

レスポンシブ・ウェブ・デザイン 講習資料

V2013/8/12 IMJ こやなぎ ともや

自己紹介

こやなぎ ともや



と、申します。

株式会社アイ・エム・ジェイ (<http://www.imjp.co.jp/>)
で、フロントエンドエンジニアをやってます。

GitHub: **@tomk79** (<https://github.com/tomk79>)

Twitter: **@tomk79** (<https://twitter.com/tomk79>)

ブログとか: **www.ptx.jp**

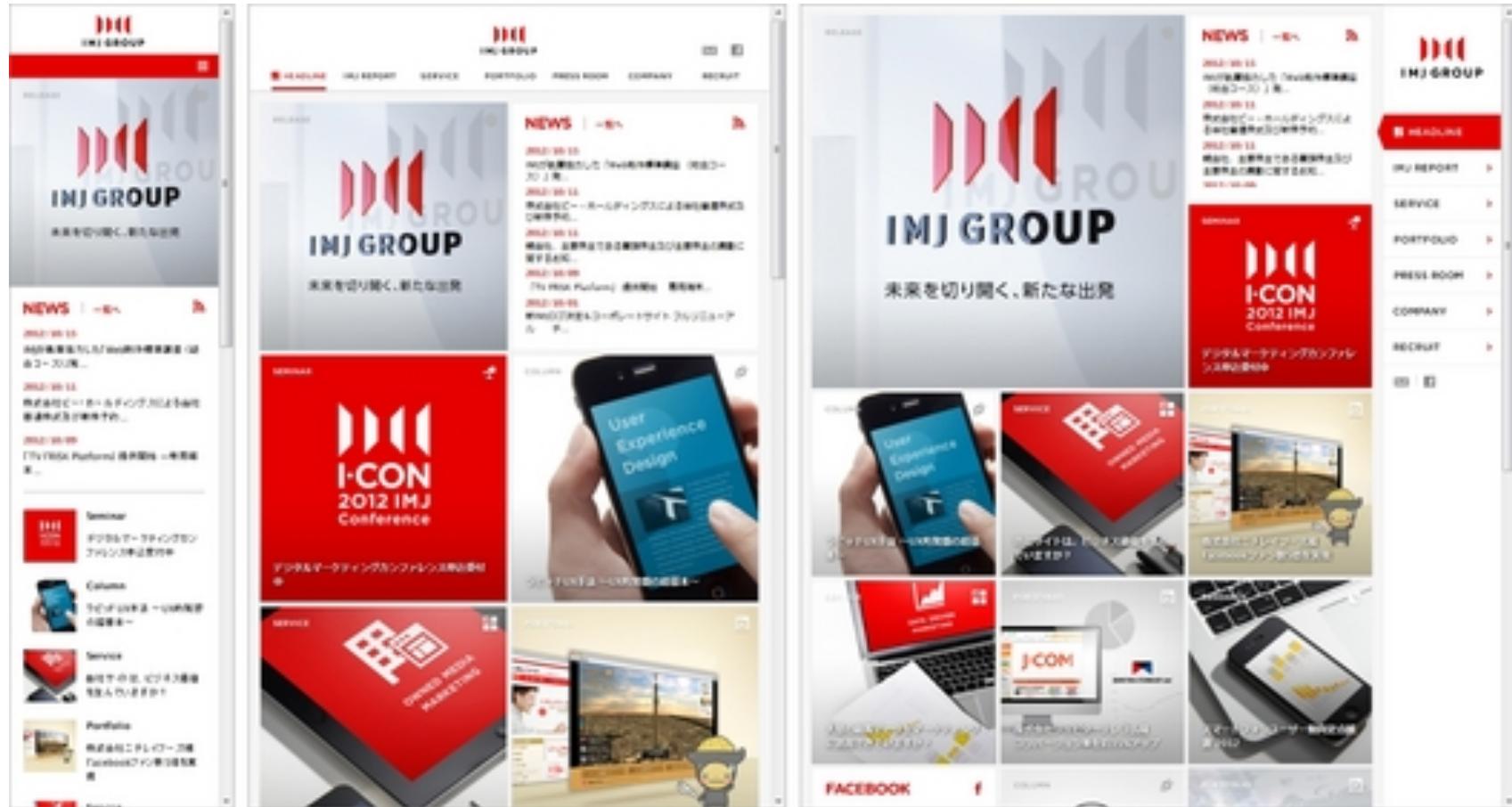
RWD

(レスポンシブ・ウェブ・デザイン)

レスポンシブ・ウェブ・デザインってなに？

- デバイスやブラウザの閲覧条件に合わせて、UI(ユーザーインターフェイス)を最適化するウェブデザインの手法。
- CSSの技術を使って、レイアウトをトランスフォーム(変形)させる。

画面の大きさに合わせて、 レイアウトをトランスフォーム(変形)



レスポンシブ・ウェブ・デザインは、
なぜ必要とされるようになったか？



スマートデバイスの登場

ウェブの閲覧環境の多様化・複雑化

ユーザーは、どんな端末で
アクセスしてくるかわからない！

マルチデバイス対応が重要に



RWD

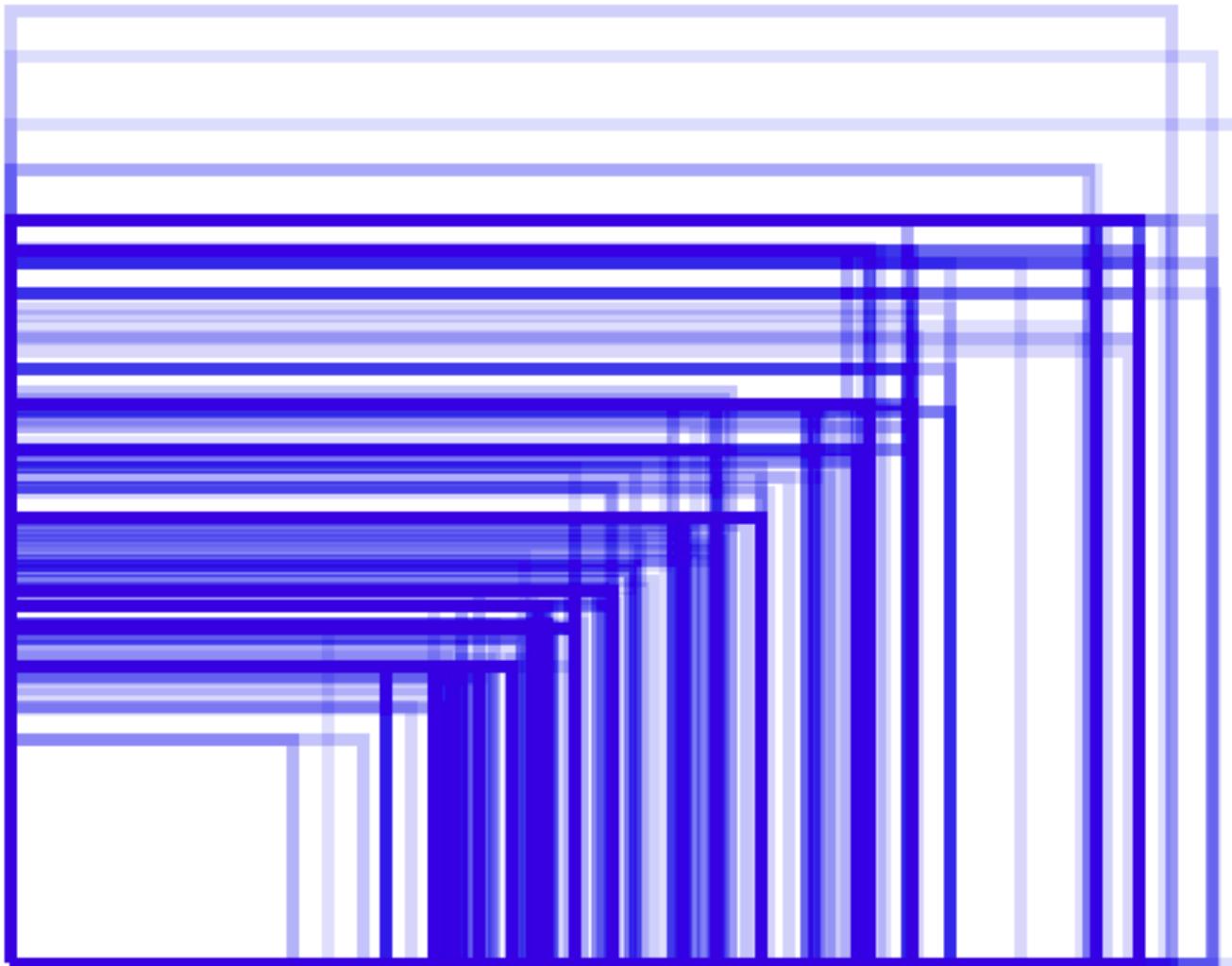
(レスポンシブ・ウェブ・デザイン)

