
Pattern Driven CSS

Structure of page design

Toru Yamaguchi <zigorou@cpan.org>


Yet Another Hackadelic (d:id:ZIGOROu)



アジェンダ

- Pattern Driven CSSとは？
- ページレイアウトとCSS
- CSSの構造と適用
- パターンとレイアウトの分離
- その他

Pattern Driven CSSとは？



Pattern Driven CSSとは？

Web design pattern (1)

■ Web design patternとは？

- ➡ ページの構成要素をパターンとみなす。
- ➡ パターンを再利用可能な形とする
- ➡ Welie.com, Yahoo, 37signals

<http://www.welie.com/patterns/>

<http://developer.yahoo.com/ypatterns/>

<http://www.37signals.com/papers/introtopatterns/>

Web design pattern (2)

■ Directory Navigation in welie.com

<http://www.welie.com/patterns/showPattern.php?patternID=directory>

➡ ディレクトリサーチ
型のパターン

➡ 言わんとする事は分
かるケドも...

実際どうマークアップし
て行くか分からん

Business & Economy

[B2B](#), [Finance](#), [Shopping](#), [Jobs](#)...

Computers & Internet

[Internet](#), [WWW](#), [Software](#), [Games](#)...

News & Media

[Newspapers](#), [TV](#), [Radio](#)...

Entertainment

[Movies](#), [Humor](#), [Music](#)...

Recreation & Sports

[Sports](#), [Travel](#), [Autos](#), [Outdoors](#)...

Health

[Diseases](#), [Drugs](#), [Fitness](#), [Medicine](#)...

Government

[Elections](#), [Military](#), [Law](#), [Taxes](#)...

Regional

[Countries](#), [Regions](#), [US States](#)...

Society & Culture

[People](#), [Environment](#), [Religion](#)...

Education

[College and University](#), [K-12](#)...

Arts & Humanities

[Photography](#), [History](#), [Literature](#)...

Science

[Animals](#), [Astronomy](#), [Engineering](#)...

Social Science

[Languages](#), [Archaeology](#), [Psychology](#)...

Reference

[Phone Numbers](#), [Dictionaries](#), [Quotations](#)...

Web design pattern (3)

■ Markup Directory Navigation (1)

Arts & Humanities

Photography, History, Literature...

Business & Economy 2B, Finance, Shopping, Jobs...	Regional Countries, Regions, US States...
Computers & Internet Internet, WWW, Software, Games...	Society & Culture People, Environment, Religion...
News & Media Newspapers, TV, Radio...	Education College and University, K-12...
Entertainment Movies, Humor, Music...	Arts & Humanities Photography, History, Literature...
Recreation & Sports Sports, Travel, Autos, Outdoors...	Science Animals, Astronomy, Engineering...
Health Diseases, Drugs, Fitness, Medicine...	Social Science Languages, Archaeology, Psychology...
Government Elections, Military, Law, Taxes...	Reference Phone Numbers, Dictionaries, Quotations...

➡ 骨格はdlで表現

➡ サブカテゴリはulで表現

Web design pattern (4)

■ Markup Directory Navigation (2)

```
<dl class="directory-navigation">
  <dt><a href="">Arts & Humanities</a></dt>
  <dd>
    <ul>
      <li><a href="">Photography</a></li>
      <li><a href="">History</a></li>
      <li><a href="">Literature</a></li>
    </ul>
  </dd>
</dl>
```

Web design pattern (5)

■ Markup Directory Navigation (3)

➡ class指定でパターン内のスタイル指定を縛る

➡ このパターン内のスタイルに関しては `div.directory-navigation>dl` のようにしてスタイルを当てる。

このパターンが実際どのように配置されるかをどう表現する？

Web design pattern (6)

Layout Directory Navigation (1)

Container

navigation (1)

navigation (2)

navigation (3)

navigation (4)

navigation (5)

navigation (6)

Business & Economy
B2B, Finance, Shopping, Jobs...

Computers & Internet
Internet, WWW, Software, Games...

News & Media
Newspapers, TV, Radio...

Entertainment
Movies, Humor, Music...

Recreation & Sports
Sports, Travel, Autos, Outdoors...

