About

名前：半田惇志 @assialiholic

会社：株式会社24-7

職種：テクニカルディレクター／エンジニア

主な担当：コーディング・CMS・HubSpot・その他雑用

主な担当②：高垣 楓・佐久間 まゆ・神埼 蘭子

プロデューサーID：166423466

お願い：LIVE中はメールやDM等お控えくださいますよう

よろしくお願いいたします。


Webのための次世代エディタ
Bracketsの教科書
著／半田 悼志
開発しながらスマホの実機確認がしたい！
いちいちコードを探すのがめんどくさい！
エディタのフォントも綺麗な方がいい！
すべて
Bracketsが解決します！
20を超える標準機能から40を超える拡張機能まで、網羅的に紹介！
ブックマーク・メモ・ハイライト完全対応の画像付き決定版解説書

書きました。

つくりました。

なぜ作ったのか？

今までの設計の悩み

1.命名規則が
ややこしい

item-block__item-title_state_current

ハイフン？
アンスコ？

単語の
区切りは…

オリジナル？
BEM風？

item-block__item-title_state_current

1つ？
2つ？

モディファ
イアは…

MindBEMdingにのっとって
「オレオレBEM」を書く人も。
ただしドキュメントはない（ゝω・）ﾃﾍﾍﾟﾛ☆

今までの設計の悩み

2. モジュールの大きさに悩む

モジュールの大きさに悩む

全体を1つのモジュールにする？

OOCSSやSMACSS、BEMの素晴らしさ巧みに取り入れ
更に進化させた強力なモジュール設計。それがPRECSSです。

全てに接頭辞を付加

PRECSSの管理下にあるクラスは、全て種類に応じた接頭辞が付加されます。これにより接頭辞を一目見ただけで役割、スコープ、依存の全てを把握することができます。
もうモジュールの大きさに悩まないでください。

最適化された命名規則

クラス名は全てスネークケースとキャメルケースの混成で構成され、それらの使い分けには明確なルールがあります。
省略語についても指針を示しているため、今までのように悩んだあげく、とても長い名前を付ける必要はもうありません。

親しみやすい設計

PRECSSは全く新しい設計思想ではありません。OOCSSやSMACSS、BEMなど今までの賞賛すべき素晴らしい思想が基になっています。
あなたがモダンな開発者であるほど、PRECSSは親しみやすいものに感じるでしょう。

他種族との共存

明確な独自記法により、あなたが書いたクラスとCMSやCSSフレームワークが出力したクラスを明確に区別することができるでしょう。
全てPRECSSのルールに従わなければならないのではなく、他の者も受け入れる柔軟さがPRECSSにはあります。

≤ コーディングルール

ドキュメントが長いと感じますか？今までのCSSが破綻した原因は、ドキュメントの短さにありました。
コーディングルールのベースになることを目指しているPRECSSは、ときにとても細かい話もします。
誰が書いても、なるべく破綻しないために。

2カラムと3カラムで分ける？

更に分解して大きさで区別する？

今までの設計の悩み

3.影響範囲が
わかりづらい

item-block

item-block

これは編集していいのか…？

他に何処で使っている…？

スタイルガイドは無いし…

プロジェクト全体に横断検索かけるしか…

SaaSだから手元にファイル無い＼(^o^)／

今までの設計の悩み

4.組み込みCSSとの見分けがつかない

昨今の開発事情

既にクラスを持っている人たち

etc.

既にクラスを持っている人たち(テーマ)

etc.

