西脇.rb 神戸.rb

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 で立ち上がります。
- 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。

3

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あんまり 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

デザインパーツ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

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をまとめてる。
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

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

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をまとめてる。
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 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。

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]

Amount

Input [Filled]

Address 2255 Spruce Street

Input [Active]

Zip 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.

Search

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

article body

recent tweets

comments

Templates（テンプレート）

- コンポーネントをページに並べたもの

Pages（ページ）

- Template に実データを投入したもの

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れる。

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

atoms（原子）

↓

molecules（分子）

↓

organisms（有機体）

↓

~~templates（テンプレート）~~

↓

pages（ページ）

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

atoms（原子）

↓

molecules（分子）

↓

organisms（有機体）

↓

~~templates（テンプレート）~~

↓

pages（ページ）

パーツ（atoms）
ブロック（molecules）
コンポーネント（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

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。
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

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。
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。

Eric Meyer's "Reset CSS" が一番使われてる

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

Watch 314 | Unstar 6,598 | Fork 1,200

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

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

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

html5-boilerplate

Pure

Normalize.css

色んなフレームワークで使われてる。

bootstrap

html5-boilerplate

Pure

今は、normalize.cssが一番人気みたい

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

```
code,
kbd,
pre,
samp {
    font-family: monospace, monospace;
}
```

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！

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。

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。

要素 “title”

BEMとは？

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

修飾 “reset”

（今回の場合、背景色が赤いことが修飾）

BEMとは？

[Block]__[Element]--[Modifire]

.home-nav__title—reset

BEMとは？

[Block]__[Element]--[Modifire]

.home-nav__items

（BEだけ、BMだけってこともある）

BEMとは？

[Block]__[Element]--[Modifire]

.home-nav__item

BEMとは？

[Block]__[Element]--[Modifire]

.home-nav__item-link

BEMとは？

[Block]__[Element]--[Modifire]

BEMとは？

僕の考えでは、まぁ、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、まぁ、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。

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。

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

```
%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”
```

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

まぁ、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

まぁ、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

まぁ、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

まぁ、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

まぁ、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
```

これが面倒なのでModifierは別 class にしています。

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多くあります。
（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くするなると思います。

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

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。

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

こっちを触って
作業をする

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

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

```
.global-nav__items
  display: table
  table-layout: fixed
  width: 100%
.global-nav__item
  display: table-cell
```

B

```
.global-nav__items
  display: table
  width: 100%
.global-nav__item
  display: table-cell
  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…

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

- 親に収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

- 親に収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ろに <> を付ける。

```
%nav.global-nav
  .container
    %ul.global-nav__items
      %li.global-nav__item<>
        = link_to "リンク", "#", class: "global-nav__item-link"
      %li.global-nav__item<>
        = link_to "リンク", "#", class: "global-nav__item-link"
      %li.global-nav__item<>
        = link_to "リンク", "#", class: "global-nav__item-link"
      %li.global-nav__item<>
        = link_to "リンク", "#", class: "global-nav__item-link"
      %li.global-nav__item<>
        = 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 を下まで持っていきたい、.global-nav に clearfix 適用する

```
.global-nav
  background-color: $teal-300
  +clearfix
.global-nav__item
  width: 20%
  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

- 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 を下まで持っていきたい、.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

- 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

```
// 半透明のスクリーン
.alert-background
  background-color: $black-dark
  position: fixed
  left: 0
  right: 0
  top: 0
  bottom: 0
  //
    これでもいける
    left: 0
    top: 0
    width: 100%
    height: 100%
```

ちなみに bourbon には position というmxin 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

```
// アラート
.alert
  width: 480px
  height: 320px
  background-color: white
  border: solid 4px $red-700
  position: fixed
  left: 50%
  top: 50%
  margin-left: -240px
  margin-top: -160px
  box-sizing: border-box
  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、文字の後ろに 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
```

8

articles

課題

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、paddig-left: 16px、paddig-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、
  paddig-left: 16px、paddig-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。

```
.articles__items
  margin-left: -16px
  margin-right: -16px
.articles__item
  margin-bottom: 32px
  float: left
  width: 33.3%
  padding-left: 16px
  padding-right: 16px
  box-sizing: border-box
.articles__item-image-container
  margin-bottom: 4px
.articles__item-image
  width: 100%
```

画像に文字を重ねる

_articles-sample-1.sassを参照してください

- .articles__item-crerated-at を重ねたいので、position: absolute を適用させる。
- .articles__item-crerated-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-crerated-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を参照してください

横幅いっぱいにしたい

- .articles__item-crerated-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-crerated-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
```

9

comment

課題

CSS Tutrial
104px
machida
152px
それは黄金の昼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
それは黄金の昼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
それは黄金の昼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
それは黄金の昼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
100%

22px
20px

課題

CSS Tutrial

104px

machida

152px

それは黄金の昼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

それは黄金の昼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

それは黄金の昼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

それは黄金の昼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だけで吹き出しってどうやって作るの？

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 の入る隙間を作ってあげる。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 分ずれてしまう。.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

```
.comment
  padding-left: 152px
  position: relative
.comment__author
  position: absolute
  left: 0
  top: 0
.comment__author-image
  border-radius: 50%
  margin-bottom: 8px
  width: 104px
  height: 104px
.comment__author-name
  width: 104px
  text-align: center
.comment__body
  background-color: $teal-50
  padding: 18px 24px
  p
    font-size: 16px
    line-height: 1.6
    margin-bottom: .9em
```

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。

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。

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

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 が適用されてるので、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、見た目はダサいんですが、cssだけを使って色んな 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。