

CSS 基礎

- ポートフォリオサイト制作 -

このパートの流れ

1、CSS を書き始めるまでの準備

2、CSS のルールを知り、実際に装飾をしてみる。

3、デベロッパーツールを使いこなす！

4、CSS レイアウトを行ってみる！

完成形の確認



About Me



My Name is

Webサイトなどの制作を中心に、Webデザインなどを教えるスクールで講師業を行なうなど、活動分野は多岐に渡る。最近では、Webメディアへの記事執筆・寄稿などライターとしての活動も行っている。趣味は旅と写真・イラスト・ハウス作りで、仕事やプライベートを問わず日本全国様々な場所に足を運び、分野を超えた交流を楽しんでいる。

Works

あけましておめでとうございます



オリジナルイラストを販売を作成しました！



マッサージ院のWebを作りました！

2018/2/6



jQuery Arrownを作成しました！

2018/1/1



ブログArrownを作成しました！

2017/8/24

[作品集をもっと見る](#)

Contact

お名前

メールアドレス

お問い合わせ内容 制作のご依頼 イベント出演依頼 その他

お問い合わせ詳細

[送信](#)



CSS 基礎

- CSS を書き始める準備と前提知識 -

1、CSSとは？

HTML・CSS・JavaScript の関係

HTML

HTMLはもともと文書。「タグ」というもので文字情報を囲むことに
よって文書構造や意味合いを定義。

CSS

Web サイトの見た目・レイアウトを記述・定義。

装飾・・・色を変えたり、文字を太くしたり。
レイアウト・・・横並びにしたり、中央寄せにしたり。

JavaScript

ブラウザに対してユーザー（人間）が何かしらのアクションを起こ
した時に発生する動きを定義。
(ページ読み込み、クリック、マウスオーバーなど)

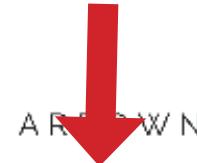
Arrown

Arrown（アロウン）「仕事を作る・生み出す」の実現を目指す、クリエイターのブログ

フォローする

- 著者の活動紹介
 - はなまき・おはらい
 - 千葉紹介
 - ジーベアナデラー
- プログラミング
 - Gccole Apps Script
 - HTML&CSS
 - JavaScript
 - Ruby on Rails
 - WordPress
 - 事件ツール
 - 事件知識
- 健康法・数学回復法
- Webサイト＆ブログ運営
- お役立ち
- 篠行・誕生日＆グッズ販売
 - LCC - 楽安移動
 - グルメ・おいしい豆知識
 - ゲストハウス
- Arrown ニュース

HTMLのみ



Arrown（アロウン）「仕事を作る・生み出す」の実現を目指す、クリエイターのブログ

著者の活動紹介 プログラミング 健康法・数学回復法 Webサイト＆ブログ運営 お役立ち 篠行・誕生日＆グッズ販売 Arrown ニュース



HTML+CSS

未経験から地方でWebデザイナーに!? 桐井
県大野市でWebクリエイター育成セミナー開
催をしています!【2016-2018活動報告】

○ 2017/7/24 ■ 県立教育会館

好みでのブログArrownでは、着石であるコスガの道
も販売させていただいているのですが、今更に、東
北県大野市でのWebクリエイター育成セミ...
[詳しく見る](#)

ブログ内を検索

人気の記事



MacBook Proの中古を秋葉原で購入!中古
Mac購入検討時におすすめな初心者向け店舗
3選

○ 2016/2/24 ■ MacBookスタートアップ記事一
止

先日、MacBook Pro 2017の15インチサイズのものを購入しました。MacBook Airの軽さが比較的決まり手...
[詳しく見る](#)



CSS = HTML に対して命令を行うもの

「目印に対して○○をする（例：フォントサイズを大きくする）！」

「**命令したい場所**」を指定して「**命令する内容**」を書く！！！

CSS は命令の対象が HTML になります。そのため、

- ・どの HTML タグに対して、命令をしたいのか？

をはっきりとさせるため、HTML タグが持っている**タグ名**やその他の目印を手がかりにして、

CSS を適用していきます。



CSS 単体では何も起きない。。。

2、CSS作成までの流れ

CSS 作成→書き始めるまでの流れ

CSS を作成し、書き始めるまでの手順を覚えておきましょう。



CSS 作成→書き始めまでの 5 つのステップ

- 1、「○○○ .css」というファイルを作成。
- 2、作成した CSS ファイルの冒頭に `@charset "utf-8"` と記入。
(ファイル、フォルダの整理に注意！！)
- 3、HTML ファイルに `reset.css` (normalize.css・sanitize.css など) を読み込む。
- 4、HTML ファイルに自身で作成した CSS ファイルを読み込む
`<link rel="stylesheet" href="CSS ファイルの場所" >`
を記入。
- 5、CSS 書き始める！！！

reset.css

reset.css ??

Web サイトは、PC からスマートフォンまで様々な端末で閲覧されますが、ブラウザに関しても Google Chrome、Firefox、Internet Exploler など様々な種類のブラウザで閲覧されます。

それぞれのブラウザは独自に CSS を持っていますが、ブラウザごとで同じタグに対して違う CSS を適用するケースがあります。
(例：Safari はデフォルトの書体が明朝体、Chrome はゴシック体)



ブラウザデフォルトの CSS を見てみよう !!

The screenshot shows the Chrome DevTools interface with the 'Styles' tab selected. The left panel displays the CSS rules for the 'body' element, which includes 'display: block;' and 'margin: 8px;'. These lines of code are highlighted with a red box. The right panel shows the 'user agent stylesheet' section, which contains the rule ':hov .cls + user agent stylesheet'. The entire screenshot is framed by a light gray border.

```
body {  
  display: block;  
  margin: 8px;  
}  
  
:hov .cls + user agent stylesheet
```

デベロッパーツールより抜粋。Google Chrome の場合、Chrome 自体
が持っている CSS によって body タグに margin:8px が設定されている。



reset.css ??

ブラウザごとのデフォルト CSS の差をいちいち考慮しながら CSS を書いていると、その分労力が増えてしまう。。。



「reset.css（リセット CSS）」と呼ばれる種類の CSS を HTML に読み込ませて、
ブラウザごとの差をなくした状態で CSS を書き始めるのが一般的 !!



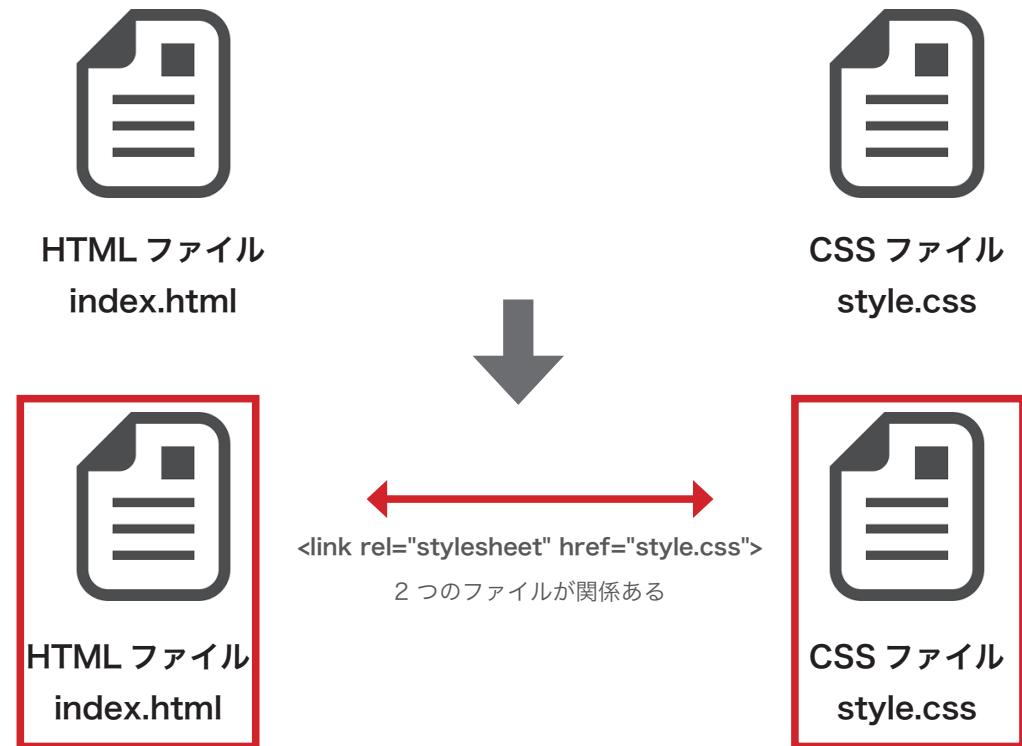
リセット CSS の種類について

- **reset.css**
- **normalize.css**
- **sanitize.css**

「reset.css 2018」などで検索してみると、新しい情報を常々インプットで
きると思います！

※上記の違いは、ブラウザごとが持つ CSS の差分の埋め方の違いです。

CSS を HTML に読み込む



CSS の読み込みについて

CSS ファイルが複数存在する場合もあるので、HTML ファイルに「**この
HTML はどの CSS と関係があるのか**」ということを明示的に書いてあげないと、CSS にいくら命令を書いても HTML の見た目は何の変化もしないのです。