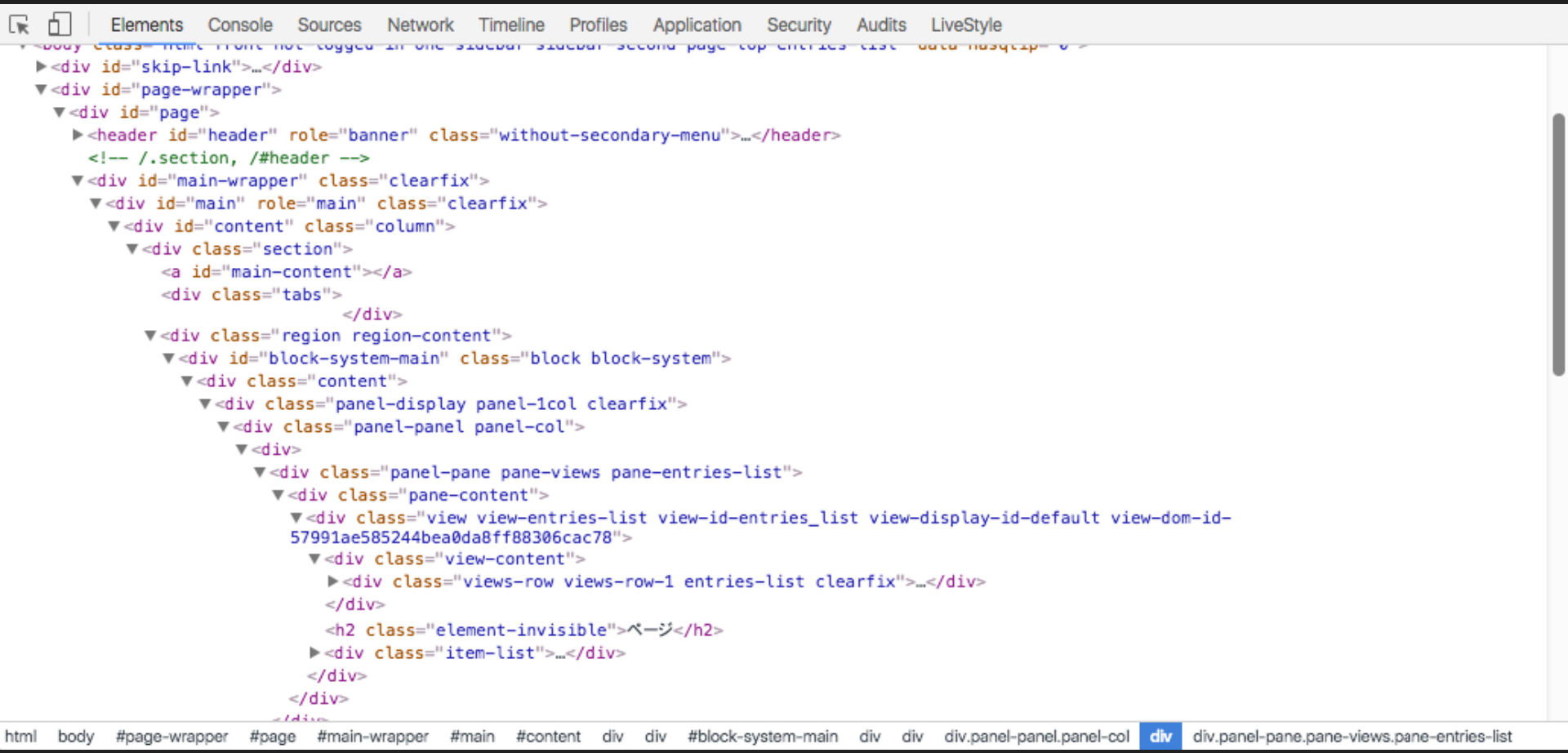
かつ、HTMLも吐き出す人たち

etc.

HTMLを部分的に吐き出す人たち

etc.