Health
Diseases, Drugs, Fitness, Medicine...

Government
Elections, Military, Law, Taxes...

Regional
Countries, Regions, US States...

Society & Culture
People, Environment, Religion...

Education
College and University, K-12...

Arts & Humanities
Photography, History, Literature...

Science
Animals, Astronomy, Engineering...

Social Science
Languages, Archaeology, Psychology...

Reference
Phone Numbers, Dictionaries, Quotations...

Web design pattern (7)

■ Layout Directory Navigation (2)

- ➡ 各パターンをコンテナにレイアウトして行く時にどうしてもfloatが必要そう
- ➡ first-child, last-child, even, oddで何かありがち

Web design patternだけでは実践的じゃないのでは無いか？

Pattern Driven CSS

■ Pattern Driven CSSとは？

- ➡ レイアウトを無視したパターンを作っていく
- ➡ レイアウトの当て込みはそのコンテナ別に別途行う
- ➡ コンテナの実際の配置によってページが出来上がる

平たく言えばこういうスタイルの書き方

ページレイアウトとCSS



ページレイアウトと CSS

Page layout of web site (1)

■ ありがちなページ構成 (1)

➡ 2カラムまたは3カラム

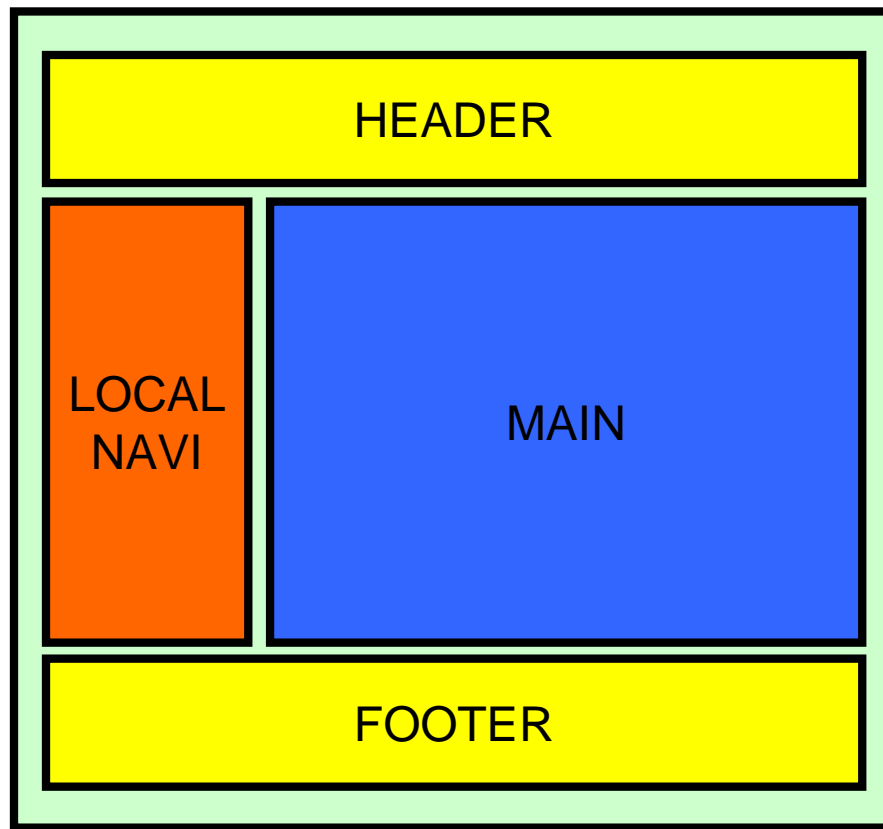
➡ ヘッダ、フッタ、両サイドはほぼ共通

➡ 残りの部分がユニークなページ

こうしたレイアウトは当然CSSの設計にも密接に関係してくる！

Page layout of web site (2)

■ ありがちなページ構成 (2)



➡ こういうのありがち

➡ だからパターン化しやすい

➡ liquid designを意識して考える

Style injection (1)

■ Style injection (1)

Container (2column)

navigation (1)

navigation (2)

navigation (3)

navigation (4)

navigation (5)

navigation (6)

Business & Economy
B2B, Finance, Shopping, Jobs...