CSS を HTML に読み込む

HTML

```
<!DOCTYPE html>

<head>

// 下記は一例、赤部分は状況に応じて変化

<link rel="stylesheet" href="css/reset.
css">// 先にリセット CSS 系の種類

<link rel="stylesheet" href="css/style.
css">// 後から自分で作成した CSS の種類

</head>

<body>

</body>

</html>
```



href の指定の仕方

- 相対パス
- 絶対パス

HTML で学んだ上記 2 つの指定方法を使います。理解があやふやな方はここでしっかりと固めておきましょう～！

これで CSS 準備完了！

3、CSSの記述ルールの基本

CSS の指定方法の基本

```
セレクタ {  
    プロパティ名 : 指定したい内容 ;  
}
```

※例

```
h1{  
    background-color:#f00;  
    /*h1 タグの背景色を赤色にする、  
    という命令です */  
}
```



注意！

全角スペースが入っていたり `{}` の閉じ忘れや `「;」` の書き漏れがあると、
CSS がうまく効かなくなります。

CSS の指定方法の基本

```
h1 {  
    color: #f00;      —→ H1タグの文字色を赤色にしたい！  
}
```

【セレクタ】

「命令したい場所」を表す。

例)

h1 … タグの名前

.title … class という種類の目印

【プロパティ】

「何の命令をするのか」を表す

例)

文字色を変える、余白を空ける…

【値】

プロパティをどうしたいのか？を指定する。

例) 緑色にする、font-size:16px に！

セレクタの種類

「HTML のタグの名前」「class の名前」「id の名前」など HTML に書いてある情報を手がかりにセレクタを指定し CSS の命令を書きます。



id と class

■ ID

1つのHTML内で同じ名前を2回以上使えない。主にJavaScriptを使用する際に使用。

■ class

1つのHTML内で**同じ名前を2回以上使用できる**。主にCSSを指定するときに使用。

※ id も class も数字で始めるのは NG。大文字と小文字を区別します。

※ classは半角スペースで区切り複数のclass名適用可能です。(例→ class="title title-ja")



id と class の付け方

■ HTML

```
<a href="#" id="point" class="text" > リンク </a>
```

■ CSS

#point …… id を目印に指定する時の記述例。#(シャープ)+id名で指定

.text …… classを目印に指定する時の記述例。.(ドット)+class名で指定

CSS の適用順位の原則

HTML

```
<!html>
<doctype html>
<head>
  <meta charset ="utf-8">
</head>
<body>
  <h1 class="page_title" id="title">タイトル</h1>
  <p> これは文書です。</p>
</body>
</html>
```

CSS

```
#title{
  color: red;
}

.page_title{
  color: blue;
}
```

例えば左記の例のように、CSS は同じ HTML に対して複数箇所から命令をすることが可能です。

同じ HTML から複数箇所から CSS が命令されている場合、どちらが優先的に適用されるのかを決めるルールが CSS には用意されています。**(CSS の優先順位の原則)**

原則：下に書いたものが優先的に適用

HTML

```
<!html>
<doctype html>
<head>
    <meta charset ="utf-8">
</head>
<body>
    <h1 class="page_title" id="title">タイトル</h1>
    <p> これは文書です。</p>
</body>
</html>
```

CSS

```
h1{
    color: red;
}

.page_title{
    color: blue;
}
```

h1 の色は青になる！

CSS の詳細度によっては記述順が無視される

HTML

```
<!html>
<doctype html>
<head>
    <meta charset ="utf-8">
</head>
<body>
    <h1 class="page_title" id="title">タイトル</h1>
    <p>これは文書です。</p>
</body>
</html>
```

CSS

```
#title{
    color: red;
}
```

```
.page_title{
    color: blue;
}
```

※ 詳細度の比較例

id で指定 > class で指定 > タグ名で指定

詳細度により、上に書かれても、#title に
書かれた CSS が適用される。

h1 の色は赤になる！



詳細度って？？

id、class など様々なセレクタを使って CSS の命令を聞かせ
ることができるが、セレクタの種類によってポイントみたいな
ものが割り振られていて、**そのポイントの合計が高い = 詳細
度が高い**となる。

※ id の方が class よりポイントが高い。

CSS レイアウト

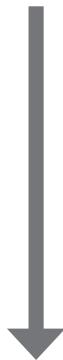
- ページコーディングしてみよう -

CSS コーディングの流れ



CSS コーディングの流れ

- 1、全体の構成・サイズ感の確認
- 2、全体のベースとなる CSS 設定
- 3、全体の大枠レイアウトの調整
- 4、各セクション・パーツの調整
- 5、各環境での確認・修正



※今回は、PCサイトコーディング→スマホコーディングの順番に進めていきます。PC コーディングで上記 1-5 を行ったらスマホコーディングで同様に 1-5 を行っていきましょう。

◆ PC 版

- ・まずはサイズ固定でコーディング設定

◆スマホ版

- ・PC 版含めて、全体がサイズ可変になるように調整。
- ・レスポンシブ Web デザインによるコーディング

What's レスポンシブ Web デザイン ??



レスポンシブ Web デザインとは

非レスポンシブ



PC 用 HTML



SP 用 HTML

PC は PC、SP は SP
の HTML を作成。それ
ぞれに CSS が命令。

レスポンシブ



HTML は 1 つ



CSS

CSS の中で PC 用と
SP 用の見栄えを分けて
記述

1、全体の構成・サイズ感の確認 (PC版)



- ・画像はブラウザ横幅に対して 100% 可変
- ・画像に対して上下左右中央にテキスト配置
- ・タイトルとナビゲーションは両端（均等）揃え、背景色は横幅いっぱいに広がる。



- ・About Me 青枠は 560px でブラウザ全体に対して中央配置
- ・タイトル（以下同様）やキャラクター画像、名前は中央揃え



- ・Works3つは、3分割（3カラム）横並びレイアウト
- ・ボタンは赤枠エリアに対して中央配置。
- ・背景色はブラウザ横幅一杯に広がる。



- ・Contact 青枠は 560px で赤枠エリアに対して中央配置
- ・テキストは中央揃え、背景色は横幅いっぱいに広がる。

全体の構成・サイズ感の確認



◆ PC 版と SP 版の基本的な境目（ブレイクポイント）

- SP … 640px 以下のデバイス
- PC … 641px 以上のデバイス

（タブレットも考慮するなどの場合はブレイクポイントが増えることも！Webサイトごとに仕様確認！）

今回のようなシングルカラムレイアウトでは、全体の大枠レイアウトを調整するというよりも、各セクションごとにコーディングを進めて行く方がやりやすいでしょう◎）

2、全体のベースとなる CSS 設定

まずはページ全体に共通する CSS の設定をしていきましょう！

CSS の記述・指定の基本を確認する機会にもなります。

主な設定内容

- ・フォントの種類
- ・ベースとなるフォントのサイズ
- ・ベースとなる文字の色
- ・行間
- ・a タグのリンク設定や色
- ・画像に関する設定
- ・背景画像や背景色があればその設定

2、全体のベースとなるCSS設定 -font-size-

```
body{  
    font-size :15px;  
}
```

font-size は**文字の大きさを指定する CSS プロパティ**です。

特に指定がない場合、ブラウザのデフォルトは**16px**で指定されています。



font-size の注意点

font-size に指定できる単位は多々存在します。(下記は代表例)

- px
- %
- em
- rem
- vw
- vh

font-size の指定事例 rem

CSS

```
html{  
  font-size:62.5%; /*16px × 62.5% = 10px( 計算しやすいように )*/  
}  
  
body{  
  font-size:1.6rem; /* 10px × 1.6 = 16px */  
}  
  
h1{  
  font-size:2.4rem; /* 10px × 2.4 = 24px */  
}
```



font-size の単位 : rem

font-size の単位には様々なものがありますが、px で指定をすると、PC とスマホで font-size を変更する際に 1 つずつ修正しないといけないなどのように、メンテナンス性が非常に悪いものになります。

そこで出てきたのが **rem** という単位です。

rem は、html 要素 (root 要素) に対して指定された font-size を基準として「何倍の font-size になるか」を **倍率で指定します。**

2、全体のベースとなるCSS設定 -line-height-

```
body{  
    line-height :2;  
}
```

line-height は **行間を指定する CSS プロパティ** です。

原則、倍率を表す数値のみ記述しておくと汎用性が高く便利です。



line-height の注意点

line-height が表す部分について、確認しておきましょう。

(下記は font-size:16px、line-height:2 の場合です。)

この文章はダミーです。文字の大きさ、量、字間、行間等を確認するために入れ
ています。



The diagram shows a horizontal line divided into three segments by two vertical arrows. The top segment is labeled "32px" and the bottom segment is also labeled "32px". This visualizes how the line-height property (32px) is distributed between the lines of text (16px font-size).

