

マルチデバイス対応について

レスポンシブWEBデザイン (RWD)

「レスポンシブWEBデザイン」と検索すると

「レスポンシブWEBデザイン (Responsive WEB Design)」は、PC、タブレット、スマートフォンなど、デバイスの画面サイズによってResponsive=反応するWEBデザインという意味で、WEBサイトを単一のHTML（ワンソース）で実現する制作手法です。ブラウザのスクリーンサイズを基準にCSSでレイアウトすることで、デバイスごとに専用サイトを用意することなく、マルチスクリーンに対応したWEBサイトを制作できます。



っとした説明です。

レスポンシブWEBデザインは様々なデバイスに対応出来るので多くのサイトで採用している技術です。

【メリット】

- ・ 1つのHTMLファイルで複数デバイスに対応できる。
- ・ 更新・修正やサイト制作の作業工数の軽減出来る。
- ・ 各デバイスのURL統一化SEO対策に効果的になる。

【デメリット】

- ・ 設計やデザインが複雑になる。
- ・ PCとモバイルの内容が同じなので、ページの読み込みが重くなる可能性がある。
- ・ 常に新しいデバイスの登場に対応が必要になる。

まずは歴史を見てみましょう。

2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019

- 2007年、アップルが米国で「iPhone（アイフォーン）」を発売
 - 2008年、米国でAndroid OSを搭載したT-Mobile G1が発売
- 2011年6月 Cascading Style Sheets, level 2 revision 1 (CSS 2.1), 勧告
- 2011年6月 Cascading Style Sheets, level 3 (CSS3) 勧告
- 2012年6月 Media Queries, 勧告
- 2008年1月22日、HTML5ドラフト（草案）が発表
- 2014年10月28日、HTML5 が勧告
- 2016年11月1日、HTML 5.1 が勧告
- 2017年12月14日、HTML 5.2 が勧告
- 2015年 4月21日 Googleがモバイルフレンドリーの使用を開始
- 2017年「iPhone X」を発売

最新情報 2018年1月18日 2018年7月よりページの読み込み速度をモバイル検索のランキング要素として使用することを発表

W3Cの定める規格

作業草稿



最終草稿



勧告候補



勧告案



W3C勧告

最初の草案段階
ベンダープレフィックスで利用可能

作業草案に戻すべきかの判断前
無くなる場合があります

実装を呼びかける段階
ベンダープレフィックスは外すことを推奨

いよいよ準備ができましたという段階
ほぼ、もう変更ないかと

理論面でもOK！実用面でもOK！

そのポイントは大きく分けて
「4」つです。

Viewport

Media Queries

Box model

UI/UX

Viewport

「viewport」と検索すると

viewport（ビューポート）とは、日本語に訳すと「表示領域」という言葉がしっくりくると思います。例えば、デスクトップ環境（PCなど）のブラウザなどでは、ブラウザウインドウの表示領域がviewportになります。... 表示のされ方は、制作者がmeta要素のviewportで指定することができます。



っとした説明です。

このMetaタグを入れると、**デバイスの画面サイズ**に合わせた数値で表示されるようになります。

- ・ いまのところパターンは数えるほど
- ・ コピペして使う事が多いのか現状
- ・ 実機検証を最後に忘れずに

【記述例】

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

Media Queries

「Media Queries」と検索すると

メディア クエリは、CSS スタイルに適用できるシンプルなフィルタです。メディアクエリを使用すると、画面の種類、幅、高さ、向き、解像度など、コンテンツをレンダリングするデバイスの特性に基づいて、スタイルを容易に変更できます。



っとした説明です。

CSSをデバイスの画面サイズ毎に内容を変えられます。

- ・ 使うコツは整理整頓
- ・ 初期値の把握
- ・ 複雑にしない

【制作作業時のコツ】

バックアップ必須！（復活の呪文を唱えられるように）
行き当たりばったり絶対ダメよ～！

「max-width」の場合は大きいサイズから

【記述例】

```
body{
    background: #000000;
}

@media only screen and (max-width: 1280px){
    body{
        background: #333333;
    }
}

@media only screen and (max-width: 960px){
    body{
        background: #666666;
    }
}

@media only screen and (max-width: 640px){
    body {
        background: #999999;
    }
}
```

