命名規則

JavaScriptで触れた命名規則ですが、HTML/CSSにも存在するので把握して、コーディングする際適切のクラス名を定義できるようになってください。

なぜ命名規則（CSS設計）を設けるのか

なぜ設けるのかを考える前いに悪いパターンを上げると。

- その場凌ぎのbox1～box5のような具体性がなくかつ、どこを指しているのかクラス名だけでは読み取れない。（謎の略称も同様）
- 共通化できるCSSのプロパティーがあるにもかかわらず同じ内容を記述してコードの肥大化
- ブロックごとの改行、"{"の前にスペースを入れない、プロパティー内のインデントがなく、人が見難いコード

これが何百、何千にも渡るCSSになったとします。

そうなると背景色1つ変えるだけでもそこに辿り着くまでにとんでもない時間がかかります。

ここに追加実装を行う場合、クラス名はその人が考えたもので統一性がない、CSSのプロパティーも共通化できる可能性があるのに肥大化した中から探すのが困難なので同じ内容を追加する。

これで大きな改修作業（リプレース）となった時に最初から書いた方が良いんじゃないかレベルの作業になってきます。

こう言ったことをできるだけ避けたいので命名規則（CSS設計）を設けます。

命名規則（CSS設計）が目指すゴール

命名規則を設けてCSSを綺麗に書くと言うのは何となく分かったかと思います。

ただ、どう言った状態が理想的なのかが分からないので説明します。

- 予測できる

  「予測できる」とは即ち、「スタイリングが期待通りに振る舞うかどうか」「スタイリングの影響範囲が予測できるか」を意味しています。新しいスタイリングを追加する、または既存のスタイリングを更新しても、自分の意図しない箇所に影響を与えないよう設計されているべきです。

  具体的には改修したときに作業箇所とは別のところがレイアウト崩れを発生させることなどです。

  原因はCSSのプロパティーの意味、影響範囲をあまり理解せずとりあえずデザイン通り作った時によく発生します。

  なので実装時にどれくらい影響があるのかを把握できるようになる必要があります。

- 再利用できる

コードをいちいち書き直したり上書きする手間がない状態が「再利用できる」状態です。そのために、スタイリングはきちんと抽象化されており、また適切に分離されている必要があります

たとえば`float`を使用した時に`clearfix`を使用して回り込みの解除をする必要がありますが、偶に見かけるのが...

- `clearfix`とは別に定義した親要素に対して`clearfix`の内容を記述
- 回り込みの影響を受けた要素に対して`clear`を使用して回り込みの解除

このような実装を進めると、回り込みの解除を行う為にその都度同じ内容のプロパティーを書く必要があります。

それで実装を進めるとコードの肥大化に繋がります。

これをflexboxにしたい時`clearfix`を使用している場合は、`clearfix`で検索して子要素の`float`削除、`clearfix`削除して親要素に`flex`を定義すれば終わりです。

ただ`clearfix`を使用していない場合はデベロッパーツールでどこに`float`、回り込みの解除を行なっているか探しながら作業する必要が発生し、非常に効率が悪いです。

それぞれの要素に対して作成したプロパティーを共通用にさせることを**モジュール化**とも言います。

- 拡張できる

「拡張できる」CSSとは、CSSに携わる人が1人であっても複数からなるチームであっても、問題なく管理できる状態を指します。そのためにはCSS設計の規則はわかりやすく、学習コストが極端に高くない状態である必要があります。

学習コストを考えて自分ルールで作らないようにしてください。

- 保守できる

コードをいちいち書き直したり上書きする手間がない状態が「再利用できる」状態です。そのために、スタイリングはきちんと抽象化されており、また適切に分離されている必要があります。

上で書いた「予測できる」、「再利用できる」、「拡張できる」を踏まえて保守できるコードにする。

代表的な命名規則

- OOCSS

  OOCSSとは、**Object Oriented CSS**の略です。CSSにオブジェクト指向の考え方を取り入れてます。

  Object-Oriented CSS

- BEM

  BEMとは「Block」、「Element」、「Modifier」の頭文字を取った略語です。

  CSSを扱う上でのBEMと言えば、一般的にはHTMLを構成する要素に対するセレクタの命名規則を表す「MindBEMding」を意味します。

  MindBEMding

- SMACSS

SMACSSとは、「Scalable and Modular Architecture for CSS」の略で、「スマックス」と読みます。

SMACSSはCSSを5種類のルールにカテゴライズして記述する手法です。

5種類のルールとは、「ベース」、「レイアウト」、「モジュール」、「ステート」、「テーマ」です。

初心者による初心者向けのSMACSSまとめ

- FLOCSS

  FLOCSS(フロックス)はOOCSS、BEM、SMACSS、SuitCSS、MCSSの考え方を取り入れた設計手法です。

  命名規則はBEMを採用し、「Foundation」、「Layout」、「Object」の3つのレイヤーで構成されています。

  Objectレイヤーは、「Component」、「Project」、「Utility」の3つの子レイヤーを持ちます。

  GitHub

命名規則の採用基準

4つ程命名規則を紹介しましたがどれを採用すべきかと言う疑問が出てきます。

ただこの時はこれと言う基準があまりなく、どう言ったWebサイトを作るかによって変わってきます。

命名規則にもメリット、デメリットがあるのでそれも加味してどの命名規則を採用するか決まってきます。

課題

OOCSS、BEM、SMACSS、FLOCSSについて調べてまとめてください。（どうやって使用するのか具体例も含めて）

またOOCSS、BEM、SMACSSのメリット、デメリットを挙げてください。