16px (font-size) ×2 (line-height) = 32px が 1 行の高さになる！

(余白が文字の上下に均等に割り振られる)

2、全体のベースとなるCSS設定 -img-

```
img{  
    max-width: 100%;  
    height: auto;  
}  
  
max-width:100%・・・画像が持つ横幅より大きくなることはない & それ以下の時は縮む。  
height:auto .....縦横比率を維持して画像が可変する。
```

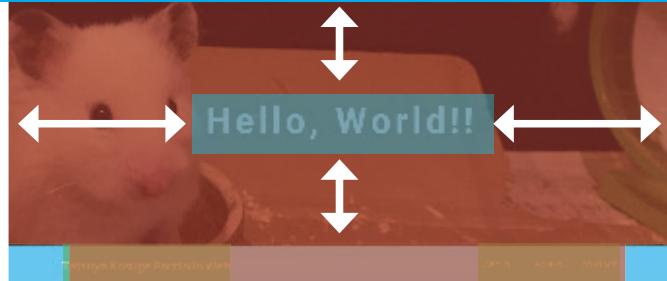


フルードイメージ

画像はそれぞれ固有の縦横サイズを持っているため、例えば画像の横幅よりも小さいスマート端末でWebサイトを見る場合、横スクロールが生じて使いづらいサイトになってしまいます。

これを回避し、ブラウザや親要素の横幅に応じて画像を可変させるようする方法が[上記の記述 = フルードイメージ](#)です。

3、全体の大枠レイアウトの調整



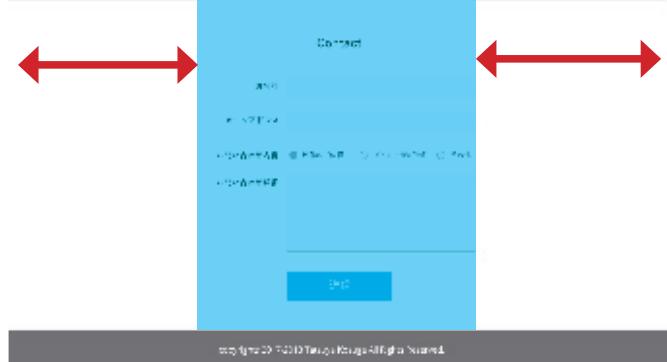
- ・画像はブラウザ横幅に対して 100% 可変
- ・画像に対して上下左右中央にテキスト配置
- ・タイトルとナビゲーションは両端（均等）揃え、背景色は横幅いっぱいに広がる。



- ・About Me 青枠は 560px でブラウザ全体に対して中央配置
- ・タイトル（以下同様）やキャラクター画像、名前は中央揃え



- ・Works3つは、3分割（3カラム）横並びレイアウト
- ・ボタンは赤枠エリアに対して中央配置。
- ・背景色はブラウザ横幅一杯に広がる。



- ・Contact 青枠は 560px で赤枠エリアに対して中央配置
- ・テキストは中央揃え、背景色は横幅いっぱいに広がる。

4-1・Aboutエリア

About エリアのコーディング



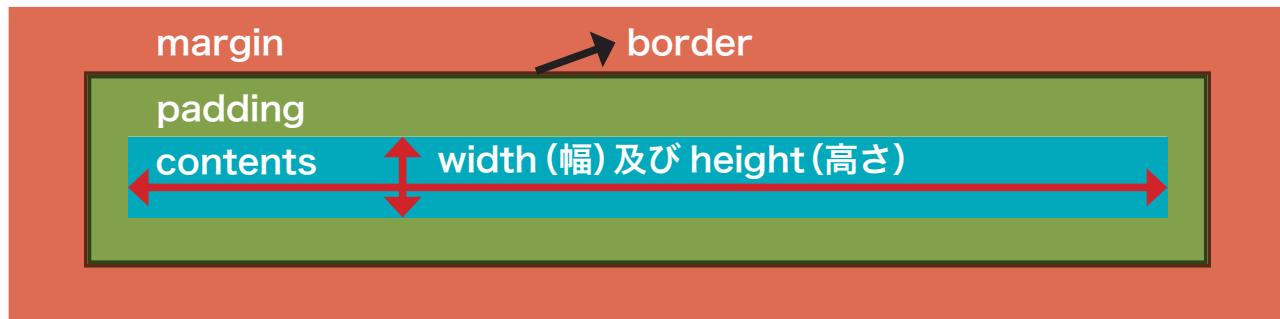
本来ならば上から順に CSS レイアウトを組んで行くところですが、CSS レイアウトの理解の段階を踏むために、About Me から先にコーディングしていきます。



About Me エリアのポイントと、学ぶ要素

- ・HTML 同士の基本的な余白の空け方、取り方。(ボックスモデルの理解)
- ・中央配置の方法
(ブラウザ全体に対して青のエリアが常に中央配置。幅は 560px)
- ・テキストや画像の中央寄せ方法

全体の大枠レイアウトの調整・ボックスモデル



HTMLは全て四角い箱・ボックスで成り立っています。ボックスは、
「中身（テキストなど）・内余白・枠線・外余白」
の4つで構成されています。
この構成要素を、**ボックスモデル**と言います。



ボックスモデルで知っておきたいポイント

- ・ボックスのcontents部分には、テキストや画像などが入ります。

ブロックレベルボックス



ボックスには大きく分けて 2 種類あります。

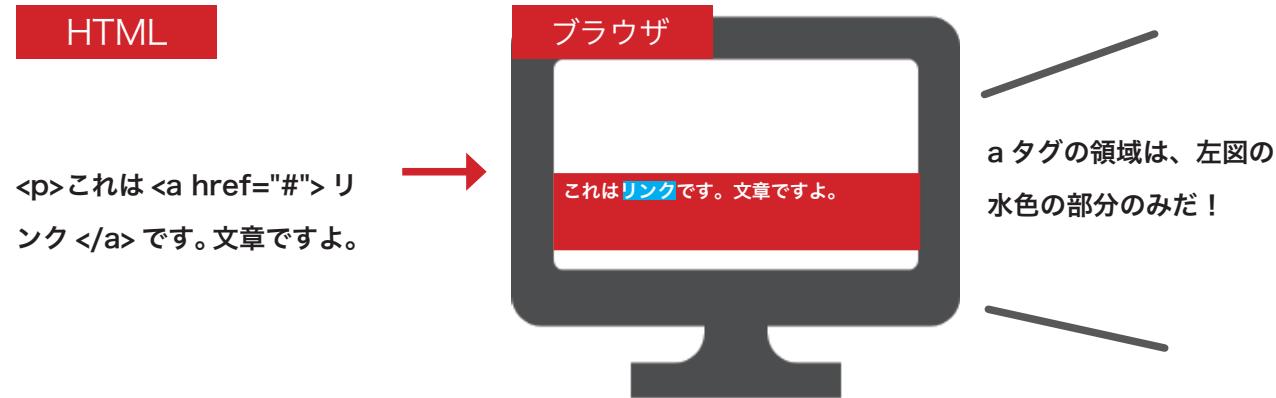
h1、h2 や p タグなどは、ブロックレベルボックスという種類に該当します。



ブロックレベルボックスの特徴

- margin、padding、width、height を自由に調整可能。
- 幅を指定しなければ横幅が親要素いっぱいに広がる。（width:100% が通常）
- CSS の display:block プロパティがデフォルトで適用されている。

インラインレベルボックス



ボックスには大きく分けて 2 種類あります。

a タグなどは、インラインレベルボックスという種類に該当します。



インラインレベルボックスの特徴

- ・ width、height を指定できないものがある。
- ・ 上下の margin を指定できない。
- ・ 特に指定をしないと、ボックスの幅は中身に含まれる要素の大きさ分になる。
(width)

marginとpaddingの基礎



余白の取り方の基本・marginとpaddingの適用方法

margin: 10px; (上下左右全て margin10px 設定)

margin: 10px 20px; (上下 margin10px・左右 margin20px 設定)

margin: 10px 20px 30px; (各 margin→上10px・左右 20px・下 30px 設定)

margin: 10px 20px 30px 40px;

(各 margin→上 10px・右 20px・下 30px・左 40px 設定。時計回り)



marginとpaddingの違いと注意点

- margin で生じた余白にはその HTML の背景色・背景画像が塗られないが、padding で生じた余白には適用される。
- margin-top:20px; padding-right:30px** のように方向ごとに指定も可能。

margin の注意点・その1

The screenshot shows a web page with a dark grey header and footer. In the center, there's a white content area. At the top of this area, the text "About Me" is displayed in a small, bold, black font. Below it is a small, colorful cartoon illustration of a person wearing a red beanie and a striped shirt. Underneath the illustration, there's an

element with the text "My Name is" in a large, bold, black font. A thin blue horizontal bar runs across the width of the element. Below the , there's a paragraph of text in a smaller black font. The text describes the author's background, mentioning work at a school teaching website design, various hobbies like travel and writing, and a desire to meet people across Japan.



