷な三人！こんな時間にこんな夢見る天気のもとで

画面を狭くしても、コメント者のサイズは変わらないようにする

# 課題

- ・画面の幅を狭くしても片方のカラムのサイズは変わらないレイアウトってどうやるの？
- ・CSSだけで吹き出しへどうやって作るの？

いじる sass ファイルの場所

/app/assets/stylesheets/components/comment/\_comment.sass

# 画面サイズを変えても一方のカラムは サイズが維持されるレイアウト

[\\_comment-sample-1.sass.sample](#) を参照してください

- まずは、サイズが変わらないカラムの方、`.comment_author` に `position: absolute` を適用する。
- すると、サイズの変わる方のカラム `.comment_body` と、`.comment_author` が重なるので、それらを囲ってる親に `padding` を付けて2つが重ならないように、`.comment_author` の入る隙間を作つてあげる。親 `.comment` に、`padding-left: 152px` を適用。
- `.comment_author` に絶対値での位置を与えていないので、`.comment_author` も一緒に親の `padding` 分離してしまう。`.comment_author` に絶対値での位置の値を付けたいが、その前にその絶対値の基準となる親である `.comment` に、`position: relative` を適用する。
- `.comment_author` の位置の基準ができたところで、`.comment_author` に位置を適用する。`.comment_author` は左上にあるので、`left: 0`、`top: 0` を適用する。
- 画面を狭めてみて、`.comment_author` のサイズは変わっていないことを確認。

# 画面サイズを変えても一方のカラムは サイズが維持されるレイアウト

\_comment-sample-1.sass.sample を参照してください

```
1   .comment
2     padding-left: 152px
3     position: relative
4     .comment__author
5       position: absolute
6       left: 0
7       top: 0
8     .comment__author-image
9       border-radius: 50%
10    margin-bottom: 8px
11    width: 104px
12    height: 104px
13    .comment__author-name
14      width: 104px
15      text-align: center
16    .comment__body
17      background-color: $teal-50
18      padding: 18px 24px
19      p
20        font-size: 16px
21        line-height: 1.6
22        margin-bottom: .9em
23
```

# 画面サイズを変えても一方のカラムは サイズが維持されるレイアウト2

`_comment-sample-2.sass.sample` を参照してください

- ・ 次は flexbox を使ったパターン。まずは横並びにしたい要素の親である `.comment` に `display: flex` を適用する。
- ・ 続いて、サイズが変わらないカラムの方、`.comment__author` に `flex: 0 0 152px` を適用。もしくは、`width: 152px` でもOK。

# 画面サイズを変えても一方のカラムは サイズが維持されるレイアウト2

[\\_comment-sample-2.sass.sample](#) を参照してください

- ・ 次は flexbox を使ったパターン。まずは横並びにしたい要素の親である .comment に display: flex を適用する。
- ・ 続いて、サイズが変わらないカラムの方、.comment\_\_author に flex: 0 0 152px を適用。もしくは、width: 152px でもOK。

これでおしまい。flexboxヤバい。

# CSSで吹き出しを作る

\_comment-sample-1.sass.sample を参照してください

- ・ 吹き出しがある要素 .comment\_\_body に擬似要素 :before を作り、その擬似要素に content: '' を適用する。

```
16  .comment__body
17    background-color: $teal-50
18    padding: 18px 24px
19    &:before
20      content: ''
21    p
22      font-size: 16px
23      line-height: 1.6
24      margin-bottom: .9em
```

すると、何もない要素が出現する。

# CSSで吹き出しを作る

\_comment-sample-1.sass.sample を参照してください

The screenshot shows the Google Chrome Developer Tools Elements tab. A red arrow points from the text "before が出現！" at the bottom left to the CSS code in the Styles panel.