マルチデバイス対応の 重要なポイント

3つ

マルチデバイス対応の重要なポイント その1
**デバイスそれぞれに
適したUIを提供すること**

Android端末の画面サイズ



その他にも！ 多種多様なネットデバイス



ゲーム端末



テレビデバイス



メガネ(?)



カーナビ(?)



スマート家電(?)

画面の大きさに合わせて、 レイアウトをトランスフォーム(変形)



押しやすいボタンサイズ

タッチデバイスの押しやすいボタンのサイズは、32px以上。

実績内容



▶ ポッカサッポロ フード & ビバレッジ 株式会社様

オウンドメディア時代の新評価軸「NPS」で、収益に貢献するWebサイトの実現へ

Web戦略立案

サイト統合リニューアル

実績内容



▶ ポッカサッポロ フード & ビバレッジ 株式会社様

オウンドメディア時代の新評価軸「NPS」で、収益に貢献するWebサイトの実現へ

Web戦略立案

サイト統合リニューアル

細かい操作は、
マウスの方がやりやすい。

タッチ画面のデバイスでは、
ボタンは大きめに。

文字情報の可読性

1行の文字数は17~21文字程度、16px相当の文字サイズ。



映画・音楽・本でつながる
新感覚マッチングサービス
「aune!(アウネ)」アプリ
提供開始のお知らせ

2013/07/22

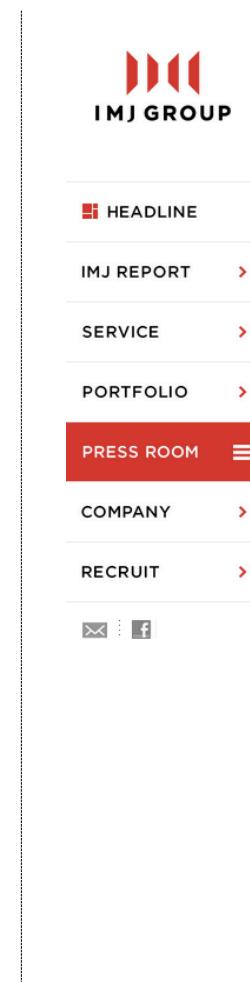
[いいね!](#) 8 [ツイート](#) 1

株式会社T-MEDIAホールディングス
株式会社アイ・エム・ジェイ
株式会社モテ

映画・音楽・本でつながる新感覚マッチング
サービス「aune!(アウネ)」アプリ提供開始
のお知らせ

～時代は「いいね！」から「あうね！」へ～

CCCグループでネット宅配レンタル、映像・音楽
配信、eコマースなどのネットエンタテインメント
事業を行う株式会社T-MEDIAホールディングス（
以下、TMH）とそのグループ会社でWebマーケテ
イング事業の最大手である株式会社アイ・エム・
ジェイ（以下、IMJ）との合併会社である株式会社
モテ（本社：東京都目黒区 代表取締役社長：久



映画・音楽・本でつながる新感覚マッチングサービス「aune!(アウネ)」アプリ提供開始のお知らせ

2013/07/22

[いいね!](#) 8 [ツイート](#) 1

株式会社T-MEDIAホールディングス
株式会社アイ・エム・ジェイ
株式会社モテ

映画・音楽・本でつながる新感覚マッチングサービス「aune!(アウネ)」アプリ提供開始のお知らせ
～時代は「いいね！」から「あうね！」へ～

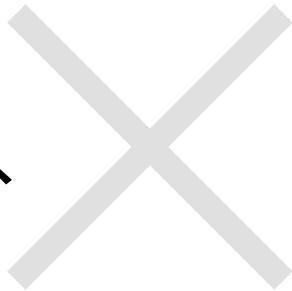
CCCグループでネット宅配レンタル、映像・音楽配信、eコマースなどのネットエンタテインメント事業を行う株式会社T-MEDIAホールディングス（以下、TMH）とそのグループ会社でWebマーケティング事業の最大手である株式会社アイ・エム・ジェイ（以下、IMJ）との合併会社である株式会社モテ（本社：東京都目黒区 代表取締役社長：久田祐通）は、映画・音楽・本の“インタレストグラフ（※1）”を活用したマッチングサービス「aune!（アウネ）」アプリ（iPhone／Android）を本日より正式に提供開始いたしました。

誕生の背景-「約7割はソーシャル疲れ」

SNS利用者は、今や国内ネットユーザーの52%にあたる4,965万人（2013年5月ICT総研）にも上り、その中でも、約7割の方はSNSのつながりに疲れている、ソーシャル疲れの症状が見られるという調査結果（※2）が出ています。本来、ネット上のコミュニケーションはもっと楽しいもの。「気軽に本当に気の合う人とつながりたい。」「ドキドキ・ワクワクしたい。」そんなユーザーのニーズに答える、全く新しいサービスとして「aune!（アウネ）」は誕生しました。