margin が重なりあうときは相殺される (padding はされない)

例えば上記の例の場合、下記の CSS 指定をすると・・・

- h3 の margin-bottom に 40px
 - p タグの margin-top に 20px
- 青色の領域は、 $40+20=60\text{px}$ にはならず、大きい方の 40px が適用されます。

margin の注意点・その2



margin が親要素をこえてみ出すケースがある

例えば上記の例の場合、下記の CSS 指定をすると・・・

- h2 の margin-top に 40px

→青色の領域は、まるで黒枠 (div タグ) の margin-top かのように適用される。

(div タグに border や padding が付いている場合はこのようになります)

margin と padding の使い分け



margin と padding の使い分け

前述した margin の注意点 2 つを踏まえると、

- ・隣合う HTML 同士の余白を空けるのは margin
- ・親要素と子要素の余白を空けるのは padding
- ・margin を使用する際は極力、上下方向どちらかに統一して適用する
- ・余白を空ける部分の背景に色や画像を表示させたいときは padding

以上 4 つの使い分けポイントが見出せそうです。

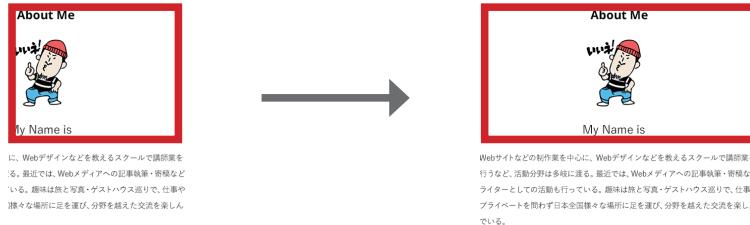
(時には上記に当てはまらない時もありそうですが。。)

テキストや画像の中央寄せ

text-align : center;

ボックスの中に入っているテキストや画像を中央寄せにしたい場合は、

text-align という CSS プロパティを使用し、値に **center** を適用します。



	A	B	C	D
1	名前	出身	性別	
2	コスゲ	埼玉	男	
3				
4				

Excel でそれぞれのセル(ボックス)の中身の
テキストを中央寄せにした時の状態。

text-align:center はこの状態と似ています。



text-align は以下の値を持たせることができます。

- **center:** 中央寄せ
- **right:** 右寄せ
- **left :** 左寄せ

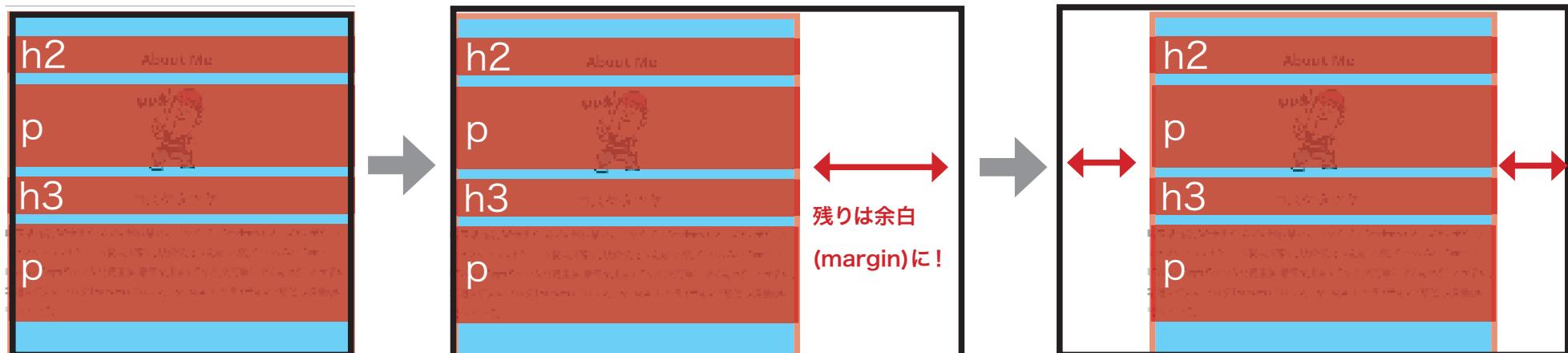
デフォルトでは左寄せ、つまり left が適用されています。

ボックスの中央配置（親要素に対して）

1. divタグで4つのHTMLを囲む

2. divタグに幅を指定する (max-width:560px)

3. divタグに **margin:auto** を適用し、余白が
ボックスに対して左右均等になるようにす



divタグを使ったボックス中央配置

ブラウザ全体あるいは親要素に対して HTML のボックスそのものを中央寄せにしたい場合、該当の HTML をまとめて div タグで囲み、div タグに対して

- **width プロパティで幅を指定**
- **margin: auto (margin: 0 auto) でボックス左右の余白を常に均等にする**

方法を用いるのが一般です。

※ div タグはブロックレベルボックスで、タグ自体に意味はありません。

4-2・Worksエリア

Worksエリアのコーディング



続いて、Works エリアにうつっていきます！

Works エリアでは、先ほど About エリアで学んだことに加えて、横並びレイアウトの概念が出てきます！



Works エリアのポイントと、学ぶ要素

- ・リンクボタンのデザインと中央寄せについて
- ・横並びの方法

※まずは About エリアで学んだ内容を生かして、タイトルの中央揃えと、Works エリア内の HTML 全体の中央配置を行ってみましょう！

リンクボタンのデザイン



HTML

```
<a href="#" class="btn">作品集をもっと見る</a>
```

CSS

```
.btn{  
    display:block; /* インラインレベルボックス→ブロックレベルボックス */  
    /* 以下ボタンのデザインに関する CSS */  
}
```

リンクボタンには a タグが使用されていますが、a タグはインラインレベルボックスの仲間なので、widthや上下余白を適用できません。

display プロパティを使用すると、CSS でインライン→ブロックレベルボックスに変更し、a タグでも width や上下余白の適用が可能になります。



display プロパティについて

- ・ block、inline、table 等様々な値が適用可能です。HTML がそれぞれ持つ元々の挙動を変更することができます。
- ・ **display:none;** を適用すると、その要素を非表示にすることができます。

できたら、ボタンの中央配置を行ってみましょう！

横並びレイアウトの実施



HTML は、CSS が適用されていない状態では、基本的に縦に積み下がって行く形になります。そのため、HTML 要素同士の横並びを実現するには、CSS の力が必要です。

■ブラウザ対応によって分かれる横並びレイアウト実施の判断

(Internet Exploler のバージョン対応次第)

- Internet Exploler11 の対応・・・Flexbox で OK
- Internet Exploler10 のカバーも必要・・・Flexbox 使用に条件がつく
- Internet Exploler9 のカバーも必要・・・flexibility.js 使用 or float で対応

float

画像

ここにテキストが入り
ます。ここにテキスト
が入ります。ここに
テキストが入ります。ここにテキストが
入ります。ここにテキストが入ります。
ここにテキストが入ります。ここにテキ
ストが入ります。ここにテキストが入り

→ 画像に float:left を適用！



float について

float は、昔からある CSS の横並びレイアウトの代表格です。

本来は上記のような回り込みレイアウトの際に使用されるためのものでしたが、他に横並びを実現するための方法がない、という理由から、float で横並びレイアウトの実施が伝統的に行われてきました。

現在も、かなり古いブラウザまで対応しないといけない場合は、float を用いて横並びレイアウトを実施することができます。

flexbox

HTML

```
<ul class="list">
  <li>リスト1</li>
  <li>リスト2</li>
  <li>リスト3</li>
</ul>
```

CSS

```
.list{
  display:flex;
  /* 単純な横並びを実施するにはこれだけ ! */
}
```



flexboxについて

flexbox は、float に代わり出てきた CSS レイアウト手法です。

- ・単純な横並び
- ・複雑な横並び
- ・上下中央レイアウト

などなど、様々なことが flexbox を使うと可能になります！

「**横並びをしたい HTML を囲んでいる HTML（親要素）に対して
display: flex を適用する**」

これが flexbox の大原則になります。

それ以外に抑えておきたい 3 つの基礎を次ページから見ていきます！

flexbox 理解までの5段階

1、親要素と子要素の理解

HTML パートの冒頭で触れました。これが flexbox を理解する上での前提条件になります。

2、display: flex; を使ったシンプルな横並びができる

Works エリアのコーディングを通じて学習します。概念を理解する上での3つの基本を抑え、display:flex を使ったシンプルな子要素の横並びができます。高さは全て揃います。

3、flex コンテナ（親要素）に適用する CSS プロパティを使いこなせる

navigation エリア・hero-image エリアのコーディングを通じて学習します。子要素の横方向・縦方向のレイアウトルールを決定します。対象例) justify-content・align-items