【ブレイクポイントの設定のポイント！】

- ・メディアクエリは同じものだけ使う
- ・ブレイクポイントは同じ数値で切り分ける
- ・記述箇所はセレクタの直下が好ましい
- ・プロパティと値は出来るだけ統一する
- ・「.css」のファイルは1ファイルが理想
- ・インデント・{}をしっかりと揃える
- ・そもそもHTML文書に無理が無いのかを確認

「min-width」は逆にスマートフォンサイズから作成するのであればコチラ

他にも書き方色々あるので、これに慣れたら必要に応じて書いてみるのも良いかも？！

CSSの点数計算記述ルールの確認

CSSの点数計算を解りやすく整理してみましょう。
HTML文書の、「id・class」は計画的に整えておくことで、誤った指定を回避できます。

CSSセレクタの優先順位の計算方法

指定方法	例	点数
タグのstyle属性	style=""	1000点
id	#hoge	100点
class	.hoge	10点
擬似クラス	a[href*="google"]	10点
要素名	ul	1点
擬似要素	:first-child	1点
全称セレクタ	*	0点

計算例

例	計算	合計点
style=""	1000(style属性)	1000点
#hoge	100(ID属性)	100点
li.color.label	1(要素名) + 10(クラス属性) + 10(クラス属性)	21点
table tr td.color	1(要素名) + 1(要素名) + 1(要素名) + 10(クラス属性)	13点
div + *[href\$="com"]	1(要素名) + 0(全称セレクタ) + 10(擬似クラス)	11点
h1 div + span	1(要素名) + 1(要素名) + 1(要素名) + 10(クラス属性)	3点
li:last-child	1(要素名) + 1(擬似要素)	2点

Box model

「マルチデバイス対応」のサイトの特徴は固定値からの解放です。



っとした説明です。

横幅の合計値をデバイスの画面幅（100%）を超えなければ、横スクロールのバーは出ません！

【ボックスモデルに関わるセレクト】

☒ width ☒ height ☒ padding ☒ margin ☒ border

【ボックスをレイアウトさせるセレクト】

☒ float ☒ clear ☒ position ☒ overflow ☒ display

合計値をデバイスの画面幅（100%）

ボックスモデル

margin-left

border-left

padding-left

width
(max-width)
(min-width)

padding-right

border-right

margin-right

width

【主な内容】

初期値はauto

数値と%の指定が可能

負の値は指定できない

HTMLのテキスト、画像（img）の適用される箇所

ポイント！

- ・デザインの都合が可能な限り「%」を使用しましょう。
- ・「max-width」「min-width」は便利

height

【主な内容】

初期値はauto

数値と%の指定が可能

負の値は指定できない

HTMLのテキスト、画像（img）の適用される箇所

ポイント！

- ・デザインの都合が可能な限り「初期値」にしましょう。
- ・「width」と違い、「%」は使い難い。

padding

【主な内容】

上下左右まとめたの指定が可能

数値と%の指定が可能

負の値は指定できない

CSSでbackgroundが適用される箇所

ポイント！

- ・「box-sizing: border-box;」を使えばだいたい解決するかも
- ・数値を入れれば絶対に余白を持ちます。（逆に効かない場合は別の何かが間違っています。）
- ・背景「background」に画像を持たせる時に有効
- ・余白の取り方に悩んだらpaddingで指定すると良いかも？！

margin

【主な内容】

上下左右まとめたの指定が可能

数値と%の指定が可能

負の値が指定できる

CSSでbackgroundが適用されない箇所

ポイント！

- ・「marginの相殺」がややこしい
- ・「ネガティブマージン」がややこしい
- ・「他のセレクトタとの関係性」がややこしい
- ・CSSが嫌われる原因はこれ…だが、理解すると自由自在のレイアウトが可能