マルチデバイス対応の重要なポイント その2
URLが1つであること

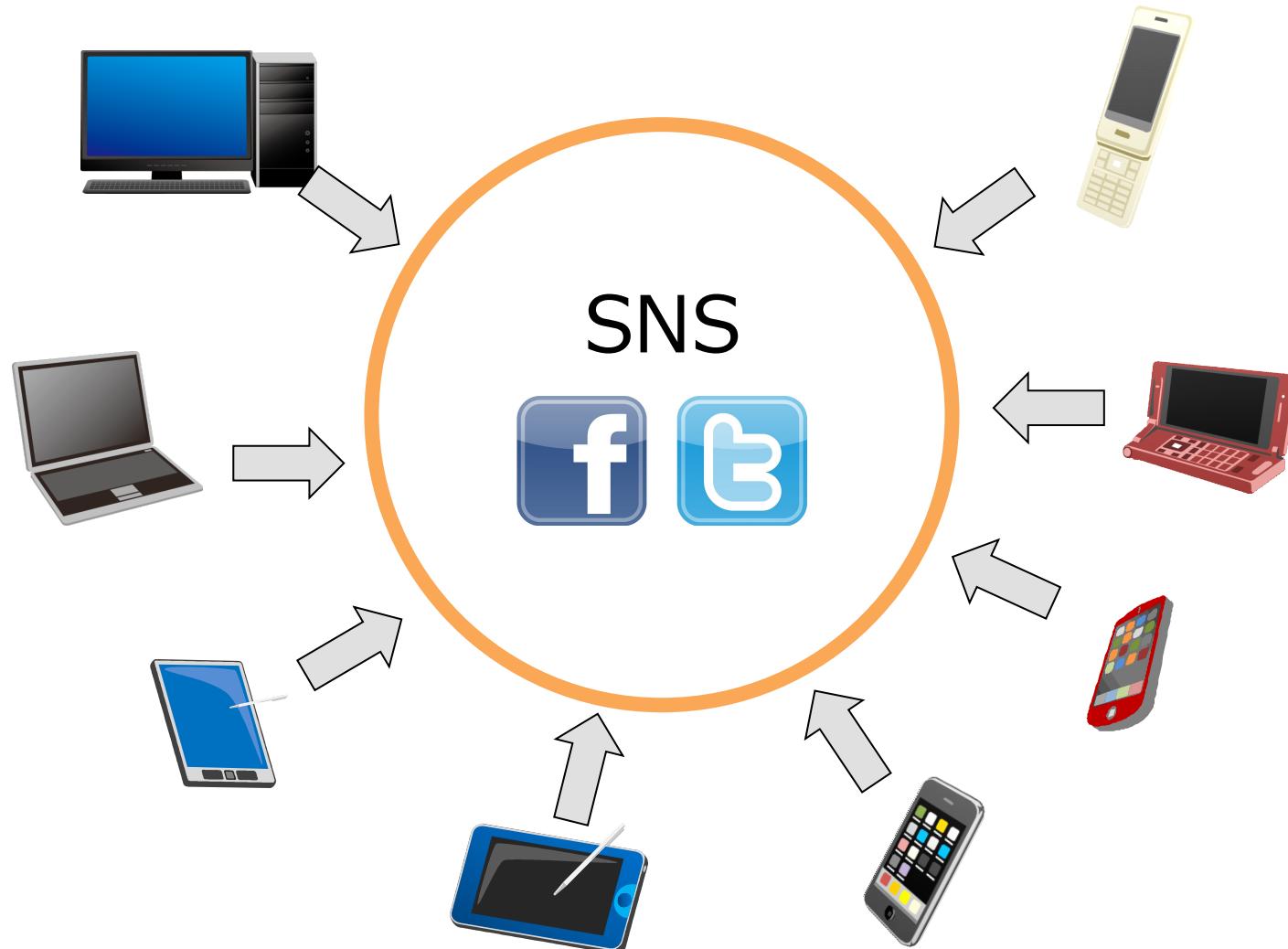
スマートデバイス



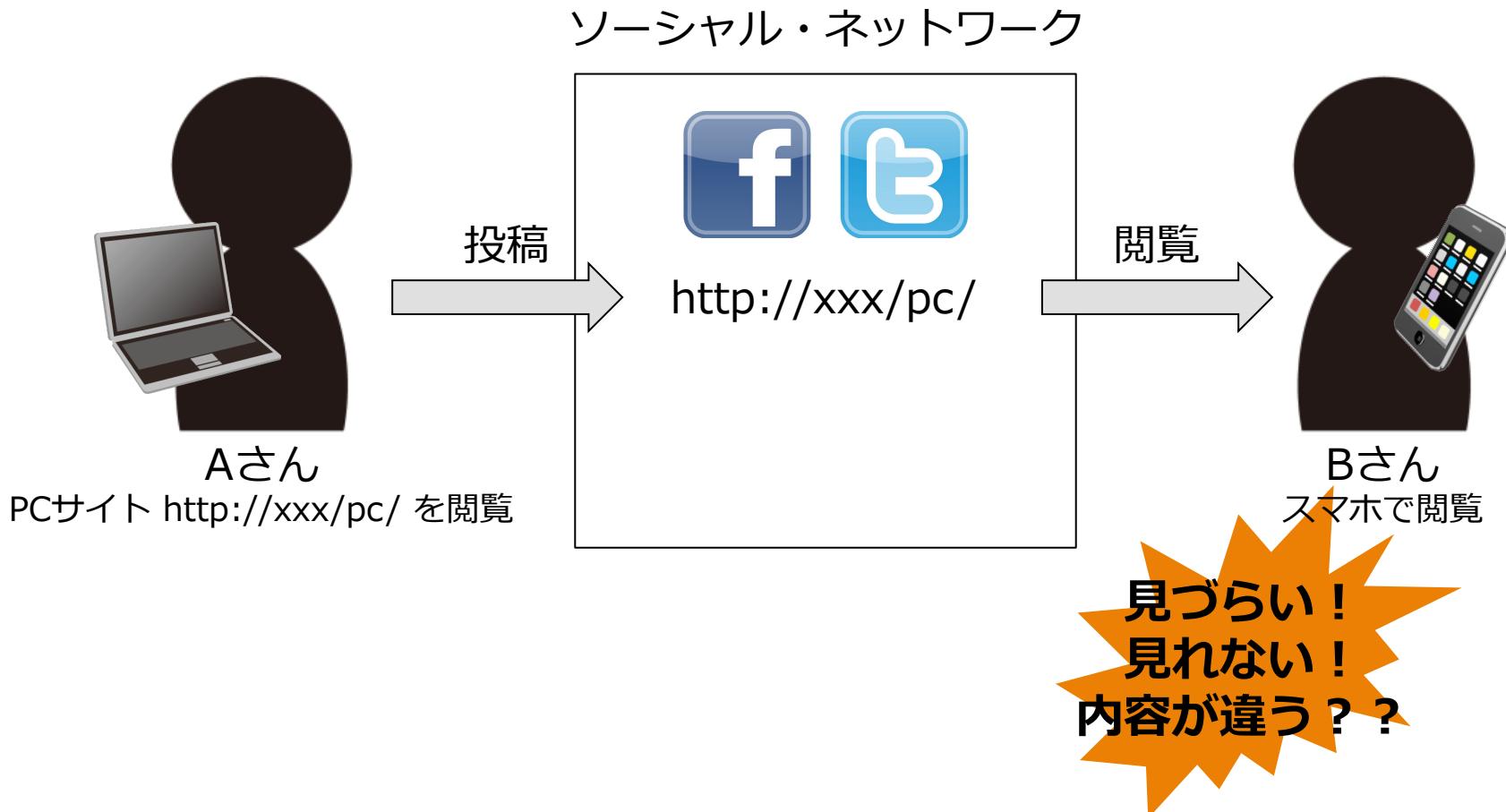
SNS

SNSを介したコミュニケーション

ユーザーはそれぞれのシチュエーションに合わせて、
多用な環境から利用している。



仮に、“PC専用サイト”と “スマホ専用サイト”に分かれていたら



マルチデバイス対応の重要なポイント その3
**デバイスを問わず、
同じ体験ができること**

“同じ体験ができる”とは

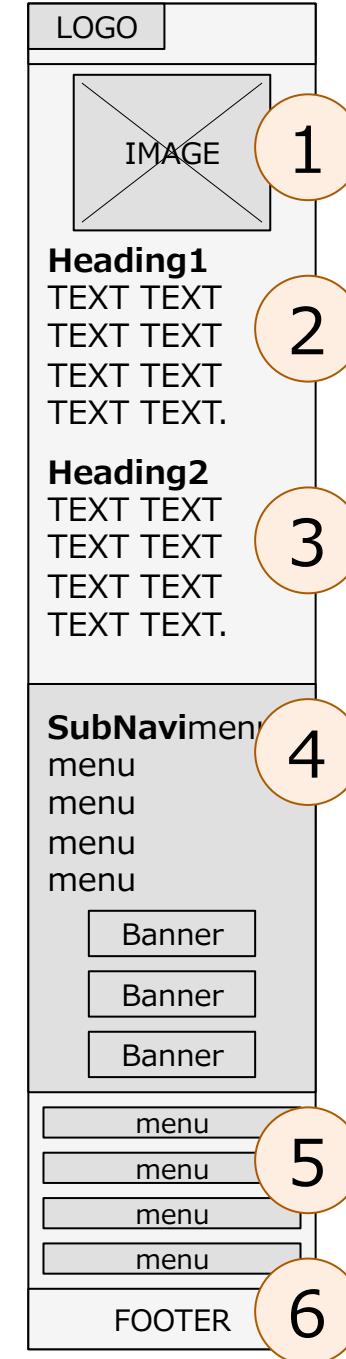
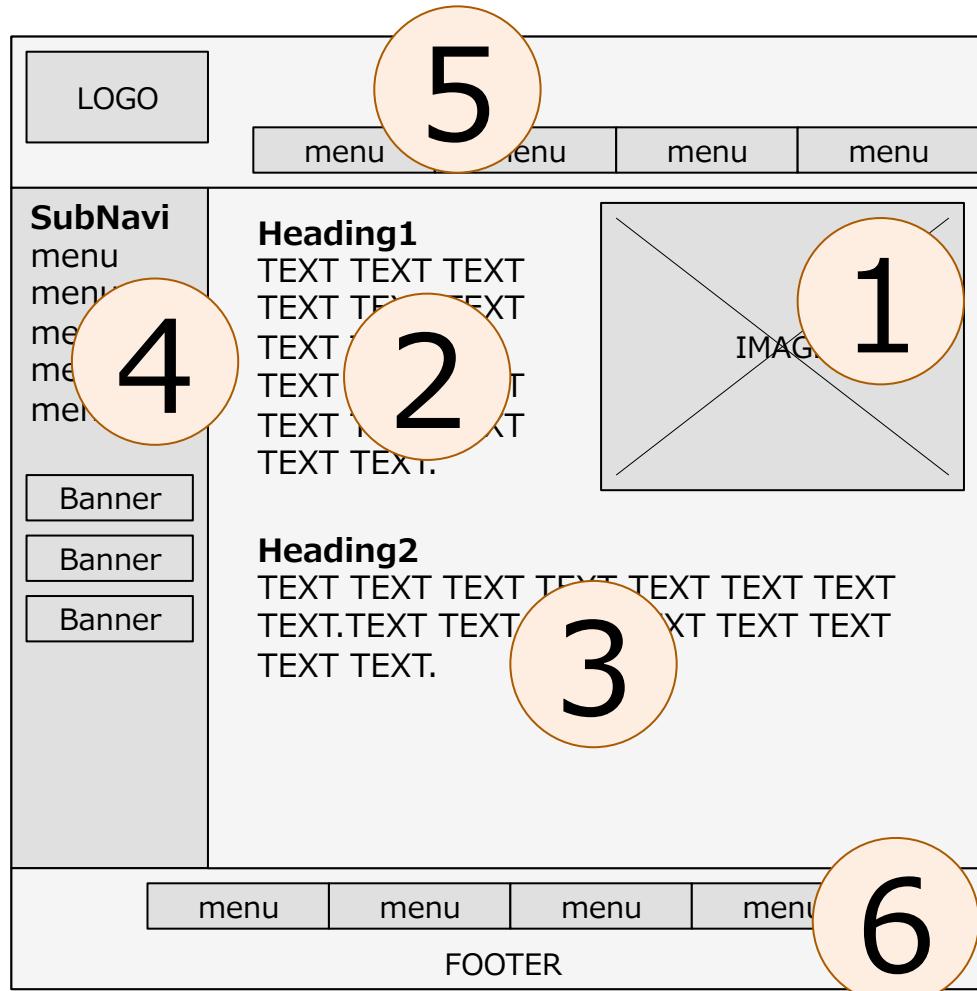
同じ情報を取得できる

同じ機能を利用できる

同じ情報を取得できる

コンテンツのプライオリティを合わせせる

ヒトは、2つの要素に同時に注目することはできません。
視線が流れる順番を合わせれば、画面が小さくても同じ体験を提供できる
ハズ。



画面は小さいけれど・・・、
「機能をけずらない」
「情報を削らない」
ように、なるべく心がけましょう。

よく誤解されがちなのですが、
RWDでできることは、
「スマートフォンサイト」「タブレットサイト」
ではありません。

デバイスやブラウザの特徴に合わせて、
UIのスタイルを最適化する手法です。

マルチデバイス対応の 重要なポイント 3つ

1. デバイスそれぞれに適したUIを提供すること
2. URLが 1 つであること
3. デバイスを問わず、同じ体験ができること



RWD