4、flex アイテム（子要素）に適用する CSS プロパティを使いこなせる

3で決めたルールからはみ出した（例外となる）子要素のレイアウトを行うことができます。対象例) align-self・order

5、flexbox を極める（flex アイテムプロパティ上級編）

flexbox を極めるポイントとなる flex プロパティを取得することで、様々なレイアウトができるようになります。

flexbox3つの基礎・その1「コンテナとアイテム」

HTML

```
<ul class="list">
  <li>リスト1</li>
  <li>リスト2</li>
  <li>リスト3</li>
</ul>
```

CSS

```
.list{
  display:flex;
  /* 単純な横並びを実施するにはこれだけ */
}
```



flexboxにおけるコンテナとアイテムについて

flexboxを学習していると、「コンテナ」「アイテム」という言葉が出てきます。

- ・コンテナ・・・親要素（横並びさせたいHTMLを囲んでいるHTML）
- ・アイテム・・・子要素（横並びさせたいHTML）

自分でflexboxについて調べていると、ほぼ間違いなく出てくる用語なので、
しっかりと抑えておきましょう！

flexbox3つの基礎・その2「孫要素に影響しない」

HTML

```
<div class="navigation">
  <ul>
    <li>リスト1</li><li>リスト2</li><li>リスト3</li>
  </ul>
</div>
```

CSS

```
.navigation{
  display:flex;
  /* この場合、li は横並びにならない */
}
```



display:flex は孫要素には影響しない

flexbox を使用する際、display:flex を使用することは前述した通りですが、

孫要素にまで影響しない点に注意です！

例えば、上記のようなコードがあり、li タグを横並びにしたい場合、li タグか

ら見た親要素→ul タグ、になるので、ul タグに display:flex をかけないと、

横並びになりません！

<div class="navigation"> に display:flex をかけても、li タグが横並びになるわけではないので注意が必要です！

flexbox3つの基礎・その3「主軸と交差軸」



主軸と交差軸

デフォルトでは横軸を主軸、縦軸を交差軸と表現します。

flexbox 関係の CSS プロパティの 適用状況によって、色々複雑に状況が変わるので注意が必要です。

Worksエリアのコーディング



前ページまでの内容ができたら、細かい調整をして、Works エリアを完成させていきましょう！



確認しておきたいCSSプロパティ一例

- font-size
- background-color
- color
- font-weight
- margin
- padding

4-3・navエリア

navigation エリアのコーディング



navigation エリアのポイントと、学ぶ要素

- flexbox を使った様々なレイアウト方法

navigation エリアのコーディングを通じて、flexbox を使った様々なレイアウト方法の基礎を学んでいきます。

flexboxでは様々なCSSプロパティが出てきますが、navigation エリアのコーディングで学ぶ際に出てくるのは全て **コンテナ(親要素)に適用する CSS プロパティ**になります。

※まずは、赤枠エリアに該当する部分が中央配置になるように、これまで学んだ内容を使ってレイアウトしてみましょう。

flexbox 理解までの5段階

1、親要素と子要素の理解

HTML パートの冒頭で触れました。これが flexbox を理解する上での前提条件になります。

2、display: flex; を使ったシンプルな横並びができる

Works エリアのコーディングを通じて学習します。概念を理解する上での 3 つの基本を抑え、display:flex を使ったシンプルな子要素の横並びができます。高さは全て揃います。

3、flex コンテナ（親要素）に適用する CSS プロパティを使いこなせる

navigation エリア・hero-image エリアのコーディングを通じて学習します。子要素の横方向・縦方向のレイアウトルールを決定します。対象例) justify-content・align-items

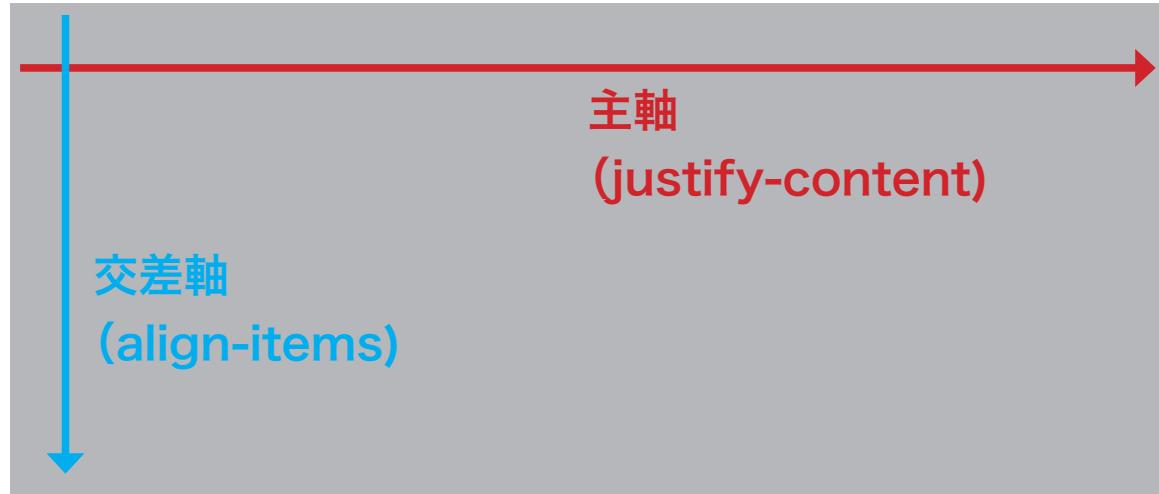
4、flex アイテム（子要素）に適用する CSS プロパティを使いこなせる

3 で決めたルールからはみ出した（例外となる）子要素のレイアウトを行うことができます。対象例) align-self・order

5、flexbox を極める（flex アイテムプロパティ上級編）

flexbox を極めるポイントとなる flex プロパティを取得することで、様々なレイアウトができるようになります。

justify-content と align-items



justify-content と align-items がおこなっていること

justify-content と align-items は、flex アイテムつまり子要素の整列ルールを決めるプロパティです。3D の概念はないので、横軸と縦軸の 2 方向に関する整列のルールを決定します。

- **justify-content** . . . 主軸の整列ルールを決める（横方向）
- **align-items** . . . 交差軸の整列ルールを決める（縦方向）

親要素の中にある子要素は上記で決めたルールに従い配置されます。

justify-content と align-items



flex-direction との兼ね合い

「横軸」「縦軸」ではなく、なぜわざわざ「主軸」「交差軸」という言い方をするかというと、flex-direction という CSS プロパティに **column** あるいは **column-reverse** が適用されている場合、縦横が逆になるからです。

- **justify-content** . . . 主軸の整列ルールを決める（縦方向）
- **align-items** . . . 交差軸の整列ルールを決める（横方向）

※デフォルトは、flex-direction:row です。

justify-content

justify-content: flex-start; (初期値)



justify-content: flex-end;



justify-content: center;



justify-content: space-between;



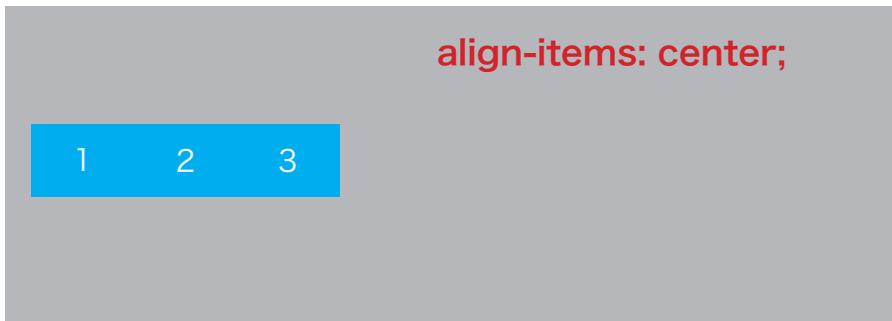
※赤矢印は同じ大きさに。2つの場合は両端揃えのようになる。

justify-content: space-around;