border

【主な内容】

ボーダーのスタイル・太さ・色の指定が可能

上下左右まとめたの指定が可能

数値と%の指定が可能

負の値は指定できない

初期値で「backgroud」を含む箇所だが、「box-sizing」で調整可能

ポイント！

- ・「box-sizing: border-box;」を使えばだいたい解決するかも
- ・後で、横スクロールの原因探しで最後まで見つからない事が多いのがコレ

float

【主な内容】

初期値はnone

floatプロパティは、指定された要素を左「left」または右「right」に寄せて配置

ただし、左か右にしか配置出来なくなり中央に揃わなくなる

その他、特殊な仕様が多々ある

ポイント！

- ・使わない時は「none」にする。（これはよく使います。）
- ・コーディングで一番悩ましい存在、極力使わないというのがコツかも？！
- ・「float」したら「clear」を徹底する
- ・横並びになれば、横並びの合計値になるので、ボックスの数値をしっかりと把握してみましょう。
- ・ややこしいが「float」が自在に使えればどんなレイアウトも自由自在
- ・「ネガティブマージン」と合わせたら更に強力！

clear

【主な内容】

初期値はnone

「float」プロパティの回り込みを解除する。

「float」と「clear」はセットで考えると良いかも

ポイント！

- ・ 値に「left」「right」あるが一緒なので使うなら「both」
- ・ 使わない時は「none」にする。（これはよく使います。）

position

【主な内容】

positionプロパティは、ボックスの配置方法（基準位置）が、相対位置「relative」絶対位置「absolute」を指定出来る。表示位置の指定には「top」「bottom」「left」「right」を併用して、基準位置からの距離を設定する。

数値と「%」の値、負の値が使用可能 初期値は「static」

絶対位置への配置でスクロールしても位置が固定の「fixed」

ポイント！

- ・ヘッダーの固定や、ページトップへのボタンの固定に便利
- ・2つのセレクタを併用しなければならないのでややこしい
- ・負の値や、親子関係のがややこしい
- ・ややこしいが「position」が自在に使えればどんなレイアウトも自由自在
- ・「float」「ネガティブマージン」と合わせたら更に超強力！

overflow

【主な内容】

overflowプロパティはボックスの範囲内に内容が入りきらない場合に、はみ出た部分の表示の仕方を指定する。

初期値はvisible

autoはブラウザ依存があります。

ポイント！

- ・ まず「float」の親要素に「overflow: hidden;」が便利
- ・ タッチデバイスのスクロールは操作しにくい
- ・ 初期値は「overflow: visible;」と「height:auto;」で縦長のページは、スマートフォンならOK！？

display

【主な内容】

displayプロパティは、ブロックレベル・インライン等の、要素の表示形式を指定する際に使用

inlineは（横幅の絶対値を持ってない）

blockとinline-blockは（横幅の絶対値を持てる）

noneは要素が表示を隠す事が出来てしまう。

ポイント！

- ・ 要素の横並びの箇所（グローバルナビ等）には必須
- ・ 非表示「display:none;」の扱いには要注意
- ・ 表組「table」の解体に必要

インライン要素とインラインブロック要素とブロック要素の違い

	インライン 「display:inline;」	インラインブロック 「display:inline-block;」	ブロック 「display:block;」
主な用途	文章内	文章内、横並び	レイアウト、サイズ調整
主なタグ	a, strong, span	img	h1, p, ul, li, div
並び方	横	横	縦
初期サイズ	内容サイズ	内容サイズ	横は100%、縦は内容
widthとheight	-	指定可能	指定可能
padding	上下左右 <small>※余白はとってくれるが、上下の要素の位置には影響しない</small>	上下左右	上下左右
margin	左右のみ	上下左右	上下左右
line-height	外側にかかる	内側にかかる	内側にかかる
text-align	-	指定可能	指定可能
vertical-align	指定可能	指定可能	-

UI/UX

「UI/UX」はデザイナーの腕の見せ所です！

さまざまデバイスに対応したコンテンツを作るには、環境・時間・空間・性別・年齢・人数…等々それらを「User Interface：ユーザインターフェイス」や「User Experience：ユーザエクスペリエンス」という言葉で表現されています。ここで活きるのは「体験という経験」になるでしょう。