(レスポンシブ・ウェブ・デザイン)

～ RWDを実装してみよう～

こんなHTMLを用意しました。

RWDの練習

ページのタイトル

ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト。

ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト。

ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト。

ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト。

ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト。

ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト、ページのテキスト。

(C)わたくし

menu

- [ページ1](#)
- [ページ2](#)
- [ページ3](#)
- [ページ4](#)
- [ページ5](#)

このページを RWD にしてみましょう。

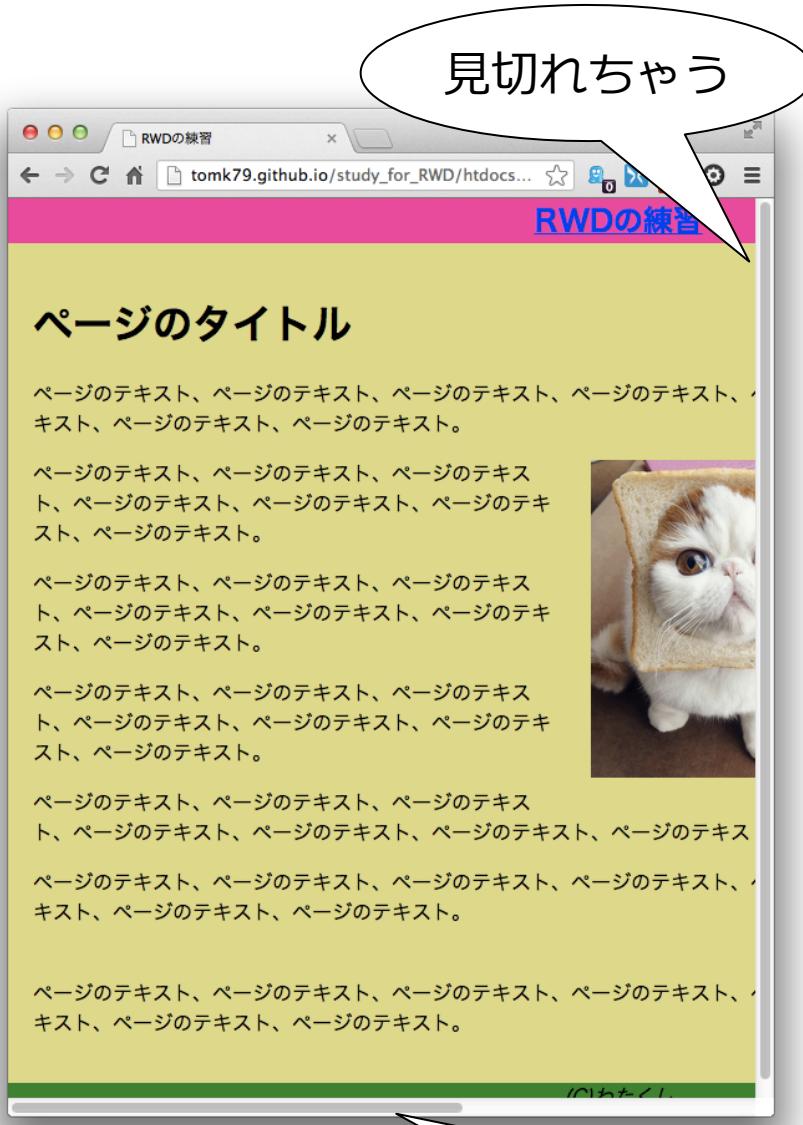
このワークショップの内容の 対象ブラウザ

OS	ブラウザ	バージョン
Windows	Internet Explorer	9以降
	Google Chrome	最新版
	Firefox	最新版
Mac OSX	Safari	5以降
	Google Chrome	最新版
	Firefox	最新版
iOS	Safari	iOS 5以上
Android	標準ブラウザ	Android 2.3以降