※青矢印が赤矢印の半分の大きさ

align-items



navigation エリアのコーディング



flexbox を 2 度使用して navigation エリアを完成させよう！

justify-content と align-items を使えば、navigation エリアのレイアウトを完成させることができます！

余白・文字の大きさ・背景色なども含めて、細かい調整を施し、navigation エリアのレイアウトを完成させていきましょう！

※ flexbox を 2 度使用する必要があります。

4-4 • contact エリア

contact エリアのコーディング



contact エリアでのポイント・学ぶ点

- flex-wrap を使用した flex-box の折り返し有無の指定

※まずは。青枠エリア（560px 相当）が全体の中央配置になるように、これまで学んだことを生かしてコーディングしてみましょう。

contact エリアのコーディング

お名前
メールアドレス
お問い合わせ内容
○制作のさくらんぼ
○依頼
○イベント出演依頼
○紹介
○その他
お問い合わせ詳細
送信

ひたすら横に並び、下に折り返さない



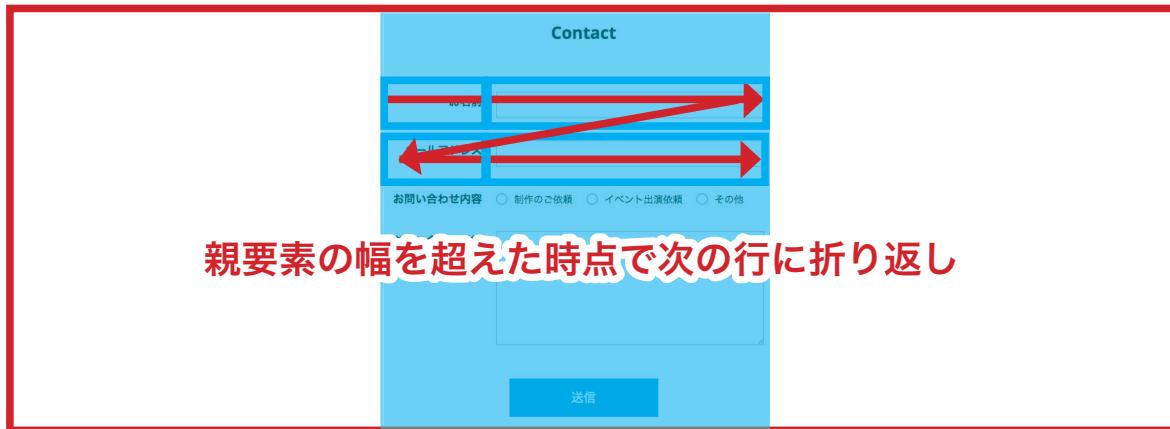
contact エリアでのポイント・学ぶ点

青枠と緑枠はそれぞれ、174px と 386px の幅です。

それぞれに width を指定し、親要素である dl タグに display:flex を適用すると、flexbox の仕様上、親要素の中でひたすら横並びになり、先ほど適用した width がまるで意味なきものになります。

折り返しが行われることなく、全て1行のレイアウトとして行われます。

flex-wrap



親要素の幅を超えた時点で次の行に折り返し



flex-wrap

flex-wrap:wrap; を使用すると、子要素の幅の合計が親要素の幅を超えた時点で、次の行に折り返して表示されるようになります。

※デフォルト

flex-wrap:nowrap という値が適用されており、折り返しません。

form 部分の CSS デザイン

- 制作のご依頼
- イベント出演依頼
- その他



- 制作のご依頼
- イベント出演依頼
- その他



form の CSS デザイン

```
-webkit-appearance:none;  
appearance:none;
```

input や textarea タグは、デフォルトだとブラウザごとによって、見栄えが
かなり異なるケースがあります。

これらを一度リセットした上で、自由に CSS を適用し、どの環境でも同じ
ように表示されるようにするため、**appearance:none** を適用します。

form 部分の CSS デザイン

制作のご依頼 イベント出演依頼 その他



制作のご依頼 イベント出演依頼 その他



チェック時のデザインを調整できる擬似クラス「:checked」

```
form-parts__btn:checked{  
    -webkit-appearance:none;  
    appearance:none;  
    background-color:#fdfdfd;  
}
```

ラジオボタンやチェックボックスがチェックされた時の CSS も自由に設定したい場合は、**:checked** という擬似クラスと呼ばれるものを使って実装することができます。（～～がチェックされた時）

border



border プロパティについて

`border: solid 1px #fff`

- ・赤部分・・・線の種類 (dotted : 丸点線、 dashed : 点線、など)
- ・青部分・・・線の太さ
- ・緑部分・・・線の色

枠線は border プロパティで実装することができます。

方向を指定して枠線を指定したい場合は、**border-top** のようにプロパティを記述します。

contact エリアのコーディング



contact エリアの仕上げ

あとは細かい余白、font-sizeなどの設定を実施し、contact エリアを完成させましょう！

4-5・hero-imageエリア

hero-image エリアのコーディング



hero-image エリアのコーディングのポイント・学ぶ点

- 上下左右中央配置 &position

hero-image エリアのコーディングでは、flexbox ではなく「**position**」と呼ばれる CSS プロパティを使って、レイアウトを実施していきましょう！

※ flexbox を使用する場合は 3 行で実現可能です

```
display:flex;  
justify-content: center;  
align-items:center;
```

position プロパティ



hero-image エリアのコーディングのポイント・学ぶ点

- 上下左右中央配置 &position

hero-image エリアのコーディングでは、「**position**」と呼ばれる CSS プロパティを使って、レイアウトを実施していきましょう！

hero-imageエリアのコーディング



HTML

```
<div class="hero-image">
  <figure class="hero-image__eyecatch"></figure>
  <p class="hero-image__text">Hello, World!!</p>
</div>
```



position とは

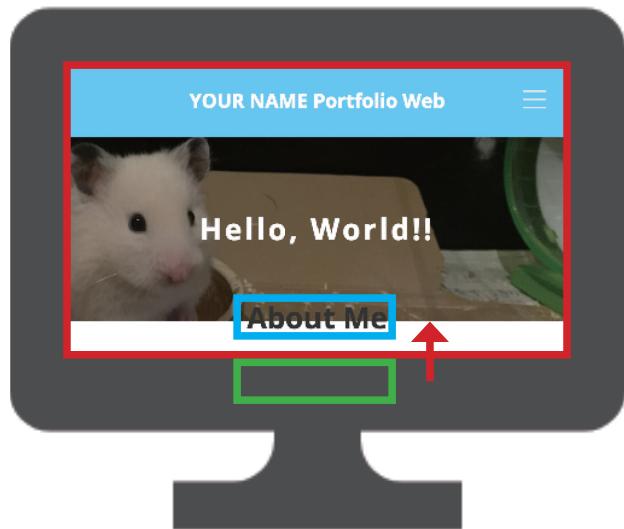
position プロパティは、HTML と HTM を重ねたい時に使用すると効果的な CSS プロパティです。

また、position は「**どこに**」配置するのかを指定するプロパティなので、

- **top**
- **left**
- **right**
- **bottom**

を使って配置位置指定を伴うことがほとんどです。（単位は px や % など）

position: relative;



position: relative

CSS

```
.about-title{  
    position: relative;  
    top: -63px;  
}
```

左記でいうと、本来緑枠あたりにあったタイトル部分が、他の HTML を邪魔することなく上にずらすことができるようになる。(青枠に移動)

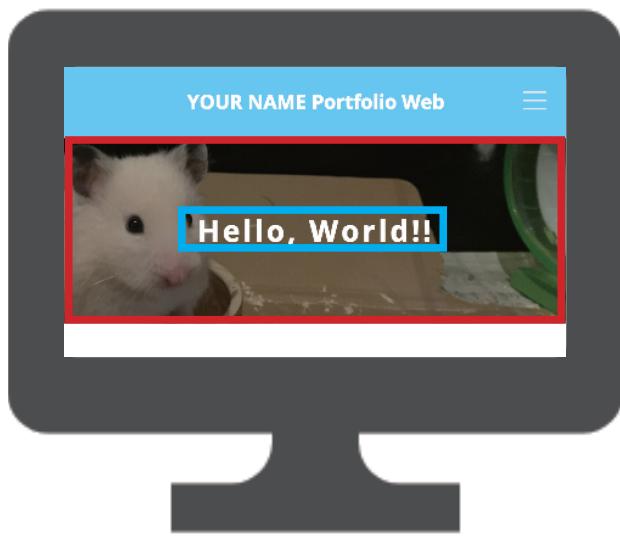
position: relative は相対配置と呼ばれ、単純に元々ある位置から少しだけ HTML をずらして（移動させて）重ねて表示したい場合に使用されます。

赤枠・・・配置位置の基準となるエリア

青枠・・・基準エリアの中で配置したい HTML タグ

※ position: relative を使用した場合、他の HTML のレイアウトに影響を与えることはありません。

position:absolute;



position: absolute

HTML

```
<div class="hero-image">
  <figure class="hero-image__eyecatch"></figure>
  <p class="hero-image__text">Hello,
  World!!</p>
</div>
```