Computers & Internet
Internet, WWW, Software, Games...

News & Media
Newspapers, TV, Radio...

Entertainment
Movies, Humor, Music...

Recreation & Sports
Sports, Travel, Autos, Outdoors...

Health
Diseases, Drugs, Fitness, Medicine...

Government
Elections, Military, Law, Taxes...

Regional
Countries, Regions, US States...

Society & Culture
People, Environment, Religion...

Education
College and University, K-12...

Arts & Humanities
Photography, History, Literature...

Science
Animals, Astronomy, Engineering...

Social Science
Languages, Archaeology, Psychology...

Reference
Phone Numbers, Dictionaries, Quotations...

Style injection (2)

■ Style injection (2)

➡ パターンは自分が含まれるコンテナに応じて如何なるレイアウトにも対応するようパターン化すべきである。

パターンのルート要素に当てる新たな属性 (classまたはid)を用意し、セレクトタにコンテナを含めルール化する。

スタイル依存性の抽出？

Style injection (3)

■ Style injection (3)

```
<ul class="double-directories-container">
  <li class="odd first-child"><dl class="directory-
navigation" /></li>
  <li class="even"><dl class="directory-navigation"
/></li>
  <li class="odd"><dl class="directory-navigation"
/></li>
  <li class="even last-child"><dl class="directory-
navigation" /></li>
</ul>
```

Style injection (4)

■ Style injection (4)

➡ `ul.double-directories-
container>li.odd>dl.directory-navigation`

➡ `ul.double-directories-
container>li.even>dl.directory-
navigation`

パターンのレイアウト、サイズ等は**より具
体化した**、これらのセレクタで行う。

Style injection (5)

■ Style injection (5)

➡ 特に大きさや位置を決めるdisplay, position, width, height, margin, padding など

➡ 特に表示の相違を出すfont系、background系など

コンテナで具体化した時に表示も具体化して行く。パターンは極力抽象的にする。

Style inheritance (1)

■ Style inheritance (1)

Container (1column)

navigation (1)

navigation (2)

navigation (3)

買う [ショッピング](#) [共同購入](#) [オークション](#) [コミック](#) [チケット](#) [旅行](#)
[出張宿泊](#) [保険](#) [宅配](#) [ネットバンク](#) [決済](#) [コンテンツストア](#)
知る [ニュース](#) [天気](#) [スポーツ](#) [ファイナンス](#) [政治](#) **NEW!**
楽しむ [映画](#) [音楽](#) [着メロ](#) [ゲーム](#) [占い](#) [懸賞](#) [本](#) [テレビ](#) [動画](#)
[ポッドキャスト](#) [ライブトーク](#)
調べる [辞書](#) [翻訳](#) [地域](#) [地図](#) [路線](#) [道路交通](#) [電話帳](#) [自動車](#)
[コンピュータ](#) [きっず](#) [知恵袋](#)
暮らす [グルメ](#) [クーポン](#) **NEW!** [結婚](#) [恋愛](#) [ビューティー](#) [健康](#)
[不動産](#) **NEW!** [ボランティア](#) [ネット検定](#) [学習](#) [セカンドライフ](#)
求人：[転職](#) [アルバイト](#) [派遣](#) [適職紹介](#) [新卒](#) [独立](#)
集まる [掲示板](#) [チャット](#) [グループ](#) [アバター](#) [ID検索](#) [ホームページ作成](#)
[ブログ](#) [フォト](#) [グリーティング](#) [メルマガ](#) [メッセンジャー](#) [SNS](#)
人気のオークション [ファッション](#) [家電、AV、カメラ](#) [エクササイズ用品](#)
[スノボ](#)

Style inheritance (2)

■ Style inheritance (2)

➡ パターンは自分が含まれるコンテナによってパターン自体のスタイルが変化するのもありうる。

コンテナをセレクタに含めた形でパターン内の要素を上書きして行く。

スタイルの継承？

Style inheritance (3)

■ Style inheritance (3)

```
<ul class="single-directories-container">
  <li class="odd first-child"><dl class="directory-
navigation" /></li>
  <li class="even"><dl class="directory-navigation"
/></li>
  <li class="odd"><dl class="directory-navigation"
/></li>
  <li class="even last-child"><dl class="directory-
navigation" /></li>
</ul>
```

