

# エンジニアとデザイナー との距離

FRONTEND CONFERENCE 2017

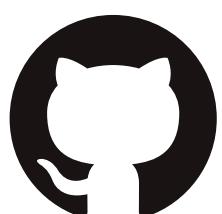
# INTRODUCTION

## 自己紹介

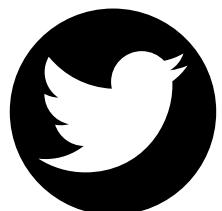


**安田 学** (Yasuda Manabu)

株式会社 TAM  
マークアップエンジニア



[github.com/manabuyasuda](https://github.com/manabuyasuda)



[twitter.com/Gaku0318](https://twitter.com/Gaku0318)



CSSをテーマにした勉強会を  
4月22日(土)に開催します。

今回のお話は、Web業界で1年働いて感じた  
デザイナーとエンジニアとの距離感について。

CSSを設計するって大変ですよね。

実践して感じたのは、コードを工夫するだけで  
CSSを設計するのは難しいということ。

プログラミングの考え方を取り入れた、  
OOCSS的なCSS設計手法にも  
デメリットがあると感じています。

# oocssのデメリット

- ・コードが複雑になるので扱いが難しい
- ・想定していないデザインの追加や変更が増えると整合性がとれなくなることが多い

問題点は大きく2つ。

# エンジニア目線の問題点

抽象化によりコードが複雑になり、  
共通化によりコード同士が密結合になりやすい。

# デザイナー目線の問題点

デザイナーがコードを理解しにくいで、  
エンジニアとの意思の疎通がとりにくい。

デザイナーとエンジニアの「認識のズレ」が  
問題を引き起こしているのかもしれません。

CSSのベストプラクティスのひとつに  
「意味のある名前にする」というものがあります。

デザインの見栄えを  
コードの名前にするのではなく、  
デザインの意図やコンテキストを  
コードの名前に使うということ。

**デザインの意図**とは  
その見栄えにしている理由のこと。

Bad

.button-arrow

詳細情報を見る >

Good

.button-detail  
.button-more

Bad

詳細情報を見る >

ショップで購入する >

Good

.button-detail  
.button-purchase

.button-secondary  
.button-primary

コンテキストとは  
前後の文脈や状況のこと。

# Webサイトでのおもなコンテキスト

- トップページ
- 詳細ページ
- カテゴリー
- 表示する場所(ヘッダーやフッターなど)