命名の全てを制御できる
サイト環境は、今後徐々に
少なくなっていきます。

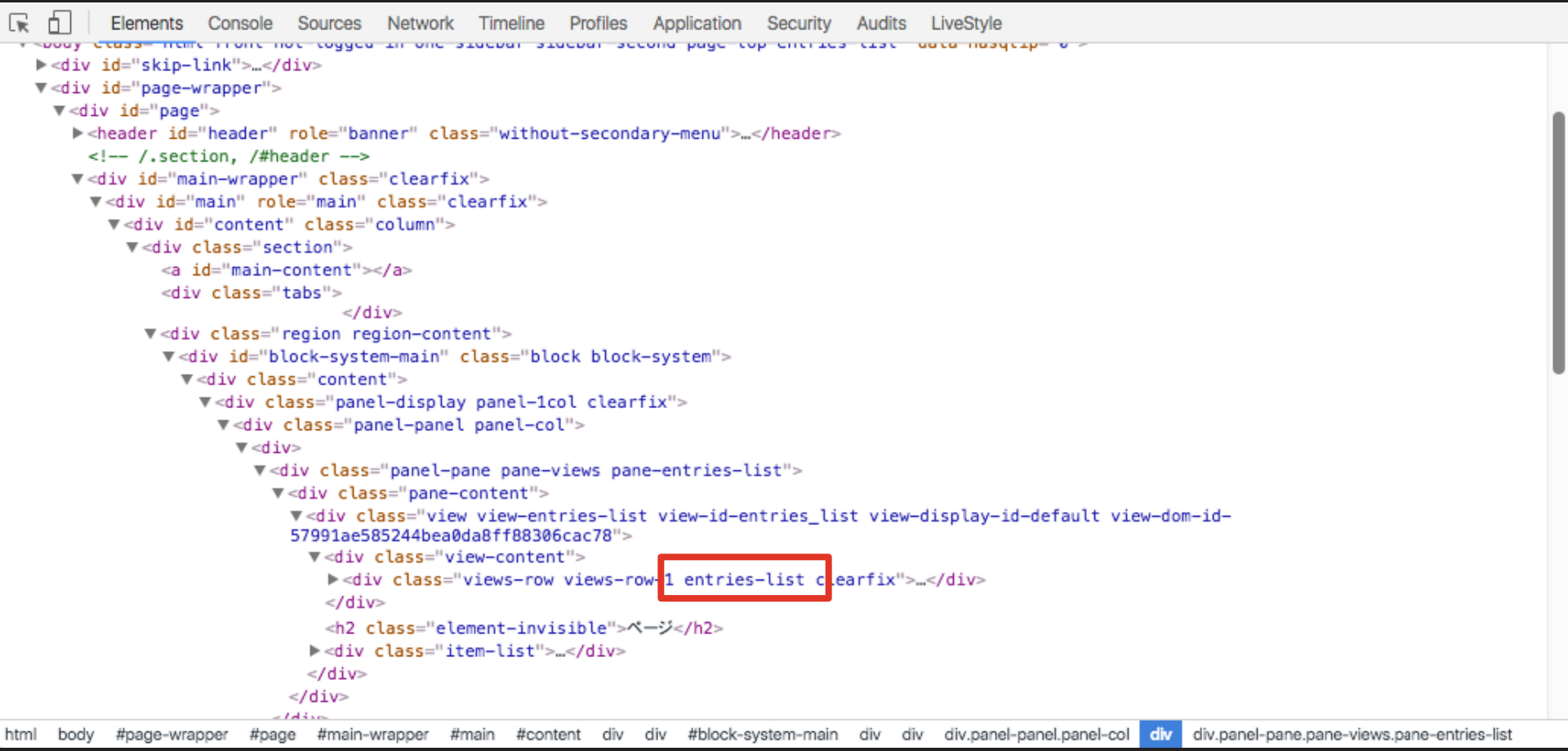

Elements Console Sources Network Timeline Profiles Application Security Audits LiveStyle
…

<header id="header" role="banner" class="without-secondary-menu">…</header>
<!-- /.section, /#header -->

…

<h2 class="element-invisible">ページ</h2>
…

html body #page-wrapper #page #main-wrapper #main #content div div #block-system-main div div div.panel-panel.panel-col div div.panel-pane.pane-views.pane-entries-list