- 古いIE(8以下)に対応しようとすると、いろいろ面倒くさい実装をいっぱいやらないといけません。
- このワークショップでは、わかりやすさを優先し、IE9以上ということにします。
- が、実際のお仕事では、IE7以上対応というのが、現在のところ主流になっています。

サンプルファイルの構成

- **/index.html**
 - HTML本体
- **/img/nyanko.jpg**
 - コンテンツに表示されるにゃんこの画像
- **/common/css/**style.css****
 - CSS環境の基礎構築
 - 基本レイアウトの実装



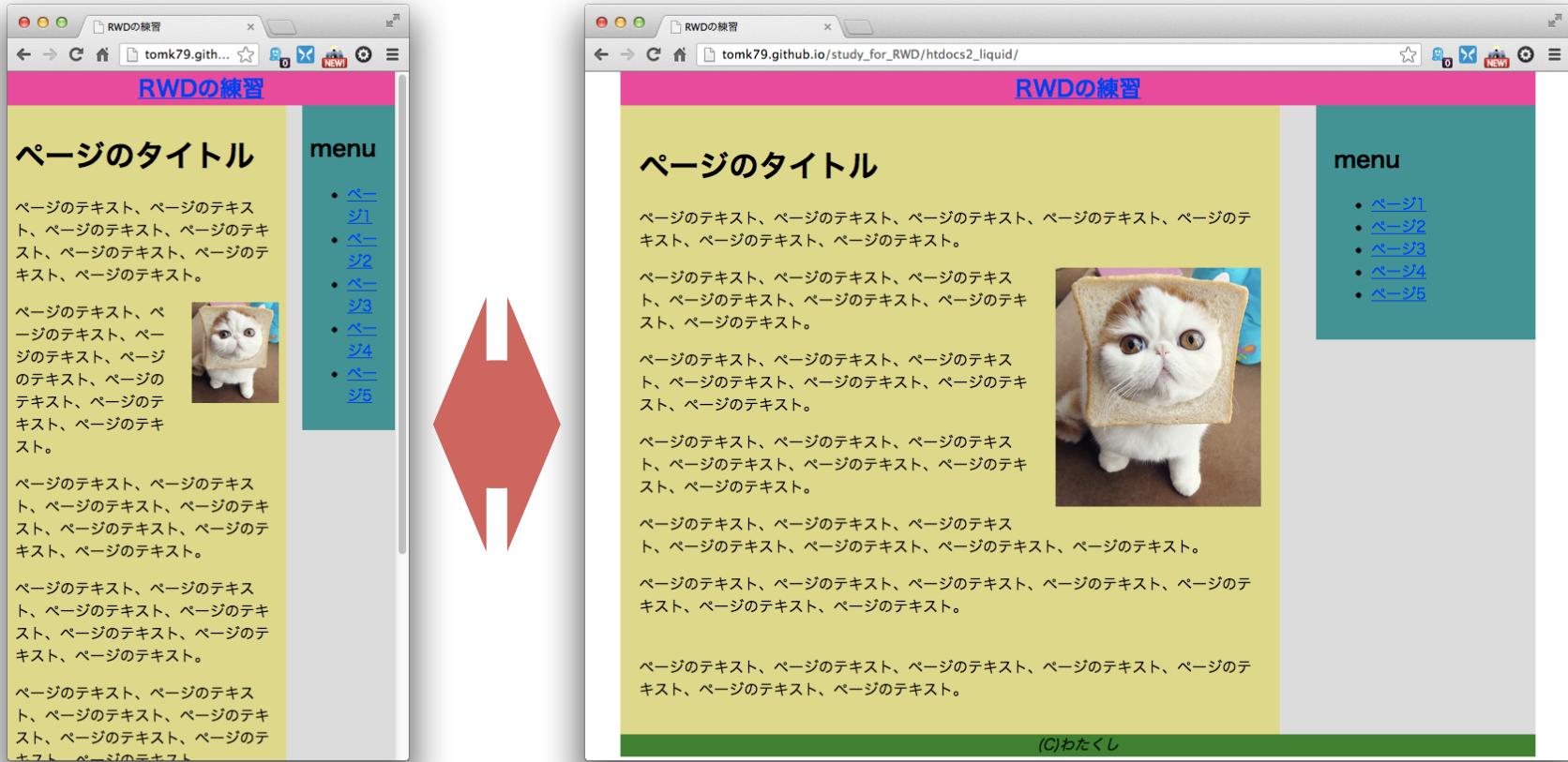
RWDに入る前に・・・ **RWDの基本は リキッドレイアウト**

ブラウザのウィンドウを小さくすると、画面の右側が見きれてしまい、横スクロールバーが表示されてしまいます。

これを、**画面の大きさに合わせて伸び縮みするように(=リキッドレイアウト)**修正してください。

リキッドレイアウトにしてみよう！

～画面の大きさに合わせて伸縮するようにする～



common/css/style.css

before

```
.outline{  
    text-align:left;  
    width:980px;  
    margin:0px auto;  
    background-color:#dddddd;  
}
```

after

```
.outline{  
    text-align:left;  
    width:auto;  
    max-width:980px;  
    margin:0px auto;  
    background-color:#dddddd;  
}
```



