株式会社 ITS MORE
2020年4月設立

Digitally signed by 佐藤 豊
Date: 2020.06.19 18:01:57 +09'00'

2020年6月19日 投稿者: SATOXITS

今どきの Context Reflective CSS

開発：どうも納得できないのは、本当にCSSでは定数名的なものとかマクロ的なものが使えないのか？ということです。

開発：そもそも統一感のあるテーマであれば、使用する色の数なんてたかが知れていて、まあ数種類ということろです。フォント名にしてもしかり。

開発：もうひとつは条件による選択。どういうコンテクストであったらこういう表現にする、って出来て当然だと思うわけです。

社長：それはひょっとすると、マクロ処理をして返すっていう処理系がどこかのURLて提供されてれば済むような気もしますね。それならCSSだけでなく、何にでも適用できる。私達的にはCのプリプロセッサが馴染みがあるけど、歴史をたどれば m4 というマクロにはすごく感動した記憶があります。40年も前ですけどね。

開発：外部のマクロ処理系を通すっていることになると、サーバ側の設定が必要になりますよね。オフラインでは使えないというのが引っかかります。まずは CSS の中だけでなんとか出来ないかと。

基盤：検索しましょう。conditional css ・・・ 一撃ヒットです。ワン・イシューごとにドメイン取るのって、社長と同じのりですね。

Conditional-CSS [conditional-css.com]

開発：うーん、やっぱりブラウザによってCSSを変えるってだけの話ですね。しかも、CSS自体の機能ではないから、別途処理系が必要。

基盤：では、css constant definition とか・・・一撃ヒットです。

Using CSS custom properties (variables) [mozilla.com]

> **Custom properties** (sometimes referred to as **CSS variables** or **cascading variables**) are entities defined by CSS authors that contain specific values to be reused throughout a document. They are set using custom property notation (e.g., `--main-color: black;`) and are accessed using the `var()` function (e.g., `color: var(--main-color);`).

開発：なーんと、やっぱりあるんですね。

基盤：でこのページの最後を見ると、IEだけはサポートしてないですが（笑）。さらにそこに、env() というのが引用されていたりします。

env() [developer.mozilla.com]

> The **env()** CSS function can be used to insert the value of a **user agent-defined environment variable into your CSS**, in a similar fashion to the `var()` function and custom properties. The difference is that, as well as being user-agent defined rather than user-defined, environment variables are globally scoped to a document, whereas custom properties are scoped to the element(s) on which they are declared.

開発：なーんと、やっぱりあるんですね。

基盤：さらに続けて。

> To tell the browser to use the whole available space on the screen, and so enabling us to use the **env()** variables, we need to add a new viewport meta value:

```
<meta name="viewport" content="viewport-fit=cover" />
```

開発：なーんと、これでしたか。viewport って、以前のブログでモバイル対応について書いた時のあれですね。あの時のはこうでした。

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

社長：スマホを横にしたり縦にしたりした時に自動的にフィットしてくれるやつって、これの仲間ですかね？

開発：いやしかし、やはりお天道様は正しい方を照らしているてわけですか。古典的なCSSの苦し紛れ技法の勉強とか自作のごにょごにょとかしないで済みました。WordPressにも依存しない。あ、でもマクロプロセッサっていうサービスはあるといいかな。

基盤：原典は WSC の CSS ホームらしいですが、MDNのはいい感じでサマライズしているって感じですね。MDNって廃屋かと思ってましたが、それは mozilla の実装管理とかの一部分なのかも知れません。

開発：いずれにしてもネット検索って古めの解説とかがバリバリ上位に出てきてしまうから、やはり原典に近いところにあたるのが近道ですね。というか、作成・更新日付で検索を絞ることができないのが、いかに不便かということです。

基盤：あと、CSSのユーザ寄りの情報よりも、ブラウザ（CSS/HTML処理系）を作ってる側に近いところの情報が旬のようです。

社長：とても明るい気分になったので、お昼に行きましょう。

＊ ＊ ＊

社長：さて、じゃあどこから手を付けますかね。

開発：お昼から帰ってしばらく調べたのですが。これ、CSSの拡張機能、やるのやめようかと思います。

社長：おや、そうですか・・・　なぜ？

開発：かったるいからです。非常に簡単なことをやるのに大変な労力とリソースを食う。頑張っても果実が適用できるのはCSSだけ。そもそもそのCSSの拡張方法が見た目に美しくない。その拡張機能はまだ2020年現在ドラフト段階で今後どう転ぶかはわからない。ブラウザによってはデフォルトでオフにされてることもある。やめましょう。われわれの活動指針は more with less です。

社長：そうですか・・・　それで代替案は？

開発：社長の言ったマクロ案を採用します。言語的には伝統と実績のC言語プリプロセッサ。コメントはCSSと同じ/* */。処理系はPython。Python なら Linux、Mac に標準装備、Win でも簡単にインストールできました。幸いCプリプロセッサをPythonで書いてくれている人もいます。http://server/cpp?URL とするとそのURLの内容をCプロセッサで処理した結果を返す。この場合マクロ処理済のCSSです。これを<link type=stylesheet http://server/cpp?URL> で参照する。これでいけるのではないかと思います。そもそもわたしはもともと、簡単な HTMLの生成も、CGIとかSSIとかJavaScriptでなく、マクロでやれば良いのでは無いかと思って来ました。

基盤：HTTPヘッダの値とかが、環境変数じゃなくて predefined として見えるって感じですね。

社長：そうですか。うーん、それも面白そうです。ただひとつ気になる点。昨日WordPressのソースを見ましたが、style.css の取り込みはこうなってました。

```
<link rel='stylesheet' id='twentyseventeen-style-css'
  href='https://its-more.jp/ja_jp/wp-content/themes/twentyseventeen/sty
```