**Elements** tab is selected. The DOM tree shows a comment element with a ::before pseudo-element applied.

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div id="xray-bar" style="display:none">...</div>
    <header class="header">...</header>
    <main class="main is-padding-top-24px">
      <div class="container">
        <div class="comment">
          <div class="comment__author">...</div>
          <div class="comment__body">
            ::before
              <p>
                "それは黄金の屋下が  
こうとかっこうだけ  
それは黄金の屋下が  
こうとかっこうだけ  
それは黄金の屋下が  
こうとかっこうだけ  
それは黄金の屋下が  
こうとかっこうだけ"
              </p>
            <p>...</p>
          </div>
        </div>
      </div>
    </main>
  </html>
```

**Styles** tab is selected. The CSS rule for the ::before pseudo-element is shown:

```
media="all"
.comment__body::before {
  content: "";
```

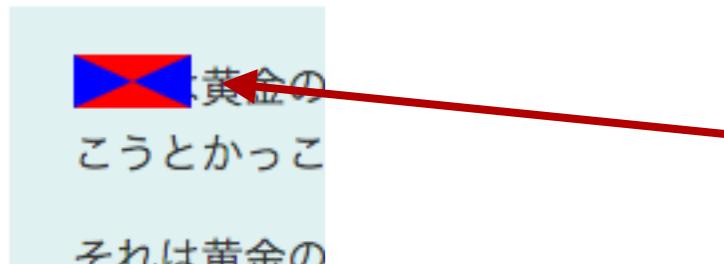
The right panel shows the computed styles for the ::before element, including margin, border, and padding.

before が出現！

# CSSで吹き出しを作る

\_comment-sample-1.sass.sample を参照してください

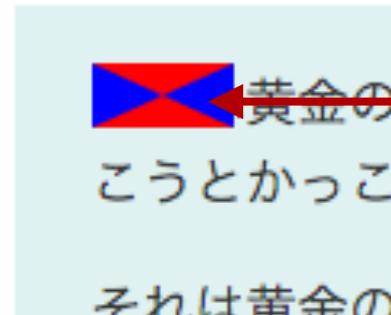
- ・擬似要素 .comment\_\_body:before に position: absolute を適用。絶対値で場所が指定できるようにする。
- ・吹き出しは border を使って作るので、まずは擬似要素 .comment\_\_body:before に border-style: solid を適用する。
- ・次に、border-width を適用する。吹き出しのトンガリが、幅22px、高さ 20px なんだけど、左方向のトンガリを作る場合は、上下のborder-width (border-top-widthとborder-bottom-width) をトンガリの高さ 20px の半分の 10px を適用させて、左右のborder-width (border-left-widthとborder-right-width) はトンガリの幅と同じ 22px を適用させる。  
擬似要素 .comment\_\_body:before に border-width: 10px 22px を適用。
- ・ひとつ前の工程で何をやってるかがわかるように、一時的に border に色を付けてみる。  
擬似要素 .comment\_\_body:before に border-color: red blue を適用してみる。  
ちなみに上下の border に red、左右の border に blue を適用。



こんなものが出現する

# CSSで吹き出しを作る

\_comment-sample-1.sass.sample を参照してください



**width、height 0 の要素に border を付けると三角形の集合になる。  
border-rightだけを生かして、他の border を透明にすることで  
吹き出しのトンガリを作ってしまおうという魂胆**

- ・ 先ほどの確認のための border-color: red blue は削除する。
- ・ 擬似要素 .comment\_\_body:before の border を一回すべて透明にするので、  
border-color: transparent を適用する。
- ・ すべて透明になったところで、生かす border-right に、親である .comment\_\_body の背景色と同じ  
色を付ける。  
border-right-color: \$teal-50 を適用する。
- ・ トンガリができたので、これをいい位置に配置したい。トンガリには position: absolute が適用され  
るので、その絶対値の基準となる要素 .comment\_\_body に position: relative を適用する。
- ・ 基準ができたらトンガリに位置の値を与えていい位置に置く。  
擬似要素 .comment\_\_body:before に left: -44px、 top: 32px を適用する。

# CSSで吹き出しを作る

\_comment-sample-1.sass.sample を参照してください

```
16 .comment__body
17   background-color: $teal-50
18   padding: 18px 24px
19   position: relative
20   &:before
21     content: ''
22     position: absolute
23     border-style: solid
24     border-width: 10px 22px
25     border-color: transparent
26     border-right-color: $teal-50
27     left: -44px
28     top: 32px
29   p
30     font-size: 16px
31     line-height: 1.6
32     margin-bottom: .9em|
```

# headings (おまけ)

# headings

- ・ /practices/heading にアクセス
- ・ app/assets/stylesheets/components/headings この中に、今日やった内容を使って、見た目はださいんですが、cssだけを使って色々な heading を作ってみました。これを参考に、自分でも作って遊んでみてください。