Style inheritance (4)

■ Style inheritance (4)

➡ dl.directory-navigation *はこのパターンに与えられた最も基本的なnamespaceのようなもの

➡ ul.single-directories-container li dl.directories-navigation *はこのコンテナに含まれる場合のnamespaceのようなもの

Page layout and CSS (1)

■ Page layout and CSS

- ➡ ページの構成要素のグループ化が複合されるにつれてセレクタが長くなる
- ➡ 難しくなる。
- ➡ 困難は分割せよ！
- ➡ だからパターンから作って行く！

Page layout and CSS (2)

■ Definition

- ➡ パターンを最小単位とする
- ➡ パターンの連続ないしは複数のパターンの組み合わせをモジュールと呼ぶ
- ➡ パターン自体もモジュールとなりうる
- ➡ モジュールを収める器をコンテナと呼ぶ

id, class名の語彙に反映させる

CSSの構造と適用

CSSの構造と適用

今までの簡単なまとめ (1)

■ 今までの簡単なまとめ (1)

- ➡ パターンはそれぞれのnamespaceのような物を持つ
- ➡ モジュール化によってさらにnamespaceを持つ
- ➡ モジュール化によって抽出されるnamespaceも出来る

今までの簡単なまとめ (2)

■ 今までの簡単なまとめ (2)

要素に直接スタイルを定義



パターン別にスタイルを定義



モジュール別にスタイルを定義



ページ別にスタイルを定義

抽象化・単純・共通

実装の向き

具体化・複雑・ユニーク

GaiaX
Empowering the people to connect

要素と基本スタイルの初期化 (1)

■ 要素と基本スタイルの初期化 (1)

➡ 3ping.orgさんのdefault.cssみたいな奴で要素のスタイルをならしておく。

➡ @import hackやhigh pass filterなどを使ってレガシーブラウザではtext/plainライクなスタイルでw

<http://3ping.org/2006/04/09/0747>

<http://css-discuss.incutio.com/?page=ImportHack>

<http://www.tantek.com/CSS/Examples/highpass.html>

要素と基本スタイルの初期化 (2)

■ 要素と基本スタイルの初期化 (2)

- ➡ 汎用的に使い、特定のパターンに属さないスタイルなども要素の初期化と同じ段階で行う。
- ➡ アンカーとかフォームの要素とか
- ➡ ボディ自体とかサイト全体のフォント設定とか

パターン別のスタイル

■ パターン別のスタイル

- ➡ 当然パターン別のスタイルを単独で作るには**パターンのみ**のHTMLが必要になる！
- ➡ この時は初期化用のCSSとパターン用のCSSのみで作り、極力シンプルなセレクタを用いる。
- ➡ width等のレイアウト系のプロパティは相対的に設定、ないしは設定しない