見栄えは30分後に変わっても、  
デザインの意図やコンテキストは  
基本的に変わらないはず。

「どんな場所で使いますか？」

「タブレットではどう表示されますか？」

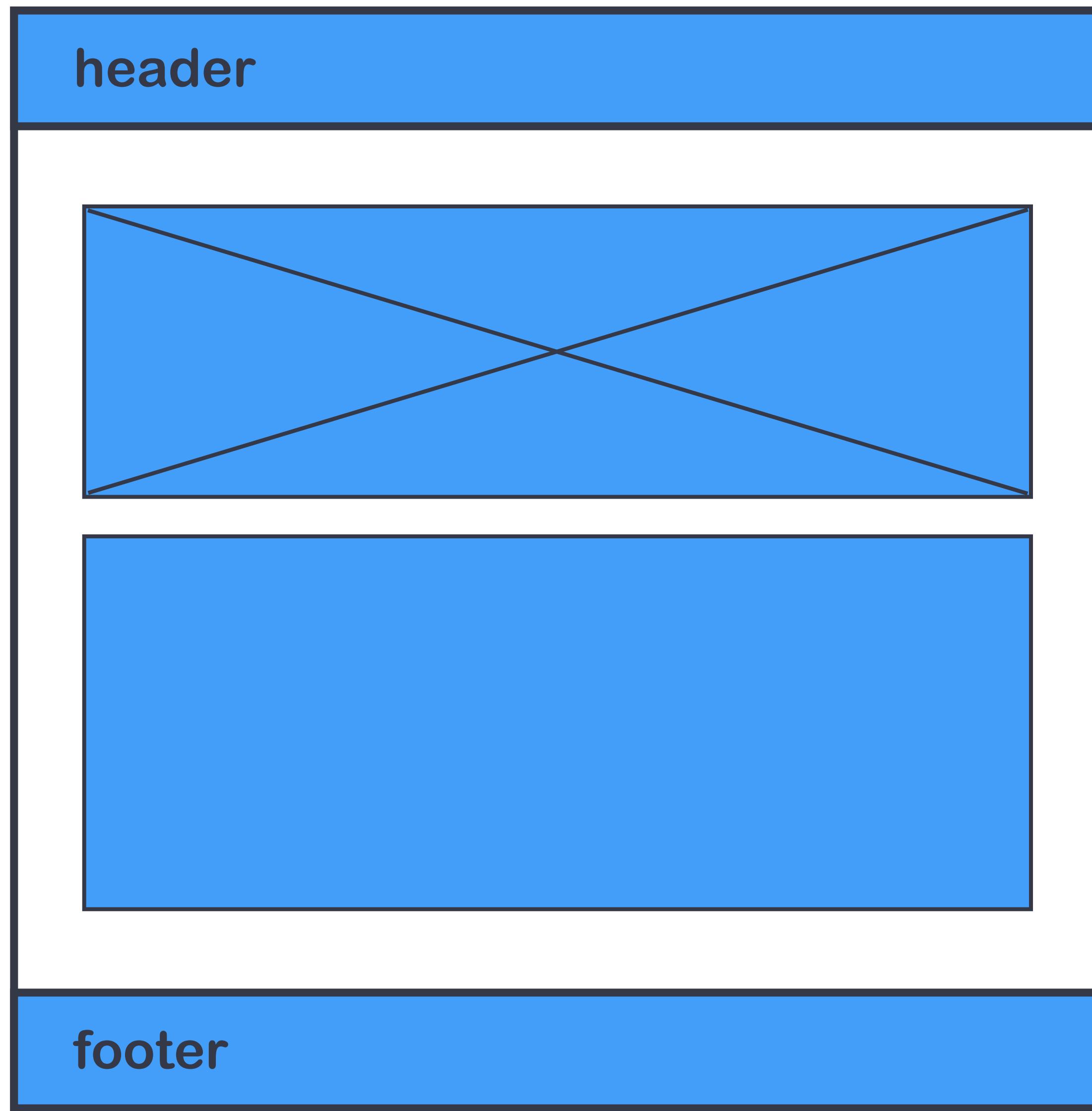
「補足的な情報は常にこのサイズですか？」

デザイナーとエンジニアの、  
デザインに対する認識のズレを  