社長：これを見た時、ver=20190507 の部分は sytle.css の中でスイッチとして使われてるんだろうと思ったわけですが。これを div[“ver=20190507”]みたいに使えるのかなとか、今日の話だと$(env)とかに使えるのかなって。

基盤：わたしもそう思ったんですが、少なくとも styles.css の中には ver とか 20190507 は出てきません。

開発：なんにしろ、たとえばコメントというものが lexical level で事前処理しないと大変なように、syntax level で全てやろうとすると、構文的に非常に面倒なことになります。別の次元から断面を見ると簡単に見える。プリプロセッサはそういうものだと思います。プリプロセッサという別フェイズで展開されたステートメントと最終的な実行を対応付けるのが大変ということは発生します。ですがこの場合、インタプリタたるCSS処理系とかインスペクタは、大元の記述の構造とか紐付けが見えにくくて、まさにマクロ的に redefine を重ねてるだけのように見えるんです。だから、事態は現状より悪化はしない。

社長：そうですか。じゃま、マクロの線でやってみますか。

開発：あ、それはそうと、わたしはさっきはじめて、CSSの基本中の基本というページを見たのですが、この selector に関するテーブルはまさに、ああこういうものを見たかった、というサマリーでした。

Selector name	What does it select	Example
Element selector (sometimes called a tag or type selector)	All HTML elements of the specified type.	`p` selects `<p>`
ID selector	The element on the page with the specified ID. On a given HTML page, each id value should be unique.	`#my-id` selects `<p id="my-id">` or `<a id="my-id">`
Class selector	The element(s) on the page with the specified class. Multiple instances of the same class can appear on a page.	`.my-class` selects `<p class="my-class">` and `<a class="my-class">`
Attribute selector	The element(s) on the page with the specified attribute.	`img[src]` selects `<img src="myimage.png">` but not `<img>`
Pseudo-class selector	The specified element(s), but only when in the specified state. (For example, when a cursor hovers over a link.)	`a:hover` selects `<a>`, but only when the mouse pointer is hovering over the link.

There are many more selectors to discover. To learn more, see the MDN Selectors guide.

Selector	Example	Learn CSS tutorial
Type selector	`h1 { }`	Type selectors
Universal selector	`* { }`	The universal selector
Class selector	`.box { }`	Class selectors
id selector	`#unique { }`	ID selectors
Attribute selector	`a[title] { }`	Attribute selectors
Pseudo-class selectors	`p:first-child { }`	Pseudo-classes
Pseudo-element selectors	`p::first-line { }`	Pseudo-elements
Descendant combinator	`article p`	Descendant combinator
Child combinator	`article > p`	Child combinator
Adjacent sibling combinator	`h1 + p`	Adjacent sibling
General sibling combinator	`h1 ~ p`	General sibling

開発：スタイルの上書きルールについても確認できました。

1. Cascading order ¶

The term “cascading” means hierarchical order in which different style sheet types interact when two styles come into conflict. The conflict occurs when two different styles are applied to the same element.

For these cases, there exists an order for style sheets according to their priority (4 has the highest priority):

- Browser Defaults.
- External Style Sheets (Linked or Imported).
- Internal Style Sheets (Embedded).
- Inline Styles.

So, it means that when a conflict arises between two styles, the last one used takes precedence. To make it clearer, you should remember these two rules:

- You must place inline styles in the <body> of the HTML document, while embedded style sheets must be placed in the <head> of the HTML document so that the inline styles will always be the last used ones and therefore they will take precedence.

開発：最後に、ずっと気になっていた、<body> の中に <style> を記述することの可否についてですが、Stackoverflow で見つけたのは、やはりそれは良くないことだが実際には使えるという話。そして、HTML 5 では「scoped 属性」を使えば、正々堂々と使える、ということでした。

社長：scoped ていわゆる普通のプログラミング言語のスコープのことですか？

開発：そうですね。たとえばある div の中にだけ適用される style タグを、その div の中に定義できる、そう宣言するためにその style に scoped という属性をつける、ということです。ただこの属性、賛否があるようで、HTMLの仕様から外されたり復活したりをしているようです。たぶん、今は外されています。

```
<div>
  <style scoped> div { color:blue; } </style>
  これは青のはず
  <div>
    <style scoped> div { color:green; } </style>
    これは緑のはず
  </div>
</dvi>
```

開発：そういうことに巻き込まれたくないので、もう class とか type とかのセレクタでいいかなって思います。これも内側に定義できるということに気づきました。ようするに、どこに書いてもグローバル、スコープが無いのが好きなんですねCSSの人たちは。

```
<div class=div1>
  <style> .div1 { color:blue; } </style>
  これは青のはず
  <div class=div2>
    <style> .div2 { color:green; } </style>
    これは緑のはず
  </div>
</div>
```

これは青のはず

これは緑のはず

社長：完全にレンダリング用の環境を作ってから、HTMLに適用したい、って気分なんですかね。まあ、そういう意味では、プリプロセッサ的発想なのかも。CSSの処理とHTMLの処理を分離したいというか。HTMLの構造から独立したいとか。

基盤：ですが、そしたらタグの中の属性の style って何なのよって気もしますが。

開発：なんしてもあれですよね、これだけ世の中を振り回す規格なのに、どういう理由でそう決めたのか、rationale というか、ほとんど説明されてないですよね。なんか、varとenvなんかW3Cのエディタのページにいってもすごくサブイ感じだし。

— 2020-0619 SatoxITS

今どきの-Context-Reflective-CSS-–-株式会社-ITS-more