Google HTML/CSS Style Guideを全部日本語に訳してみた【CSS編】

CSS コーディング 2015年4月1日

前回、「Google HTML/CSS Style Guide」のHTML部分を翻訳した記事を書きました。
→Google HTML/CSS Style Guideを全部日本語に訳してみた【HTML編】
今回、後半のCSS部分もすべて翻訳したので、記事にしたいと思います。
前回もお伝えしましたが、すでに多くの方が翻訳されており、私もこちらの記事を大いに参考にさせて頂きました。キレイに要点がまとまっているので、とりあえずルールだけ把握したい！という方にはおすすめです。
「Googleが推薦するHTMLとCSSのコーディング方法」

※私の英語力はまだまだ心許ない上に意訳も交じってます、ご了承ください(^^;)ではいきます。

CSSスタイルルール

正しいCSS

可能な限り正しいCSSを使う

CSSバリデーターのバグ対応または特有のシンタックスが必要でない限り、正しいCSSを使ってください。
W3C CSS validatorのようなツールでテストするといいでしょう。
正しいCSSを使えば、ある程度基本的な品質を保てます。一部のCSSが他から影響を受けづらくなったり、動かしやすくなったりします。CSSがふさわしい使用法をされていると保証されます。

IDとクラス名

IDとクラス名には、意味があるまたは一般的な名前を使う

表現的な、もしくはわけのわからない名前にしてはいけません。常に要素の目的を反映した名前か、一般的な名前にしましょう。
明確で要素の目的を反映した、もっとも理解しやすく、もっとも後に変更する必要がないであろう名前にしましょう。
一般的な名前とは、例えば特別な意味が無い要素や、兄弟要素と異なる要因のない要素の代替としてつけるものです。概して、「ヘルパー」として名前が必要とされる場合です。
機能的または一般的な名前を使うことは、不必要なドキュメントやテンプレート変更を減らすことに繋がります。

```
/* NG: 意味が分かりません */
#yee-1901 {}

/* NG: 見た目を表しています */
.button-green {}
.clear {}

/*OK: 意味があります */
#gallery {}
#login {}
.video {}

/* OK: 一般的な名前です */
.aux {}
.alt {}
```

IDとクラス名のスタイル

IDとクラス名は、できるだけ短くするが必要なだけ長くする

できるだけシンプルなIDとクラス名にするよう心掛けてください。
この方法で名づけをすれば、理解しやすい良レベルのコードに繋がりますし、コードを効率よく書けます。

```
/* NG */
#navigation {}
.atr {}

/* OK */
#nav {}
.author {}
```

タイプレセクター

タイプセレクタによってIDとクラスを制限しない

（例えばヘルパークラスとして使うなど）必要に迫られない限り、IDとクラスの前に要素名をつけてはいけません。
不必要な先祖セレクタを使わないことは、パフォーマンス的にも良いことです。

```
/* NG */
ul#example {}
```

```
div.error {}

/* OK */
#example {}
.error {}
.author {}
```

ショートハンドプロパティ

可能な限りショートハンドプロパティを使う

CSSは様々なショートハンドプロパティ（fontとか）を用意しています。これはたとえ値が一つしかセットされていなくても、できるだけ使うべきです。
ショートハンドプロパティを使うことはコードを読みやすく、効率的にします。

```
/* NG */
border-top-style: none;
font-family: palatino, georgia, serif;
font-size: 100%;
line-height: 1.6;
padding-bottom: 2em;
padding-left: 1em;
padding-right: 1em;
padding-top: 0;

/* OK */
border-top: 0;
font: 100%/1.6 palatino, georgia, serif;
padding: 0 1em 2em;
```

ゼロと単位

「0」の値に続く単位を省く

特に必要でない限り、「0」に単位をつけません。

```
margin: 0;
padding: 0;
```

先頭ゼロ

値から先頭の「0」を省く

-1から1の間の数字では、先頭の0を表しません。

```
font-size: .8em;
```

16進数

可能な限り3文字の16進数を使う

カラー値を3文字で表し得るときは、16進数をより短く簡潔に表現します。

```
/* NG */
color: #eebbcc;
/* OK */
color: #ebc;
```