モジュール別のスタイル

■ モジュール別のスタイル

➡ モジュール別のCSSでは依存するパターンのCSSをCSS冒頭で@importする。

➡ スタイルの上書きないしは抽入に必要な差分のスタイルのみシンプルに書く。

➡ HTMLも当然単独で必要。

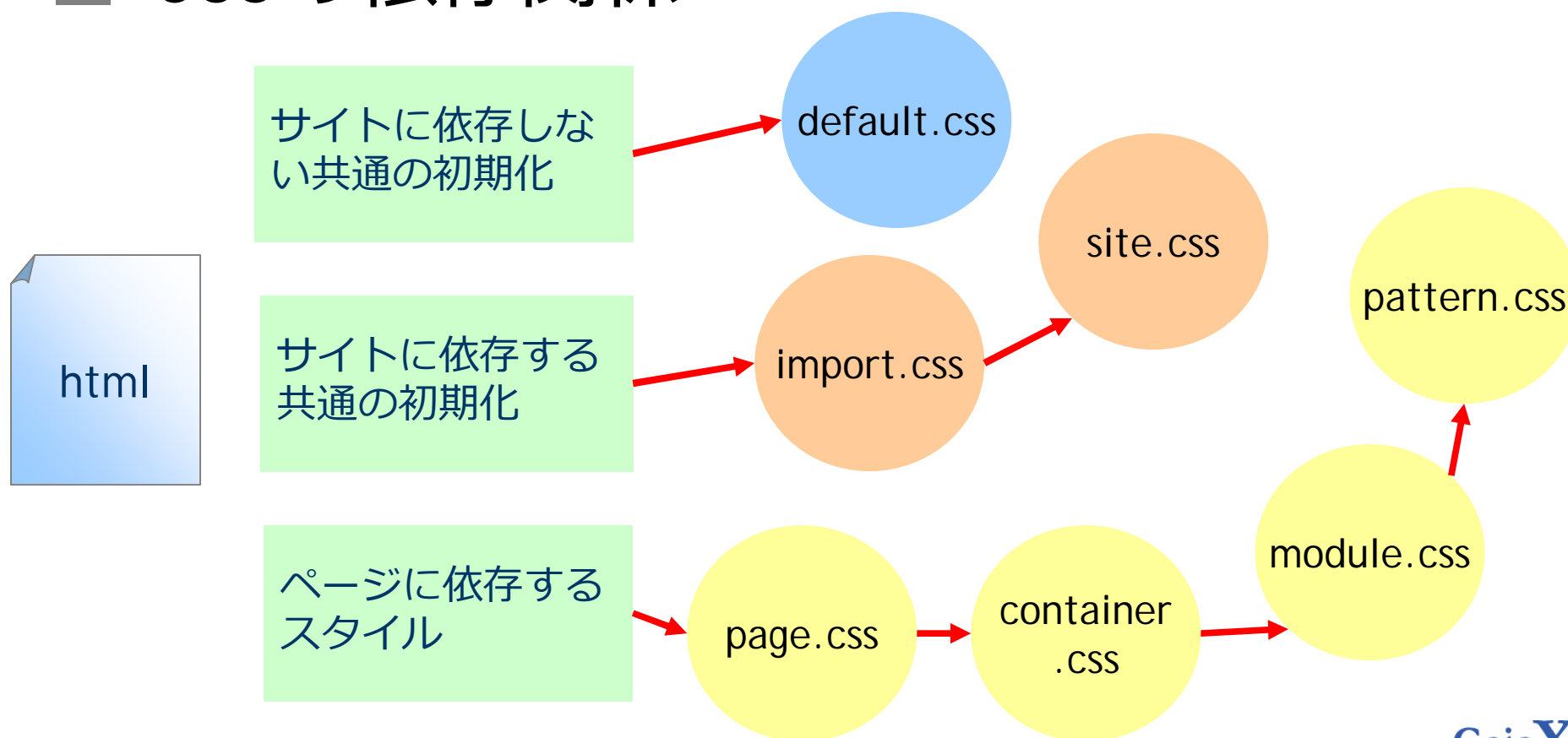
ページ別のスタイル

■ ページ別のスタイル

- ➡ ページ別のスタイルは@importだけで完結するのが理想系。
- ➡ htmlに記述するlinkによるCSSのロードは3つが基本
- ➡ default.css, import.css(共通のCSSのロード), page.css(ページ毎)

CSSの依存関係

■ CSSの依存関係



パターンとレイアウトの分離

■ パターンとレイ アウトの分離

カスケード処理 (1)

■ 出所(origin)について (1)

➡ 文書作成者(author) : 文字通りそのhtml
作った人

➡ ユーザー(user) : 文字通り見てる人

➡ ユーザーエージェント(user agent) : 平たく
言えばブラウザの事

あてがわれるCSSの出所には上記の3パターンが存在する。

カスケード処理 (2)

■ 出所(origin)について (2)

- ➡ user cssとはユーザーが任意で作成出来る、そのユーザーだけのCSS
- ➡ user agent cssとはブラウザのデフォルト表示
- ➡ default.cssなどはuser agentのデフォの差異を極力埋めるもの

カスケード処理 (3)

■ 出所(origin)について (3)

➡ author > user > UA の優先度がデフォルト

➡ !importantがあるプロパティ値は何よりも優先される

```
p {  
  text-indent: 1em !important;  
}
```

カスケード処理 (4)