できるだけ少なくすることが大切です。

設計手法は「[Enduring CSS\(ECSS\)](#)」  
から始めるのが最適。

ECSSはコンテキストを重視している  
CSSの設計手法だから。

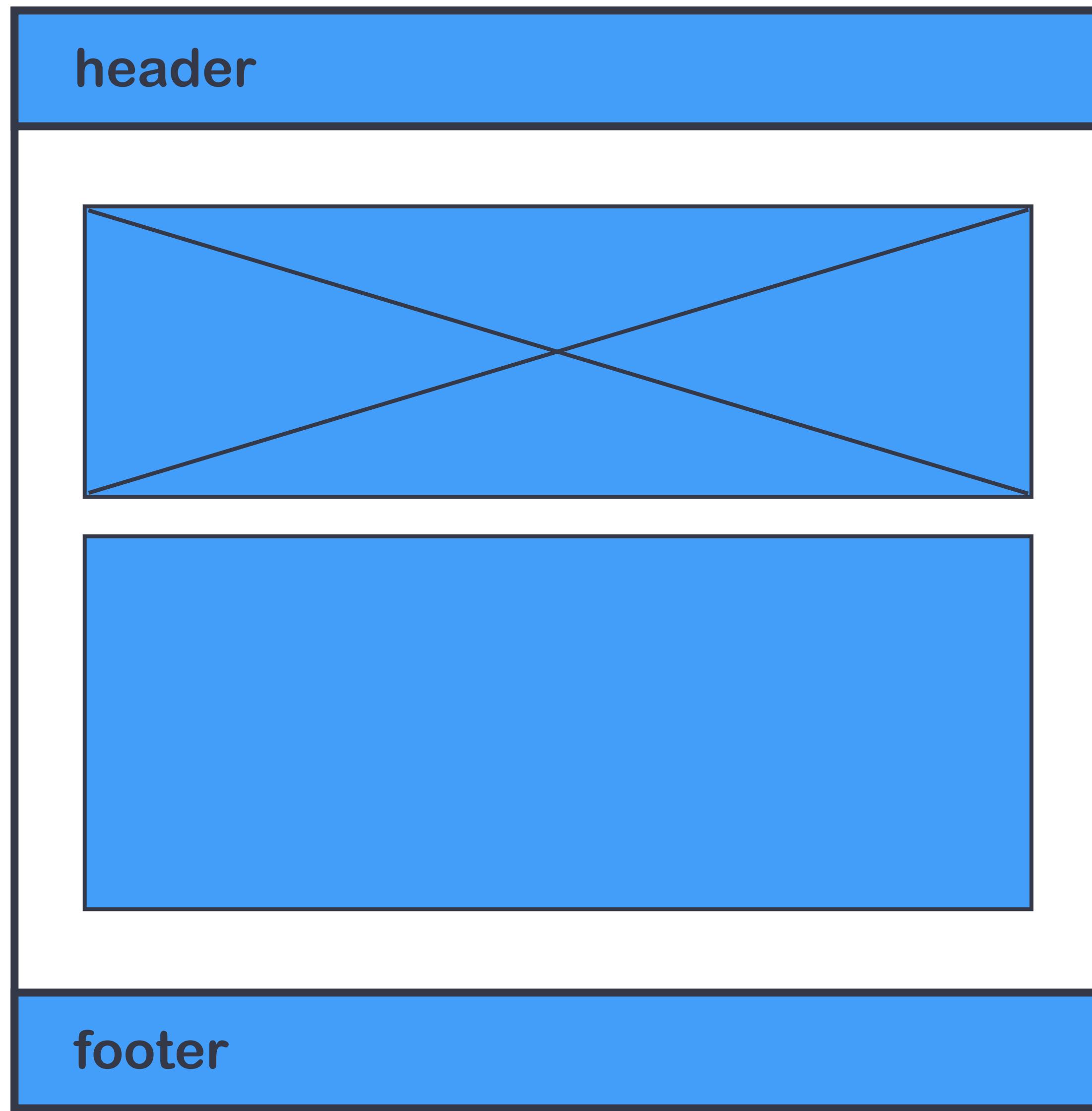
※詳しくは [ecss.io](https://ecss.io) を確認してください