プリフィックス（接頭辞）

セレクタには、アプリケーションを明確にするプリフィックスを付ける（オプション）

規模の大きいプロジェクト内や、他のプロジェクトまたは外部サイトに埋め込まれたコードには、（ネームスペースとしての）プリフィックスをIDとクラス名に加えます。短く、ユニークな識別子をハイフンでつなぐこと。
ネームスペースを使うことは、名前のコンフリクトを防ぎます。また、たとえば検索・置換機能が使えるように、メンテンナンスを容易にします。

```
.adw-help {} /* AdWords */
#maia-note {} /* Maia */
```

IDとクラス名の区切り文字

IDとクラス名の単語はハイフンで区切る

理解しやすく読みやすいものにするため、ハイフンでセレクターの単語や略語を鎖状につなぎます。（間に何も置かないことも含め）他の文字を使ってはいけません。

```
/* NG: “demo”と“image”が分けられていない */
.demoimage {}
```

```
/* NG: ハイフンの代わりにアンダーバーが使われている */
.error_status {}

/* OK */
#video-id {}
.ads-sample {}
```

ハック

ユーザーエージェント検出や、CSS「ハック」の仕様を避ける―まずは別の手段を探すこと

ユーザーエージェント検知やCSSの特別なフィルター、回避策、そしてハックといった相違点を埋めるスタイリングには魅力があります。効率的でマネジメントしやすいベースコードのためには、これらの手法を最後の手段として熟考すること。
他の手段をとりましょう。プロジェクトとは、楽なほうへ楽なほうへと流れがちなもの。長い目で見ると、検知やハックというフリーパスはプロジェクトを蝕むでしょう。つまり、一度の使用は数度の使用につながり―数度の使用はより頻繁な使用へと繋がっていくからです。

CSS書式のルール

宣言の順番

アルファベット順にする

覚えやすく扱いやすい方法として、矛盾のないコードを達成するためにアルファベット順の宣言を行いましょう。
この順番で並べるとき、ベンダープリフィックスは除外します。しかし、一つのCSSプロパティが複数のベンダープリフィックスを持つ場合は、その中で並び替えます（たとえば、-mozプリフィックスは-webkitの前に来ます）。

```
background: fuchsia;
border: 1px solid;
-moz-border-radius: 4px;
-webkit-border-radius: 4px;
border-radius: 4px;
color: black;
text-align: center;
text-indent: 2em;
```

ブロックコンテンツのインデント

全てのブロックコンテンツをインデントをする

すべてのブロックコンテンツはインデントします。これはルールごとはもちろん宣言ごとに適応され、上下関係を反映しわかりやすいコードを作ります。

```
@media screen, projection {

  html {
    background: #fff;
    color: #444;
  }

}
```

宣言の終わり

すべての宣言の後セミコロンをつける

一貫性・拡張性という視点から、すべての宣言の終わりにセミコロンをつけます。

```
/* NG */
.test {
  display: block;
  height: 100px
}

/* OK */
.test {
  display: block;
  height: 100px;
}
```

プロパティ名の終わり

プロパテイ名のコロンの後ろにスペースを入れる

一貫性を保つため、常にプロパティと値の間にシングルスペースを入れます（しかし、プロパティとコロンの間には入れない）。

```
/* NG */
```

```
.test {
h3 {
  font-weight:bold;
}

/* OK */
h3 {
  font-weight: bold;
}
```

宣言ブロックのセパレーション

最後のセレクターと宣言ブロックの間にはスペースを入れる

最後のセレクターと宣言ブロックの最初のカッコ”{“との間に常にスペースを入れます。
最後のセレクタと”{“は改行せず、同じラインに書くように。

```
/* NG: スペース忘れ */
#video{
  margin-top: 1em;
}

/* NG: 不必要な改行 */
#video
{
  margin-top: 1em;
}

/* OK */
#video {
  margin-top: 1em;
}
```