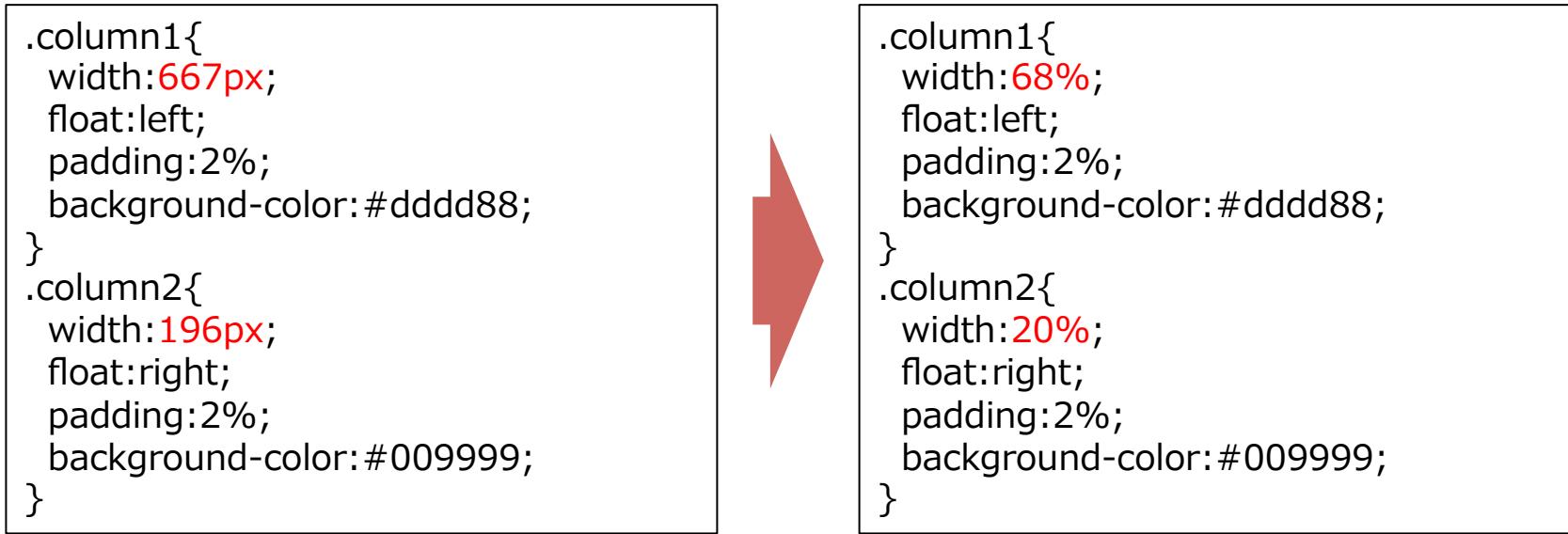
common/css/style.css

before

```
.column1{  
width:667px;  
float:left;  
padding:2%;  
background-color:#dddd88;  
}  
.column2{  
width:196px;  
float:right;  
padding:2%;  
background-color:#009999;  
}
```

after

```
.column1{  
width:68%;  
float:left;  
padding:2%;  
background-color:#dddd88;  
}  
.column2{  
width:20%;  
float:right;  
padding:2%;  
background-color:#009999;  
}
```



common/css/style.css

before

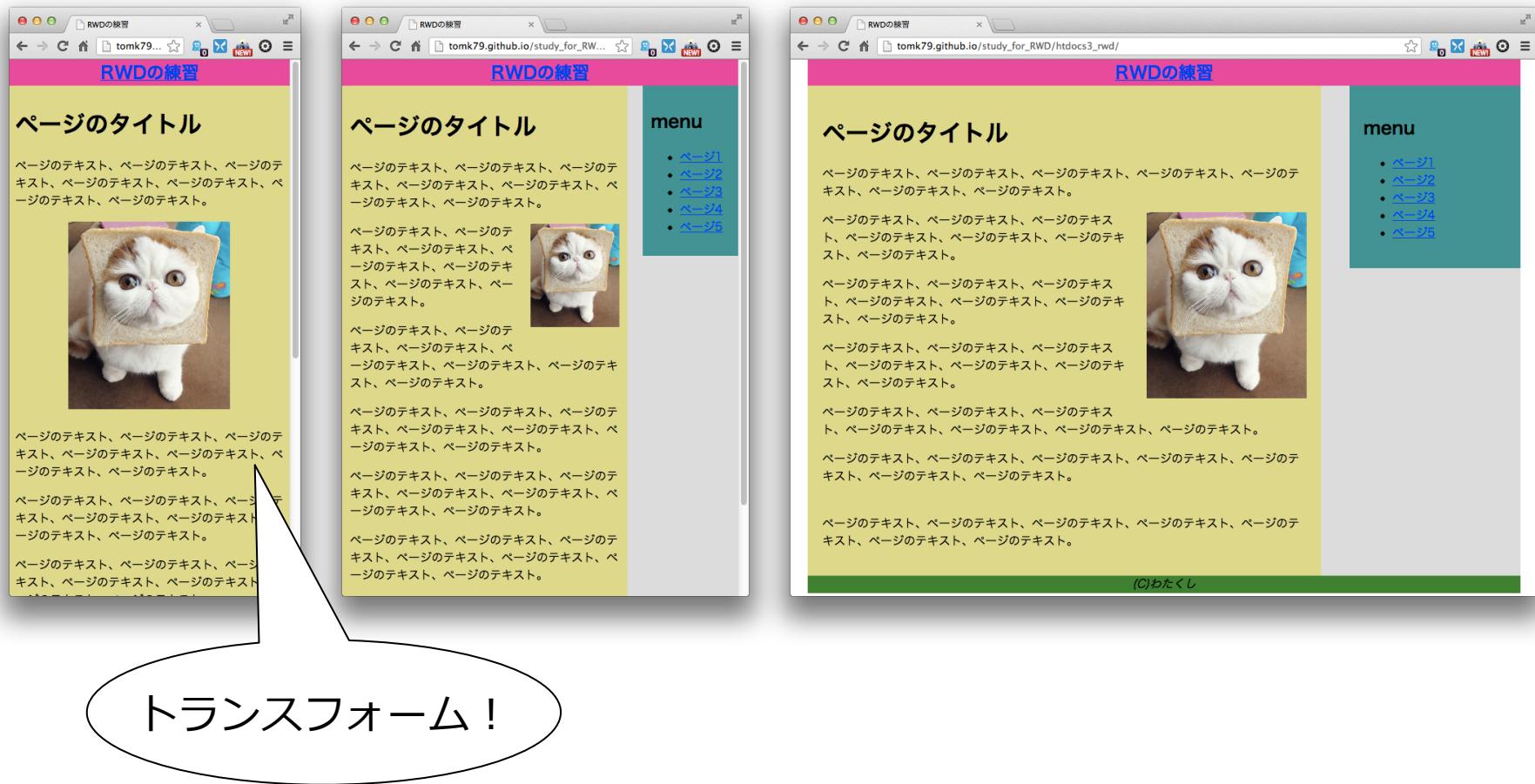
```
.nyanko{  
  float:right;  
  width:219px;  
  margin-left:20px;  
  margin-bottom:20px;  
}
```

after

```
.nyanko{  
  float:right;  
  width:33%;  
  margin-left:20px;  
  margin-bottom:20px;  
}
```



画面が一定の幅より狭くなるとき、 レイアウトをトランスフォームさせる (RWD)



修正するポイント

イメージは大きめに
中央に表示。



サイドバーは
コンテンツ領域の
下に落とす。



RWDはどう実現するのか？

RWDはこの技術でできている

- Media Types
- Media Queries (CSS3)

CSS の “Media Types” とは？

『ウェブページの出力媒体』のこと。

ユーザーが閲覧に使用している、
デバイスの種類、とも言える。

“Media Types” の種類

Media Type	説明
screen	非ページ型のコンピュータ・スクリーンを示す。
tty	固定幅の文字グリッドを用いたメディア、例えば、テレタイプ、端末、表示能力に制限のある携帯デバイスなどを示す。このメディアでは、画素単位 px は使えない。
tv	テレビ型デバイス（解像度、色数が低く、スクロール能力に制限がある）を示す。
projection	プロジェクタを示す。ページ媒体固有の整形モデルが存在する。
handheld	携帯デバイス（小画面、モノクロ、ビットマップ画像、大域幅に制限がある）を示す。
print	ページ形式の不透明な物質及び、印刷プレビュー・モードで見る文書を示す。ページ媒体固有の整形モデルが存在する。
braille	点字の触覚をフィードバックするデバイスを示す。
embossed	点字のページを出力するプリンタを示す。
speech	音声合成機器を示す。
all	全てのデバイスに適当。

“screen”



“handheld”



“screen”



“screen”