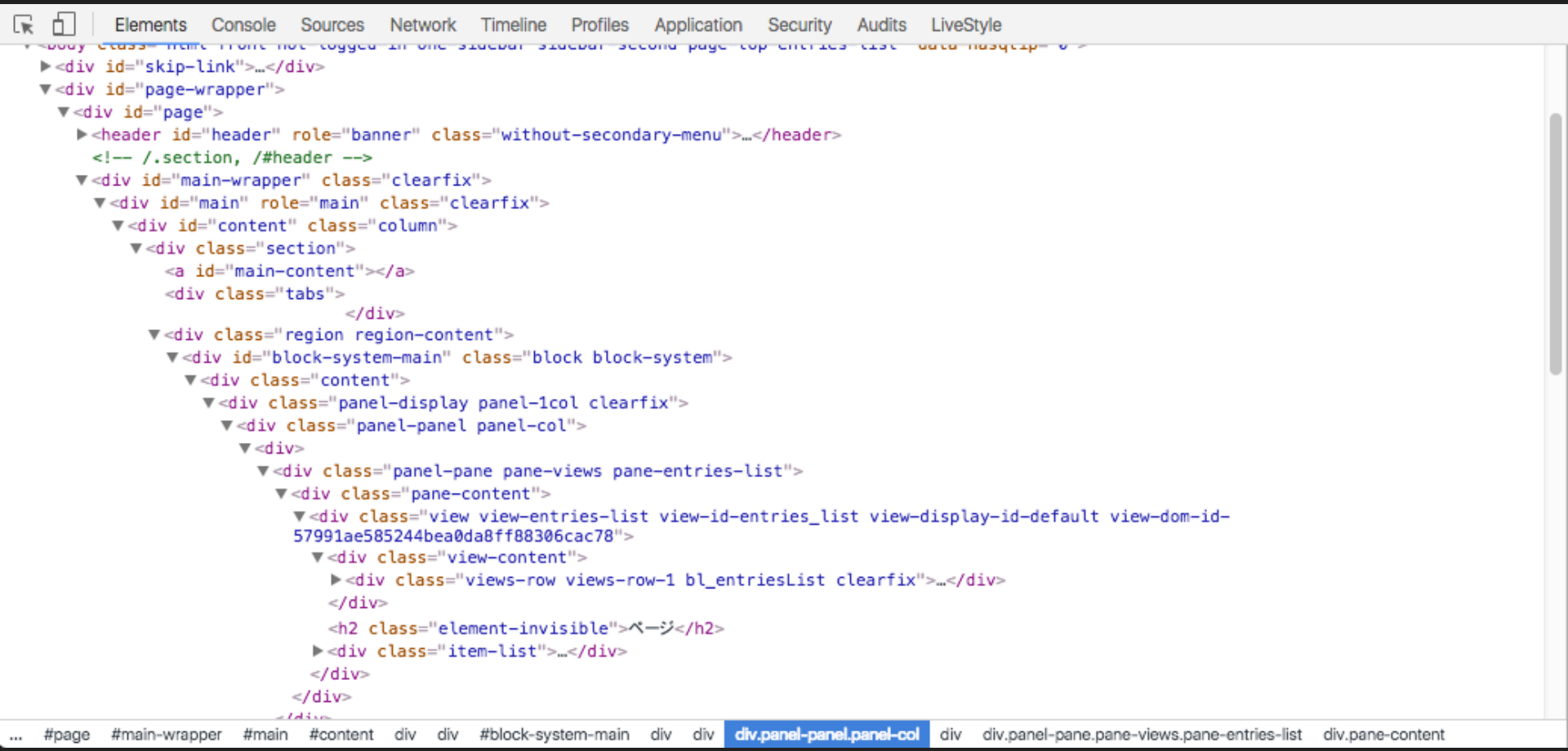

Elements Console Sources Network Timeline Profiles Application Security Audits LiveStyle
…

<header id="header" role="banner" class="without-secondary-menu">…</header>
<!-- /.section, /#header -->

…

<h2 class="element-invisible">ページ</h2>
…

html body #page-wrapper #page #main-wrapper #main #content div div #block-system-main div div div.panel-panel.panel-col div div.panel-pane.pane-views.pane-entries-list