セレクターと宣言ブロックのセパレーション

複数のセレクターと宣言ブロックは新しい行で区切る

複数のセレクタそれぞれと、宣言ブロックは常に新しい行で書き出します。

```
/* NG */
a:focus, a:active {
```

```
  position: relative; top: 1px;
}

/* OK */
h1,
h2,
h3 {
  font-weight: normal;
  line-height: 1.2;
}
```

ルールのセパレーション

ルールごとに新しい行で区切る

ルールの間は常に一行開けます（つまり2回改行します）。

```
html {
  background: #fff;
}

body {
  margin: auto;
  width: 50%;
}
```

CSSクオテーションマーク

属性セレクターとプロパティ値にシングルクオテーションを使う

ダブルクオテーション(“”)より、シングルクオテーション(’)を使いましょう。URI値（URL()）にクオテーションは使えません。
例外：もし@charsetを使う必要があれば、ダブルクオテーションを使いましょう…シングルクオテーションは使えません。

```
/* NG */
@import url("//www.google.com/css/maia.css");

html {
  font-family: "open sans", arial, sans-serif;
}
```

```
/* OK */
@import url(//www.google.com/css/maia.css);

html {
  font-family: 'open sans', arial, sans-serif;
}
```

CSS書式のルール

セクションのコメント

コメントによってセクションを分ける（オプション）

もし可能なら、 コメントを使ってスタイルシートのセクションをグループ化しましょう。それぞれのセクションを新しい行で区切ります。

```
/* Header */

#adw-header {}

/* Footer */

#adw-footer {}

/* Gallery */

.adw-gallery {}
```

終わりの言葉

一貫性を保て
もしあなたがコードを編集しているなら、周辺のコードを熟視する時間をとり、スタイルを決定しなさい。すべての算術演算子にスペースが使われていれば、あなたもそうしなさい。コメントの周りにハッシュマークで小さなボックスを作っていれば、あなたのコメントにも同じようにハッシュマークボックスをつけなさい。
スタイルガイドを採用するポイントは、コーディングの共通ボギャブラリーを持つこと。そうすれば、メンバーはコードがどう使われているかではなく、何を意味しているかということに集中して取り組めます。私たちはこのドキュメントで、みなさんが語彙を得るためのグローバルなスタイルルールを提供しますが、ローカルスタイルもまた重要なのです。もしあなたが加えたコードが現在のコードと大いに違っていれ

ば、読み手を彼らのリズムから外してしまうでしょう。それは避けるべきです。

訳してみたまとめ

やっと終わった…。おそらく必要最低限のルールなのでしょうが、全部読むと結構な量でした。
HTMLのガイドラインと同じように、CSSもまたとても丁寧に理由が書かれています。終わりの言葉が心に迫り、今までの自分の所業に土下座したくなります（^^;）
他のコーダーを（そして未来や過去の自分を！）考えながら、統一性を大切にコードを書いていこうと思います。

« Google HTML/CSS Style Guideを全部日本語に訳してみた【HTML編】
ノンプログラマーでも怖くない！書評『Web製作者のためのGitHubの教科書』 »

カテゴリー

WEBデザイン
Photoshop
コーディング
HTML5
CSS
Javascript
WordPress
concrete5
WEB制作全般
GoogleAnalytics
制作Tips
その他

英語

プロフィール

なみ

1986年生まれのWEBデザイナー。3年間某喫茶店チェーンで働いたのち、WEBの世界に転身。
以来デザインの他コーディング・プログラミングにも興味を持ち、制作業務をこなしています。WEB制作は天職です。 これまで多くのWEB屋さんブログに助けられ、困難を乗り越えてきました。ので、自分も技術の研鑽に励む人たちへ、役に立つ情報を発信したいです。

TAGS

作ってみた 書評 行ってきた

MESSAGE

お名前（必須）

メールアドレス（必須）

コメント（必須）

送信する