区別できない！

CSS の “Media Queries” とは？

『Media Types を大幅に拡張する仕様』

デバイスの種類だけではなく、
「画面の大きさ」、「アスペクト比」、
「表現可能な色の数」など、
詳細な特徴によってスタイルを変更できる。

“Media Queries” でできること

Media Query	説明
width, min-width, max-width	view port の横幅、つまりウィンドウの横幅 (css の長さの単位を使える)
height, min-height, max-height	view port の高さ、つまりウィンドウの高さ (css の長さの単位を使える)
device-width, min-device-width, max-device-width	デバイスの横幅、つまりディスプレイの横幅 (css の長さの単位を使える)
device-height, min-device-height, max-device-height	デバイスの高さ、つまりディスプレイの高さ (css の長さの単位を使える)
device-aspect-ratio, min-device-aspect-ratio, max-device-aspect-ratio	デバイスのアスペクト比 (16/9 や 4/3 のように整数とスラッシュと整数を指定する)
color, min-color, max-color	256bit カラーなら 8 、 8bit カラーなら 3 というように bit を二進数で表したときの桁。カラーデバイスじゃないなら 0
color-index, min-color-index, max-color-index	デバイスのカラールックアップ表のエントリーの数 (??)
monochrome, min-monochrome, max-monochrome	256bit モノクロなら 8 、 8bit モノクロなら 3 というように bit を二進数で表したときの桁。モノクロデバイスじゃないなら 0
resolution, min-resolution, max-resolution	解像度 (dpi、 dpcm という単位を使う)
scan	プログレッシブか、インターレースか (テレビだけ)
grid	グリッドの幅を指定する

※min- max- は以上、以下という意味です。たとえば、(min-height: 300px) だと「ウィンドウの高さが 300px 以上」という意味。

“Media Queries” の書式

all and (max-width:530px)

(MediaType部分) (MediaQuery部分)

“Media Queries” の4つの書き方

1. linkタグに media 属性を指定する。

```
<link rel="stylesheet" type="text/css" href=" . . . ." media="all and (max-width: 530px)" />
```

2. styleタグに media 属性を指定する。

```
<style type="text/css" media="all and (max-width:530px)">  
. . . .  
</style>
```

3. CSS内に @media句 を指定する。

```
@media all and (max-width:530px){  
. . . .  
}
```

今回は3の書き方で
やってみます！

4. CSS内の @import句 に指定する。

```
@import url(" . . . .") all and (max-width:530px);
```

common/css/style.css

append

```
.column1{  
width:68%;  
float:left;  
padding:2%;  
background-color:#ddd88;  
}  
  
.column2{  
width:20%;  
float:right;  
padding:2%;  
background-color:#009999;  
}
```

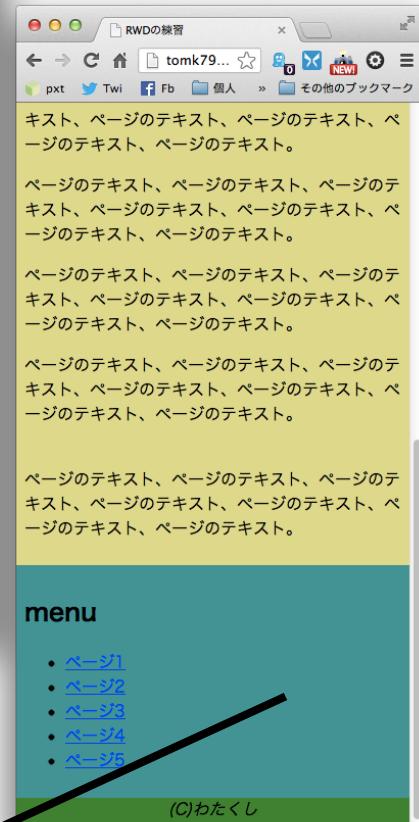
```
@media all and (max-width:530px){  
.column1{  
width:auto;  
float:none;  
}  
.column2{  
width:auto;  
float:none;  
}  
}
```

「ブレイクポイント」
と
い
ま
す。



width:auto;
float:none;

width:auto;
float:none;



common/css/style.css

append

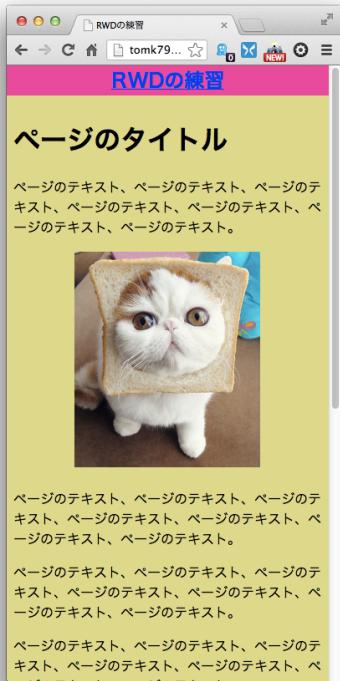
```
.nyanko{  
    float:right;  
    width:33%;  
    margin-left:20px;  
    margin-bottom:20px;  
}  
  
.nyanko img{  
    width:100%;  
    height:auto;  
}
```

```
@media all and (max-width:530px){  
    .nyanko{  
        float:none;  
        width:auto;  
        text-align:center;  
        margin-left:0;  
    }  
  
    .nyanko img{  
        width:60%;  
    }  
}
```

```
float:none;  
width:auto;  
text-align:center;  
margin-left:0;
```



RWD完成です！



viewport設定

スマートフォンのブラウザで、
ページの表示のされ方を設定する機能。

設定名	説明	値
width	デバイスの幅の論理値を設定。	実数値(px), または device-width デフォルトは 980px
height	デバイスの高さの論理値を設定。	実数値(px), または device-height
initial-scale	初期表示の拡大率。	1を100%とする数値(0~10)
minimum-scale	最小拡大率。	1を100%とする数値(0~10)
maximum-scale	最大拡大率。	1を100%とする数値(0~10)
user-scalable	ユーザーによる拡大・縮小操作を有効にするか。	yes, または no

書き方

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

ちなみに
Appleのマルチデバイス対応の考え方
(考察)