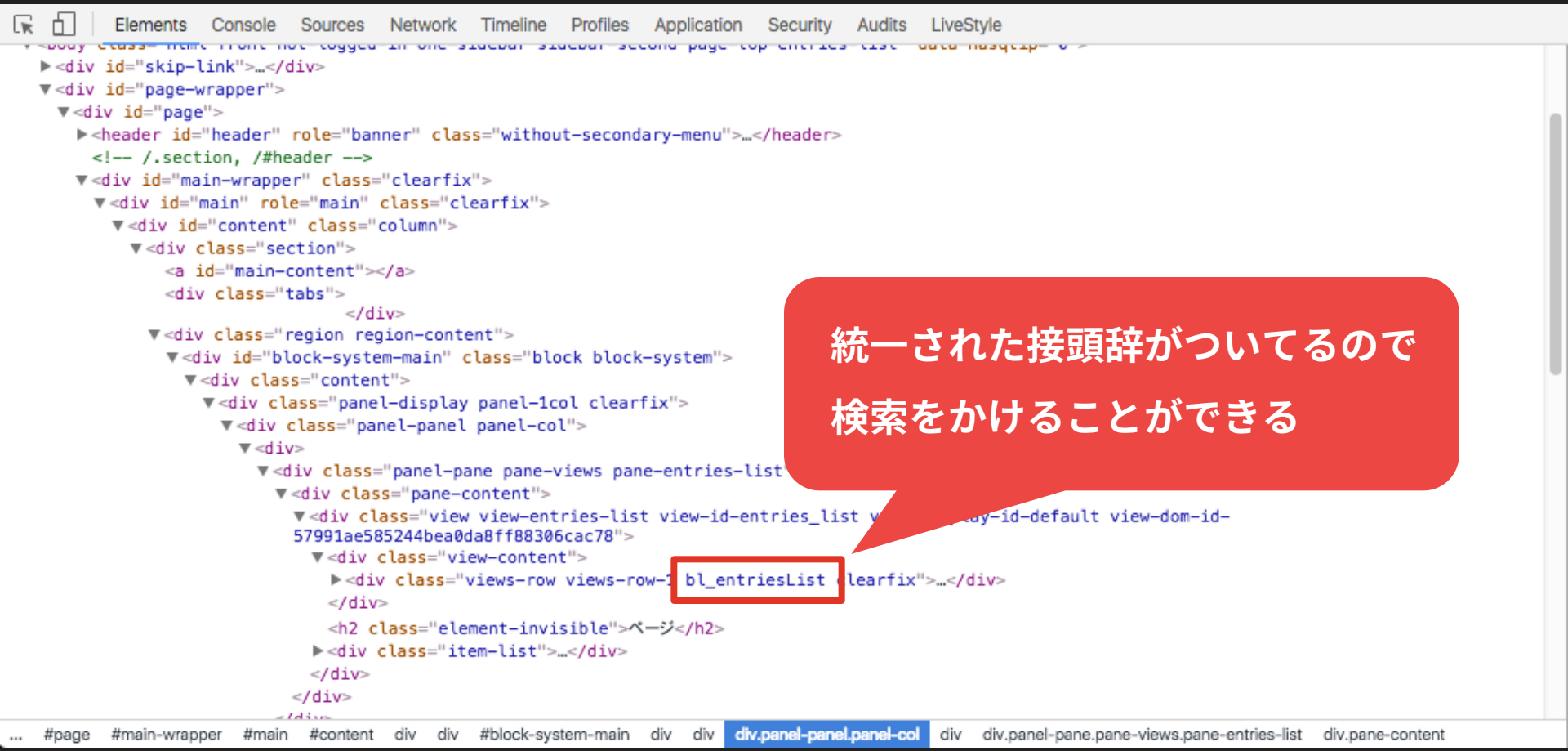

Elements Console Sources Network Timeline Profiles Application Security Audits LiveStyle
…

<header id="header" role="banner" class="without-secondary-menu">…</header>
<!-- /.section, /#header -->

…

<h2 class="element-invisible">ページ</h2>
…

... #page #main-wrapper #main #content div div #block-system-main div div div.panel-panel.panel-col div div.panel-pane.pane-views.pane-entries-list div.pane-content


Elements Console Sources Network Timeline Profiles Application Security Audits LiveStyle
…

<header id="header" role="banner" class="without-secondary-menu">…</header>
<!-- /.section, /#header -->

<div class="panel-pane pane-views pane-entries-list

<div class="view view-entries-list view-id-entries_list
57991ae585244bea0da8ff88306cac78">

…

<h2 class="element-invisible">ページ</h2>
…

統一された接頭辞がついてるので
検索をかけることができる
... #page #main-wrapper #main #content div div #block-system-main div div div.panel-panel.panel-col div div.panel-pane.pane-views.pane-entries-list div.pane-content

結局

自分の中で経験則からの最適解が
なんとなくあればいいんですが、
他人に渡したときにそれを保てるのか？

（特にモジュールの大きさとか）

New CSS architecture
PRECSS

OOCSS + SMACSS + BEM

CONCEPT

Manage your CSS with prefixes.