っとした説明です。

「使い易い」「役に立つ」「解りやすい」をポイントにデザイン設計を考えてみましょう。

- ・ タッチデバイスを意識したボタン（リンク）
- ・ 迷子にさせない導線
- ・ ページの表示スピード対策「画像」
- ・ ページの表示スピード対策「プログラム」
- ・ 使い易さの追求

タッチデバイスを意識したボタン（リンク）

■ タッチデバイスではボタンの領域を広く取りましょう。

～の要素タグはインライン要素になります。

その為、幅と高さを持たせる為に「display :block;」を上手く活用してみましょう。

■ ポインターが無いので、リンク箇所のデザインルールを変えてみましょう。

サイト全体でテキストリンクの箇所は「このデザイン」というルールを統一しリンクが付いていることを解りやすくしてみましょう。

■ リンクの付いたバナーデザインはより解りやすく

画像で作るバナーは、よりクリックしたら詳しい説明のページへ飛んでいくと解りやすくデザインしてみましょう。

迷子にさせない導線

■ 画面の面積が少ないデバイスでは、解りやすい導線を組みましょう。

サイト設計の段階で、無駄なリンク無駄なページは排除するのが理想です。

PCの特性、スマートフォンの特性を考えレイアウトを設計してみましょう。

■ 1クリックの動作で、1割のユーザーが離れると言われています。

無駄なページ移動はお客様が離れるだけです。スマートフォンやタブレットで縦に長いページは、そんなにストレスを感じさせないはずなので、1ページの内容（ストーリー）を充実させましょう。

■ ファーストビューで「最優先のコンテンツ」フッターで「各ページ」へ

ページを開いて最初に見えるのが「目的の内容」⇒深く読み進めて「次への誘導」

そこに興味のない方には「他の興味」へご案内出来るようにしてみましょう。

ページの表示スピード対策 「画像」

■ Retinaディスプレイや4Kテレビ等々への対応

Retinaディスプレイの場合解像度が2倍になるので、2倍のサイズの画像を用意し50%のサイズで指定をして画像は表示させましょう。ただし容量には要注意です！

■ imgとbackground-imageの違いを確認しましょう。

「img」要素はHTMLの文章構造で存在する意味が必要になります。

「background-image」は背景のCSSなので、文法上の意味はありません。上手く使い分けてみましょう。

■ Webアイコンや「canvas」「SVG」を上手に取り入れましょう

「jpg,gif,png」以外の画像表示やコンテンツが実装出来る段階に入ってきているので、上手に取り入れてみましょう。

ただし、実機検証やブラウザ依存には要注意です！

ページの表示スピード対策「プログラム」

■ 動的プログラムは表示スピードを低下させます。

JavaScriptはとても便利！…ですが、プログラムとしてはページの表示速度を低下させます。
ユーザビリティ、アクセシビリティという部分を考えて最低限の利用方法を考えてみましょう。

■ CMS等のテンプレートについて

簡単にWEBサイトを作成してくれるというツールがあったら理想的ですが、便利な分「何かしらの処理」が行われている場合、表示スピードに影響します。良く確認をしてから利用していきましょう。

■ 誤ったプログラムも表示スピードの低下に繋がります。

HTMLやCSSの記述が誤っていてもブラウザは独自解釈して表示してくれます。
これも表示スピードの低下に影響しますし、SEO（検索エンジン対策）にもマイナスの影響がでます。
検証サイト等を利用し、正しく記述できるよう頑張りましょう！

使い易さの追求

■ WEBフォントの利用

日本語に対応したWEBフォントも多く出てきました。無料で使える「源ノ角ゴシック」「源ノ明朝」が国内でも多く広まって来ています。上手く取り入れてみましょう。

■ jQuery の利用

HTML5やCSS3でまだ実現できない動きはまだまだあります。

ページを重くするという問題はあるかもしれませんが、それよりも優先すべき「使い易さの追求」なのであれば、上手くjQueryを取り入れてみましょう。

■ Googleアナリティクスの利用

ページのアクセス数や表示スピード、滞在時間等のデータを取得出来るGoogleアナリティクスは無料で利用することが出来ます。数字から解析しより良いコンテンツを作成しましょう。