CSS

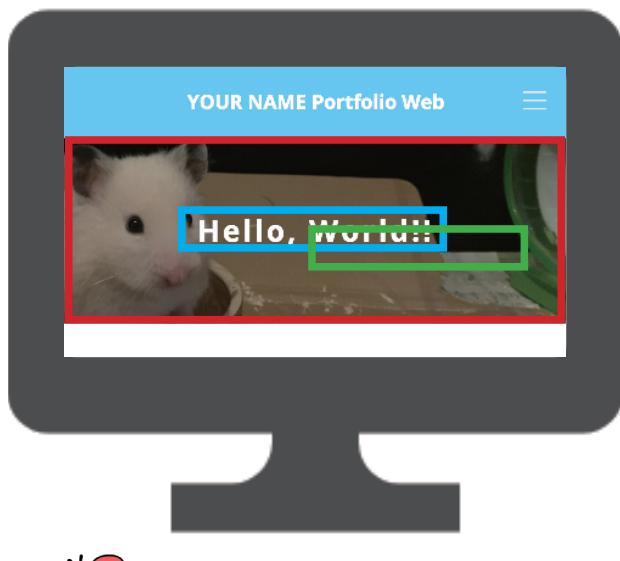
```
.hero-image{
  position: relative;
}

.hero-image__text{
  position: absolute;
  /* 以下、位置指定のプロパティが続く */
  /* .hero-image の範囲内を基準に配置 */
}
```

position: absolute は 絶対配置と呼ばれ、**position: relative** が適用されている親要素の範囲の中で、どこの位置に重ねて表示したいかを指定します。
(親要素に position: relative が指定されているものが全くない場合は HTML 全体を基準にしてどこに配置するか、という考え方になります)

※ position: absolute を使用した場合、他の HTML 要素（特に後ろに続いて書かれている HTML）のレイアウトに影響を与えることがほとんどですので、注意が必要です。

position:relative × position:absolute



position:relative と absolute の組み合わせで上下左右中央配置

HTML

```
<div class="hero-image">
  <figure class="hero-image__eyecatch"></figure>
  <p class="hero-image__text">Hello,
  World!!</p>
</div>
```

CSS

```
.hero-image{
  position: relative;
}

.hero-image__text{
  position: absolute;
  top:50%;
  left:50%;
  transform:translate3d(-50%,-50%,0);
}
```

position:relative と position:absolute とを組み合わせると、上下左右中央配置を実現できます。上記例でいうと

- ・テキスト部分に **top:50%; left:50%;** を適用することでテキストの HTML (緑枠) の左上が、赤枠の上下左右中央に位置する。
- ・**transform: translate3d(-50%,-50%,0);** を適用することで、テキストボックスの横方向・縦方向の半分に相当する量だけマイナス方向に移動。

4-6・footer エリア

footer エリアのコーディング



footer エリアのポイント・学ぶ点

- ・コピーライトのテキスト中央寄せ
- ・positionを使ったページトップボタンの配置

基本的にはこれまで学んだことを使っての仕上げとなります。

まずはコピーライトの中央寄せや基本的な装飾を実現させましょう！

footer エリアのコーディング



position:relative と absolute を使ってページトップボタン配置

HTML

```
<footer class="footer text-center">
  <div class="inner">
    <a href="#" class="btn-top"></a>
    <small class="copyrights">copyrights
      All Rights Reserved.</small>
  </div>
</footer>
```

CSS

```
.footer .inner{
  position: relative;
}
.btn-top{
  position: absolute;
  /* 以下、位置指定して上に移動 */
}
```

CSS レイアウト

- スマホ対応 -

完成形の確認

Your Name Portfolio Web



Hello, World!!

About Me



My Name is

Webサイトなどの制作業を中心に、Webデザインなどを教えるスクールで講師業を行うなど、活動分野は多岐に渡る。最近では、Webメディアへの記事執筆・寄稿などライターとしての活動も行っている。趣味は旅と写真・ゲストハウス巡りで、仕事やプライベートを問わず日本全国様々な場所に足を運び、分野を越えた交流を楽しんでいる。

Works



あけましておめでとうございます

オリジナルイラスト年賀状を作成しました！

2018/2/6



笑顔と健康、美容のために。
1人1人と向き合い、

マッサージ院のWebを作りました！

2018/1/1

Contact

お名前

メールアドレス

お問い合わせ内容

制作のご依頼 イベント出演依頼 その他

お問い合わせ詳細

送信

copyrights 2017-2018 Your Name All Rights Reserved.

Works



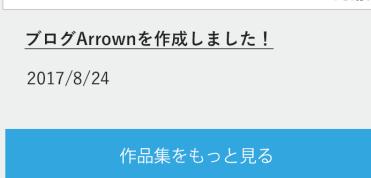
あけましておめでとうございます

jQuery

write less, do more.

ブログArrowを作成しました！

2017/8/24



オリジナルイラスト年賀状を作成しました！

2010/2/6

作品集をもっと見る

ARROWN

Arrown（アロウン）「仕事を作る・生み出す」の実現を目指す、クリエイターのブログ

動画
プログラミング 健康・疲労回復法 ブログ運営 旅・旅行・観光スポット LCC・格安移動 グルメ・おいしい飯

ブログ内を検索

Arrownの中の、

jQuery

write less, do more.

ブログArrowを作成しました！

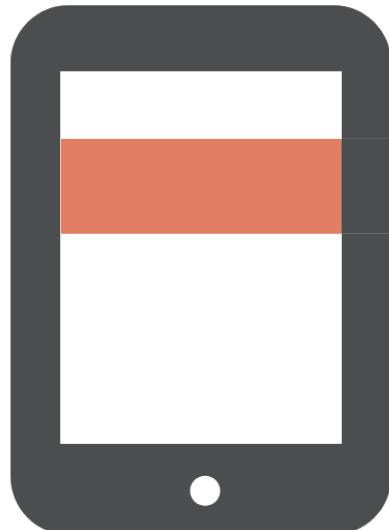
2017/8/24

スマホ対応時の3つのポイント



スマホ対応させる時の3つのポイント

- 1、メディアクエリーや viewport など基本的な設定
- 2、デバッグ環境の用意（極力 iPhone&Android の実機で確認！）
- 3、width などは固定幅にしない（横スクロール発生させない）
(max-width を用いたり、単位に % を用いたりする)



iPhone7・8

端末幅 375px

```
div{  
    width:1140px;  
}
```

固定幅だと
はみ出してしまう！！

1、メディアクエリーと viewport

メディアクエリー

CSS

```
/*PC 時の CSS を記入 */
```

```
@media screen and (max-width: 640px){  
    /*SP 時の CSS をここに記入 */  
}
```

※先に SP の CSS を書き、後から PC 用の CSS を追記する場合もあります。

その際は、max-width → min-width となります。



メディアクエリーの書き方

レスポンシブ Web デザインにおいては、メディアクエリーというものを使用して、PC 用の CSS と SP 用の CSS を切り分けて書くことができます。

- 1、PC 用 CSS を先に記述、SP 用 CSS をメディアクエリーで記入
- 2、SP 用 CSS を先に記述、PC 用 CSS をメディアクエリーで記入

上記の 2 パターンがありますが、今回は先に PC 用 CSS を書いているので、1 番のパターンが該当します。

Viewport

HTML

```
<meta name="viewport" content="width=device-width">
```

※上記を `head` タグの中身に記述します。

※他にも記述できる内容がたくさんありますが、最低限は上記で OK です。



viewportについて

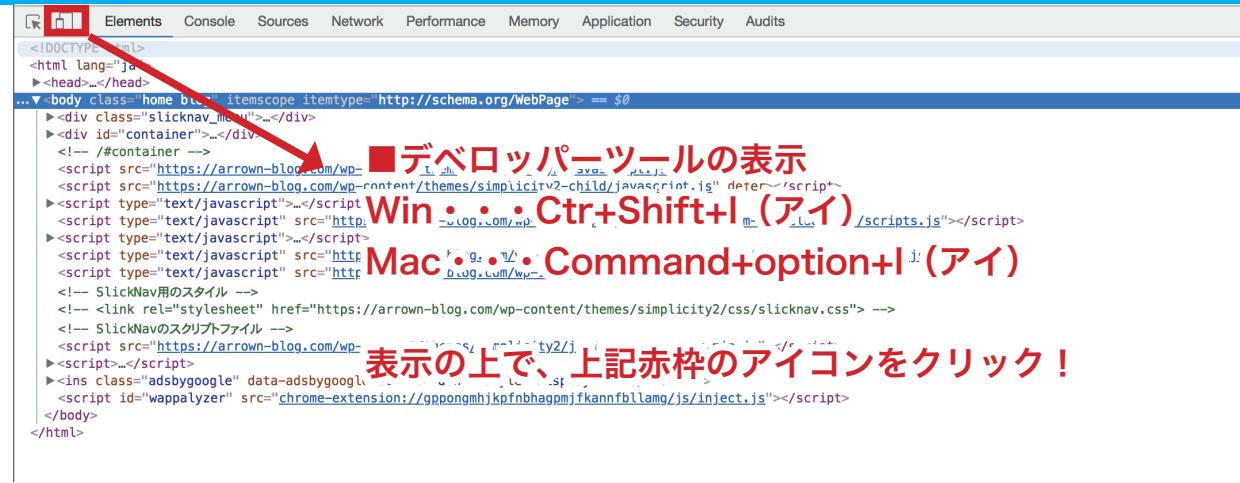
スマートフォンが出た当初は、PCで見た Web サイトの見栄えをそのままスマートフォンの画面に埋め込むような形で表示されるケースが多くありました。