.header

.contents

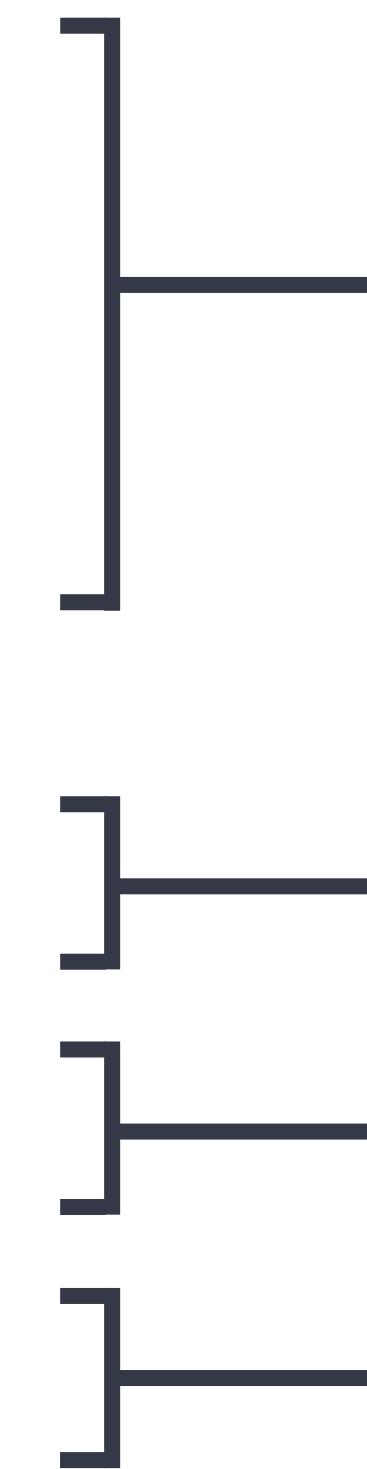
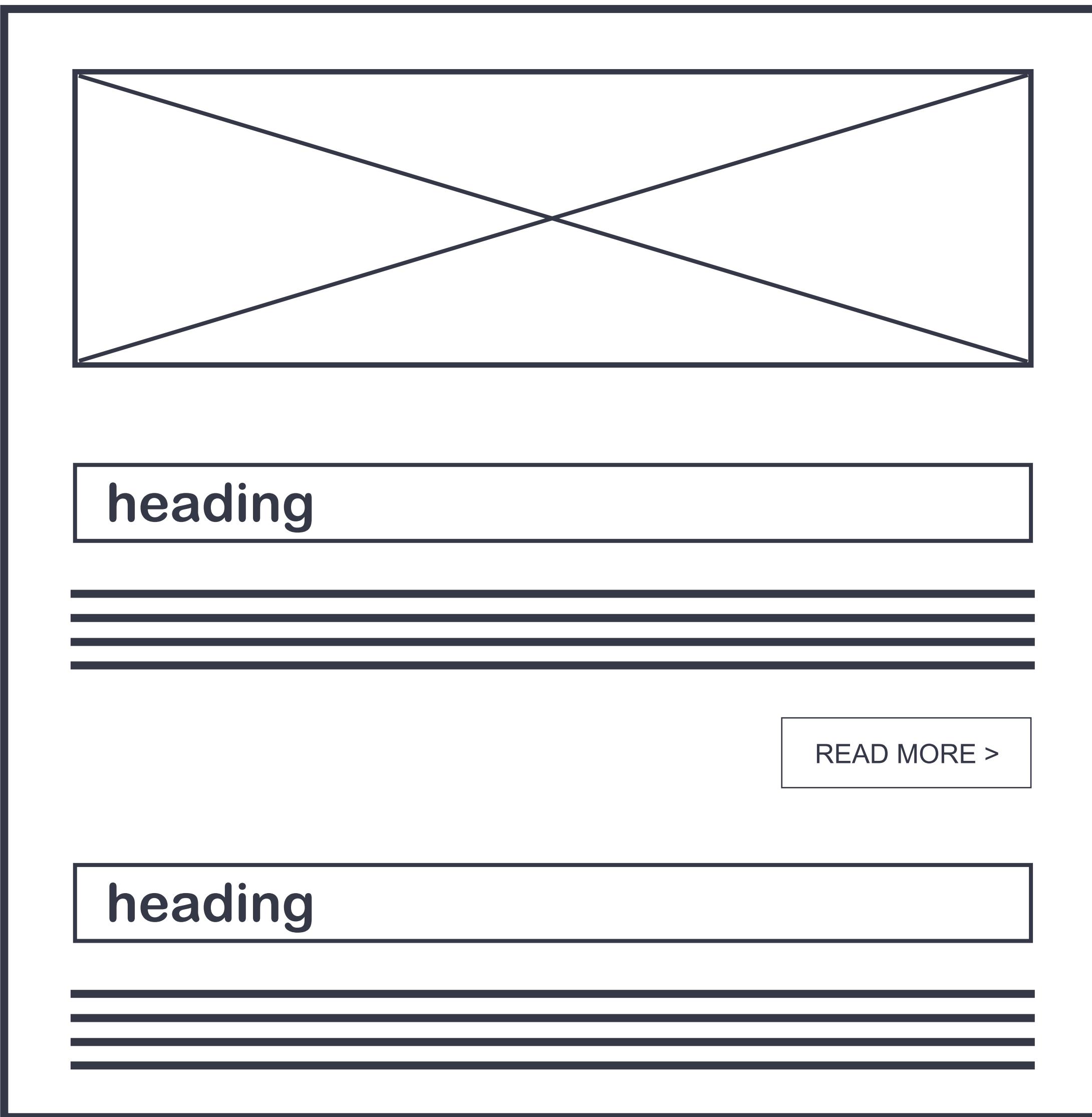
.footer