■ カスケード処理の順番

- ➡ media type(screen, print, tvなど)でルール絞込み
- ➡ 出所による優先度
- ➡ セレクタの詳細度による優先度ソート
- ➡ 出現順番による優先度(上書き)

カスケード処理 (5)

■ セレクタの詳細度の計算 (1)

➡ idセレクタの数 : a

➡ その他の属性と擬似クラス(first-child, link, visited, hover, active, focus, lang)の数 : b

➡ 要素の数 : c

カスケード処理 (6)

■ セレクタの詳細度の計算 (2)

- ➡ 擬似要素(first-letter, first-line, before, after)は無視
- ➡ 全称セレクタ(* の事)はCSS2では詳細度0でCSS1では詳細度1
- ➡ font要素やalign属性のようなCSS以外で適用されたルールは詳細度0

カスケード処理 (7)

■ セレクタの詳細度の計算 (3)

➡ 詳細度をDとすると、次の式で出てきます。

$$D = a \times 10^2 + b \times 10 + c$$

➡ # >> . or [] >> element みたいに覚えるといいかも。

カスケード処理とPattern Driven CSS

■ 複雑・具体化、カスケードの詳細度

➡ 構造的により具体化して行く作業とカスケードの詳細度の算出はかなり近い

➡ つまり適切なモジュール化、パターン化をしていけばセレクタの優先順位をそれほど気にしなくて良いはず

➡ カスケードの順位がゴチャゴチャになりにくい。

カスケードを利用した分離 (1)

■ パターンに当ててるCSS

```
dl . directory-navi gati on {  
  font-si ze: 100%;  
}
```

```
dl . directory-navi gati on > dt {  
  font-si ze: 120%;  
  font-wei ght: bol d;  
}
```

```
dl . directory-navi gati on > dd > ul > li {  
  di spl ay: i nl i ne;  
}
```

カスケードを利用した分離 (2)

■ モジュール化する際のスタイル

```
ul .director es-modul e > li {  
  float: left;  
  width: 50%;  
}
```

```
ul .director es-modul e > li > dl .directory-  
navi gati on {  
  border-bottom: solid 1px #FF0000;  
}
```

その他

 その他

CSS SAC と CSSOM

■ CSSのparser

➡ CSS SACはXMLで言うSAX

➡ CSSOMはXMLで言うDOM

➡ CSS SACはCPANモジュールとして存在する。
parseすると全称セレクタが律儀にアスターになるorz...

parseさえ出来れば、アレもナニも出来そう

csstidy

- 実はcsstidyって存在します

<http://csstidy.sourceforge.net/>

- ➡ でもなんかインデント潰れる
- ➡ XHTML1.1のモジュール別とか
- ➡ 詳細度別とかやって欲しい...

作っちゃうか？

Pattern Driven CSS toolkit (1)

- 名前はまだ無いけど
- ➡ helperでパターン生成
- ➡ パターンやらモジュールやらをJson(ML)とかYAMLとかで管理出来たらウマー
- ➡ ページはどのモジュールを組み合わせるか...だけ書けばOKとかね。

Pattern Driven CSS toolkit (2)

■ 利点とか

- ➡ ミニマムスタートで尚且つ構造化しやすい段階からマークアップが出来る
- ➡ ベースパターンのhtmlないしはCSSを書き換えれば全ページに適用されるような仕組み

出来れば2007年(度)中には出したいNAR
共同開発者ぼしゅーちう?

Pattern と API

- API化出来るデータ構造は
 - ➡ UI上でもパターン化出来るんじゃないだろうか
 - ➡ 実は親和性が高い？
 - ➡ XML系のWeb serviceプロトコルとかJSONPとかをHTML化とかとか

Webappの設計も綺麗になるんじゃないのかなあーと

まとめ

■ まとめ

- ➡ Pattern Drivenでもっとロジカルに尚且つもっと簡単にCSS出来るヨ
- ➡ Pattern Drivenなら保守性も上がるヨ
- ➡ CSSはまだまだ開発者の未開拓ゾーンだから、結構面白いヨ

Thanks

ご清聴
ありがとうございました。

Toru Yamaguchi <zigorou@cpan.org>

質問ある？