シンプルな命名規則

bl_itemBlock_img__large

役割に応じた
接頭辞

子要素は
アンスコ

bl_itemBlock_img__large

単語結合は
キャメルケース

モディファイア
は2個

明確な5つのグループ

1.ベース

1.ベース

reset.cssやnormalize.css等による素地作りの他、サイト全体にまつわるスタイルを要素セレクタに直接指定。

SMACSSと同様。

接頭辞：なし

2.レイアウト

2.レイアウト

ヘッダー、ボディエリア、メインエリア、サイドエリア、フッター等の大きなレイアウトを形成する要素に使用。

接頭辞：ly_

.ly_header

.ly_body

.ly_footer

3-1.ブロックモジュール

3-1.ブロックモジュール

特有の複数の子要素や、エレメントモジュールを内包し、一つの塊として持ち運び可能なモジュール群を形成。

単体でのwidthは100%であることが望ましい。（カラムを形成する際はラップ用のモジュールを作る）

使い回すことが前提のため、運用時に既存のモジュールは修正しないことを推奨。（影響範囲を把握している場合を除く）

接頭辞：bl_

.bl_itemBlock

.bl_sectBlock

.bl_sectBlock__red

全て同じモジュールです。

.bl_featureBlock

それぞれ親モジュールでラップしてレイアウト指定

.bl_featureUnit__col2
> .bl_featureBlock

.bl_featureUnit__col3
> .bl_featureBlock

3-2.エレメントモジュール

3-2.エレメントモジュール

ボタンやラベル、見出し等の最小単位のモジュールで、単体で持ち運ぶことが可能。

使い回すことが前提のため、運用時に既存のモジュールは修正しないことを推奨。（影響範囲を把握している場合を除く）

接頭辞：el_

.el_title

.el_button

.el_lable

4.ヘルパー

4.ヘルパー

基本的に1つのスタイルのみで、意図的な上書きのため!importantを付加。

命名規則は基本的にEmmetのショートハンドに準ずる。

接頭辞：hp_

.hp_w400

.hp_mb20

.hp_fz2e

5.プログラム

5.プログラム

JavaScript等のプログラムで要素にタッチする際の専用クラス。

要素取得のためのjs_と、状態管理のためのis_の2つに分かれる。

接頭辞：js_ & is_

.js_carousel

.js_accordion

.js_accordion.is_active

One more thing

6.ユニーク

↑New!

6.ユニーク

1つのページでしか使用しないスタイル。ユニークであるため、改修の際に影響範囲を気にしなくてよい。モジュールの粒度も自由。

濫用し過ぎると拡張性に欠けるため、バランスは考慮する必要あり。

（策定中）

接頭辞：uq_

.uq_topPageBlock

.uq_sectionTitle

.uq_topPageBlock

.uq_sectionTitle

トップページでしか使わない
特殊なスタイル

汎用名であっても
臆せず編集してよい

よきアーキテクチャとは？

- 予測しやすい
- 再利用しやすい
- 保守しやすい
- 拡張しやすい

- 予測しやすい
- 再利用しやすい
- 保守しやすい
- 拡張しやすい
- 把握しやすい ←New!

把握しやすい

- **どのような役割を担っているのか？**

  - レイアウトを形成してるの？モジュール？大きさは？

- **誰が書いたCSSなのか？**

  - 既存テーマCSSなの？プロジェクトで書いたCSSなの？

- **どこに影響するのか？**

  - ユニークの場合は何も考えずに編集してOK

接頭辞が、これらの
「把握しやすさ」を
実現しています。

人によるブレが少ないよう
ルールが厳格・シンプルであること

かつ、他のCSSとも十分に
共存でき見分けやすいこと

イレギュラーのための
逃げ道もあること

オリジナルの接頭辞グループを作ってもOK（.sp_→スマホ用 .tb_→タブレット用）

迷ったらとりあえずユニークを使ったらOK（後から誰でも、いつでも直していい目印）

固さと柔らかさ

どちらもあります。

CSSって泥沼だと思いませんか？

その泥沼で咲いた花、

それがPRECSSです。

(後付けなんだけどね)

おためしください。

http://precss.io/

Prefixes with you.