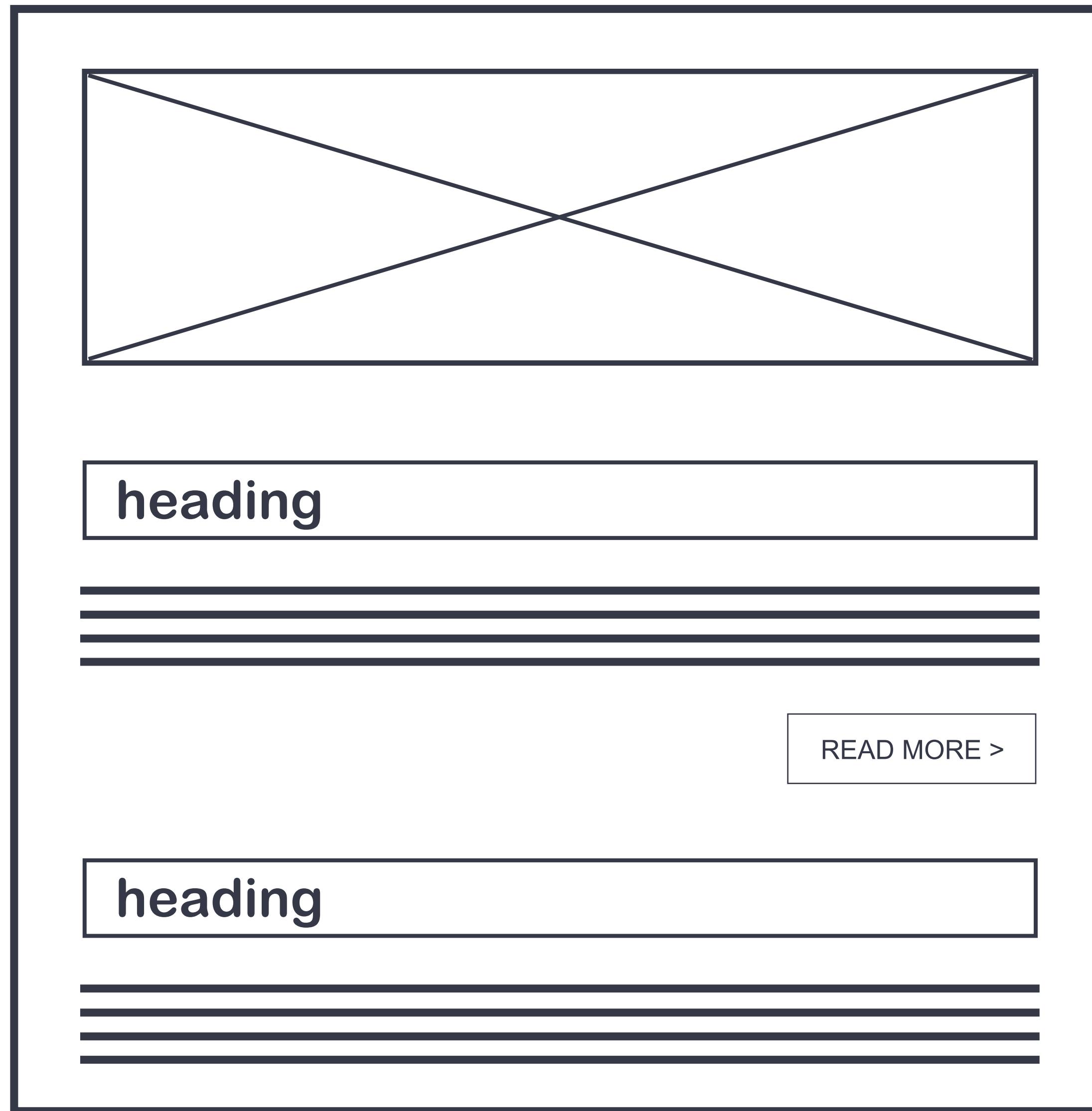


.st-Header

.st-Contents

.st-Footer





.hero-Standard



.tp-Heading



.tp-Text



.tp-MoreButton



.sw-Button

基本的には DRY(重複を避けること)を  
目指しませんが、  
変数と最小限の mixin などで共通化はします。

タイプグラフィー、カラーパレット、  
サイズなどを変数化して、  
デザイナーとエンジニアの認識をあわせます。

# ECSSのメリット

- ・コードがシンプルで扱いやすくなる
- ・デザインの変更に対応しやすい
- ・コードがどこで使われているか把握できる

# ECSSのデメリット

- ・ コンテキスト単位で  
デザインの共通化が必要になる
- ・ コードをうまく共通化しないと、  
デザイン変更で変更箇所が多くなる

いずれの場合も、  
デザインの意図とコンテキストがコードの  
名前と一致するようになります。

コミュニケーションコストが  
増えそうだと思いましたか？

デザインの意図とコンテキストを共有すると、  
長期的にはコミュニケーションコストの  
削減につながります。

ページの量産や運用フェーズに入った状況を  
考えてみます。

# 見栄えと一致させた場合

デザイナーはすべてのページに対して、  
細かく指示をしたり、チェックをする  
必要があります。

## デザインの意図やコンテキストと一致させた場合

使う場所は自然と決まってくるので、  
細かく指示をする必要がありません。

デザイナーの手離れが早くなり、  
エンジニアも迷うことが少なくなります。