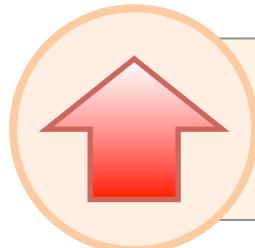




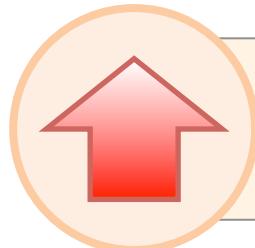


RWDの得手・不得手

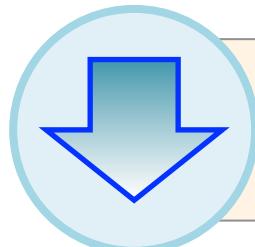
RWDを採用すると



可読性が向上



操作性が向上



一覧性は下がる

RWDにしなくてもよいケース



イメージで選択できるメニュー
テキストが読めなくても、絵でおおよその内容を伝えられる。

あまり詳しく読まない
ざっくりと全体像をつかむのが目的のページ。

例：
インデックスページなど

RWDにしなくてもよいケース



主にイメージで情報を伝えるページ
テキストが読めなくても、絵でおおよその内容を伝えられる。

テキストは少量、1行21文字におさまる
文章量が多すぎない。ズームインした状態で1画面に収まりきる程度。

例：
写真が多いプロダクト詳細ページ

RWDにした方がよいケース

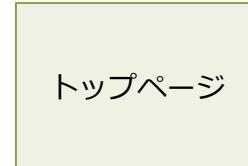


文字要素がメインのコンテンツ
絵や図で直感的に伝えるのが困難な、テキスト中心のコンテンツ。
特に長文になるほど、可読性が低いと読み難くなる。

例：
FAQページ
マニュアル
ニュースや読み物系の記事
文字の説明が多いサービス詳細ページ
ユーザー投稿コンテンツ

サイト全体としての構成例

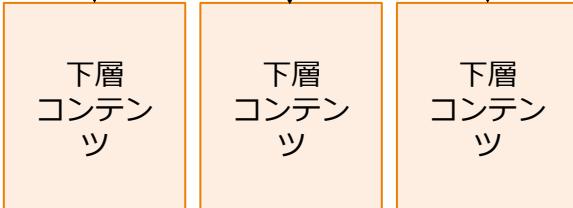
PCレイアウトのみで提供



カテゴリ
index

カテゴリ
index

カテゴリ
index



レスポンシブ・ウェブ・デザインで提供

～まとめ～

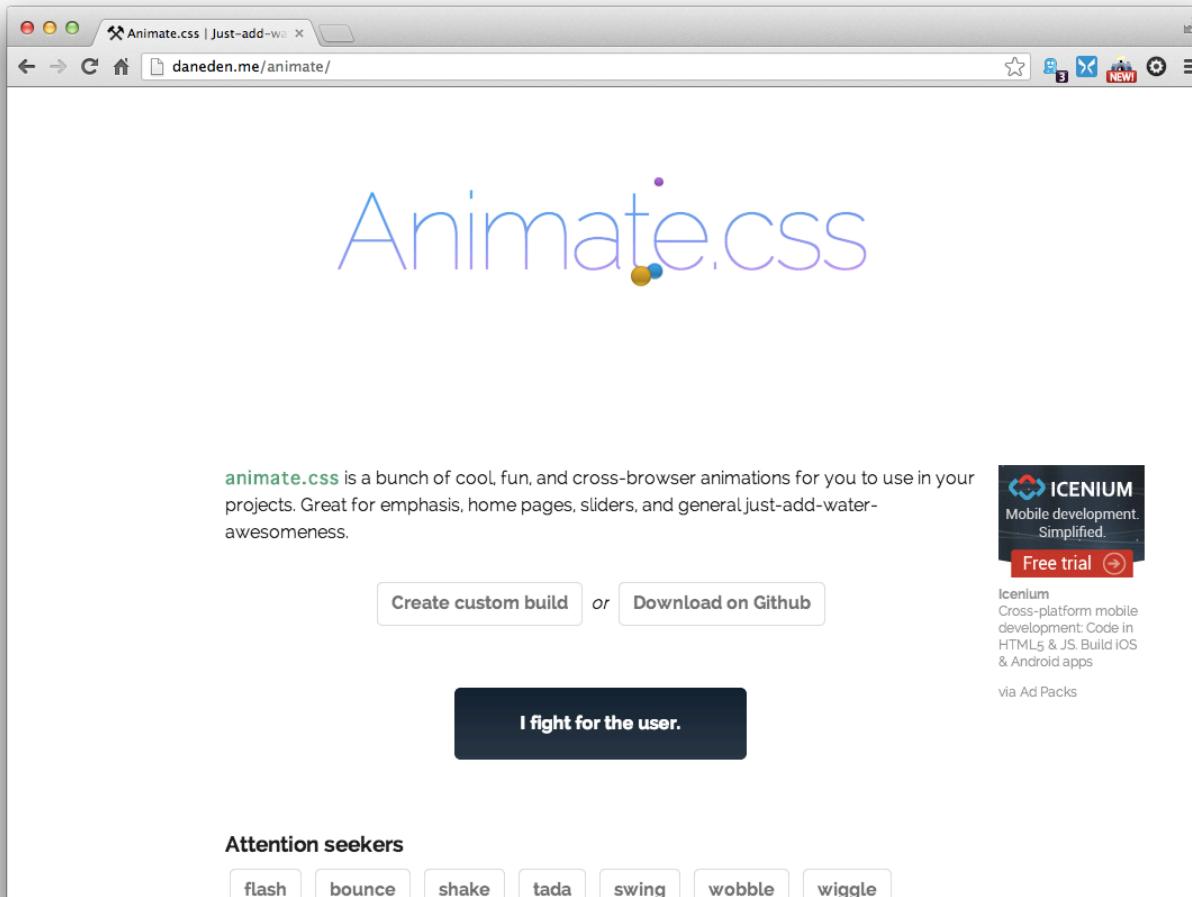
～まとめ～

- RWD とは、デバイスやブラウザの閲覧条件に合わせて、UIを最適化する手法。
- マルチデバイス対応の重要な 3つのポイント
 1. デバイスそれぞれに適したUIを提供すること
 2. URLが 1 つであること
 3. デバイスを問わず、同じ体験ができるこ
(「スマホサイト」ではない！)
- CSS の MediaType, MediaQuery という技術で実現される。
- RWDは、いついかなる場合にも正解とは限らない。
 - 「可読性」「操作性」が向上し、「一覧性」が下がる。

～おまけコーナー～

マルチデバイスウェブにアクセントを！
**アニメーション系のCSSを
使ってみましょう。**

Animate.css を、読む！



<http://daneden.me/animate/>

omake/index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>animate.css のデモ | RWDの練習 ~おまけコーナー~</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" type="text/css" href="./animate.css" />
  </head>
  <body class="">
    <h1>animate.css のデモ</h1>

    <div class="animated flash"></div>
  </body>
</html>
```

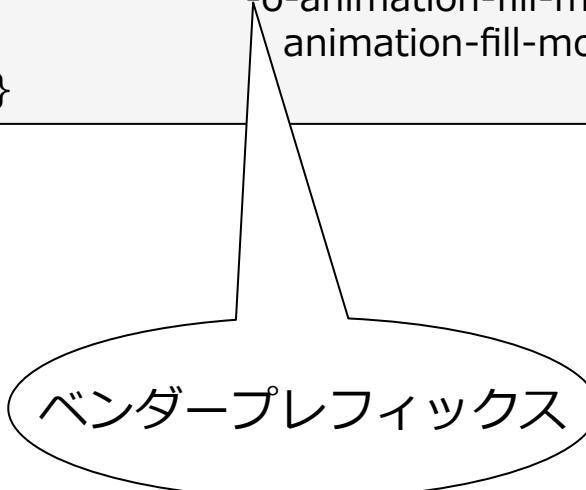
ライブラリ読み込み

クラス animated は必須

flash のところは
いろいろ選べる

omake/animate.css

```
.animated {  
    -webkit-animation-duration: 1s;  
    -moz-animation-duration: 1s;  
    -o-animation-duration: 1s;  
    animation-duration: 1s;  
    -webkit-animation-fill-mode: both;  
    -moz-animation-fill-mode: both;  
    -o-animation-fill-mode: both;  
    animation-fill-mode: both;  
}
```



ベンダープレフィックス

omake/animate.css

```
@-webkit-keyframes flash {
    0%, 50%, 100% {opacity: 1;}
    25%, 75% {opacity: 0;}
}

@-moz-keyframes flash {
    0%, 50%, 100% {opacity: 1;}
    25%, 75% {opacity: 0;}
}

@-o-keyframes flash {
    0%, 50%, 100% {opacity: 1;}
    25%, 75% {opacity: 0;}
}

@keyframes flash {
    0%, 50%, 100% {opacity: 1;}
    25%, 75% {opacity: 0;}
}

.animated.flash {
    -webkit-animation-name: flash;
    -moz-animation-name: flash;
    -o-animation-name: flash;
    animation-name: flash;
}
```

おしまい！