特にサイトの規模が大きくなるごとに、  
デザインの意図とコンテキストの共有は  
とても大切です。

ドキュメントとして残しておけば、  
プロジェクトを引き継いだ  
デザイナーやエンジニアにも優しい。

さいごに

まずは、  
「コミュニケーションのとりやすい環境」  
をつくっていきたいと思っています。

これからしていこうと考えていることを  
少しだけお伝えします。

デザインキャンプについて、  
困っていることや知りたいことを伝える

- ・共通するタイポグラフィーや色などを知りたい
- ・書き出しがすぐ出来る状態で渡して欲しい
- ・デザインの意図を知りたい

困っていることは早めに伝えたほうが  
お互いにとって楽になりますよね。

手戻りにならないように、  
デザインカンプの作成後ではなく、  
作成前に共有しておく必要があります。

Webデザインで想定しておくべきことを  
テンプレート化しておく

- ・ある画面幅での見え方
- ・文章が長いときの見え方
- ・エラーがあったときの見え方
- ・アニメーションの速さや動き方

問題点を探す時間より、  
一緒に考える時間を多くしていきたい。

完璧なデザインキャンプを目指すよりも、  
デザインキャンプをコミュニケーションツール  
として有効活用してみる。

まとめ

- ・ それぞれの職域から得られたノウハウは最終的なアウトプットの質に大きく影響する
- ・ コミュニケーションをするコストだけでなくコミュニケーションをしない損失も大きい
- ・ だからこそ伝えることは大切

ありがとうございました。

slide writing : yasuda manabu

slide design : nakajima eri