viewport を指定することで、スマートフォンで Web サイトを閲覧できる最適な幅・文字サイズなどの見栄えを追求できるようになったのです。

色々な項目を設定できますが、まず最低限として上記をきちんと記述することを忘れないでおきましょう。

2、スマホサイトのデバッグ方法

2-1・Google Chrome のデベロッパーツール



■デベロッパーツールの表示
Win : ⋮ Ctr+Shift+I (アイ) Mac : ⋮ Command+option+I (アイ)
表示の上で、上記赤枠のアイコンをクリック！

```
<!DOCTYPE html>
<html lang="ja">
<head></head>
...<body class="home b2c" itemscope itemtype="http://schema.org/WebPage"> == $0
  <div class="slicknav_menu">...</div>
  <div id="container">...
    <!-- #container -->
    <script src="https://arrown-blog.com/wp-...
    <script src="https://arrown-blog.com/wp-content/themes/simplicity2-child/javascripts/print.js" defer></script>
    <script type="text/javascript" src="https://arrown-blog.com/wp-content/themes/simplicity2-child/javascripts/scripts.js"></script>
    <script type="text/javascript" src="https://arrown-blog.com/wp-content/themes/simplicity2-child/javascripts/...
    <script type="text/javascript" src="https://arrown-blog.com/wp-content/themes/simplicity2/css/slicknav.css"...
    <!-- SlickNav用のスタイル -->
    <!-- <link rel="stylesheet" href="https://arrown-blog.com/wp-content/themes/simplicity2/css/slicknav.css" -->
    <!-- SlickNavスクリプトファイル -->
    <script src="https://arrown-blog.com/wp-content/themes/simplicity2-child/javascripts/slicknav.js"></script>
    <ins class="adsbygoogle" data-adsbygoogle="...
    <script id="wappalyzer" src="chrome-extension://gppongmhjkpfnbhagmjfkannfbllang/js/inject.js"></script>
</body>
</html>
```



iPhone 6/7... ▾ 414 x 736 100% ▾ Mobile ▾ Online ▾

機種選択できたり、自分でサイズ決めたり、表示倍率を設定したりできます！！！
(サイトによっては、一度リロードする必要があります)

Responsive

Blackberry PlayBook
Galaxy Note 3

Galaxy S III
Galaxy S4
Galaxy S5
Pixel 2
Pixel 2 XL
iPhone 5/SE
iPhone 6/7/8
✓ iPhone 6/7/8 Plus
iPhone X
iPad
iPad Pro

Word

Edit...

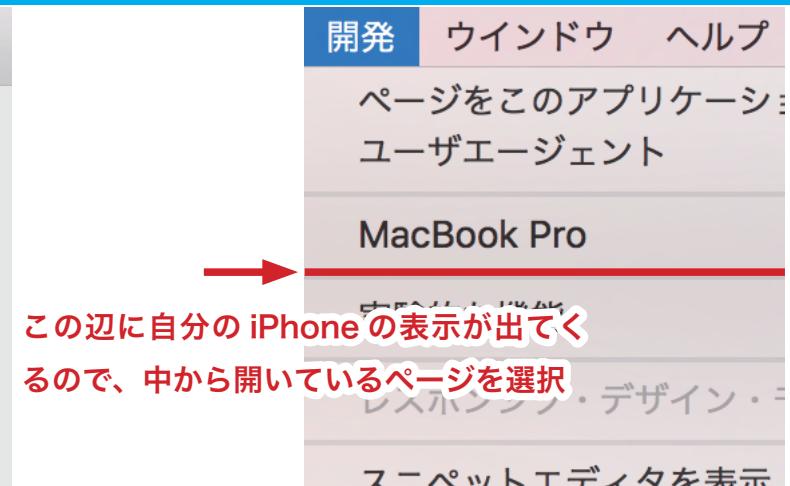
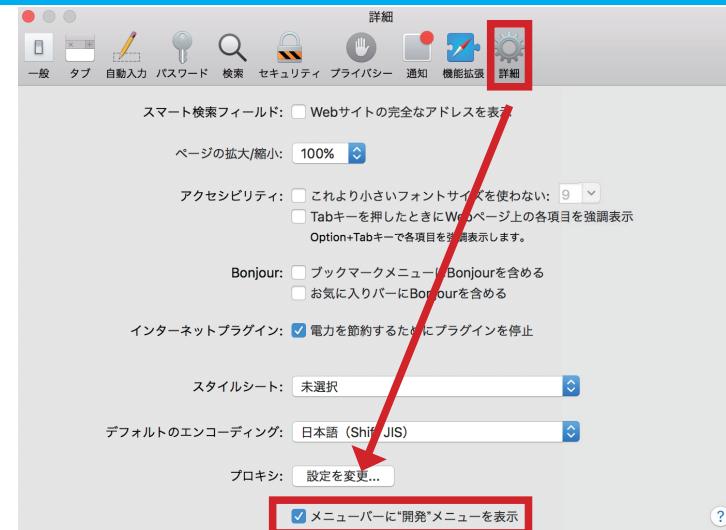
Mac購入検討時におすすめ初心者向け店舗
2018/3/28 Mac/PCスマートフォン 製作ツール
Macのフォント.ttcファイルをttfファイルに変換するのに便利なtransfonter
MacBook Pro 2017 15インチサイズのものを購入しました。MacBook Airの軽さが比較的決まります。
記事を読む

2-2・iPhone × Mac の場合・USB デバック



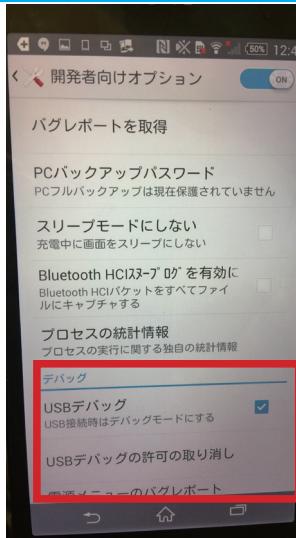
設定方法

1. Mac と iPhone を USB で繋げる
2. iPhone 側の Safari の設定で「Web インスペクタ」をオン。
3. Mac の Safari 側の環境設定→詳細→開発メニューを表示にチェック。



※この方法は、インターネット上に公開されている Web サイトを閲覧・デバッグする時に有効な方法です。

2-3・Android × Chrome の場合・USB デバッグ



設定→開発者オプション→ USB デバッグをオン
(開発者オプションは機種により表示設定方法あり)



設定方法

- 1、Android と PC を USB で繋げる。
- 2、Android 側「設定→開発者向けオプション→ USB デバッグ」をオン。
- 3、Android Chrome で対象ページを開く。
- 4、PC の Chrome で「chrome://inspect/」を開く。
- 5、対象ページが一覧に出てきたら、そのページの inspect をクリック。

※この方法は、インターネット上に公開されている Web サイトを閲覧・デバッグする時に有効な方法です。Windows や Mac などの OS は問いません。

A screenshot of the Chrome DevTools interface under the 'Devices' tab. On the left, there's a sidebar with links like 'Devices', 'Pages', 'Extensions', etc. On the right, there's a main area with sections for 'Discover USB devices' (checkbox checked), 'Discover network targets' (checkbox checked), and 'Open dedicated DevTools for Node'. A large red rectangular box highlights the 'Discover USB devices' section.

上記あたりに接続している Android 機種の Chrome で閲覧しているページが出てくるので、選択するとデベロッパーツールで確認可能

2-4・ローカル開発環境を使って設定

<https://arrown-blog.com/debug-smartphone-web/>

3、width などは固定幅にしない（横スクロール発生させない）

About Me エリアのスマホコーディング

The screenshot shows a mobile browser displaying an 'About Me' section. At the top is a cartoon character icon. Below it is the name 'コスゲタツヤ'. Underneath the name is a bio text block. A red arrow points from the original state to a modified state shown below.

About Me

コスゲタツヤ

埼玉県出身。Webサイトなどの制作業だけでなく、デジタルSTUDIOでのトレーナーや、デジタルハリウッドが運営するエンクール「G'sAcademy（ジーズアカデミー）」でのトレーナーと導師や、映像教材の開発・出演、Schoo WebCampus「Google Apps Script 入門講座」の講師を担当するなど、教育業に深く関わることが多い。最近は個人のブログ「Arrown（アロウン）」を中心としたライティング活動も積極的に行っている。



The screenshot shows the same mobile browser after a modification. The bio text is now wrapped in a

element, demonstrating how responsive design can be implemented. The rest of the page content remains the same.

About Me

コスゲタツヤ

埼玉県出身。Webサイトなどの制作業だけでなく、デジタルハリウッドSTUDIOでのトレーナーや、デジタルハリウッドが運営するエンジニア養成スクール「G'sAcademy（ジーズアカデミー）」でのトレーナーとしての現場指導や、映像教材の開発・出演、Schoo WebCampus「Google Apps Script 入門講座」の講師を担当するなど、教育業に深く関わることが多い。最近は個人のブログ「Arrown（アロウン）」を中心としたライティング活動も積極的に行っている。



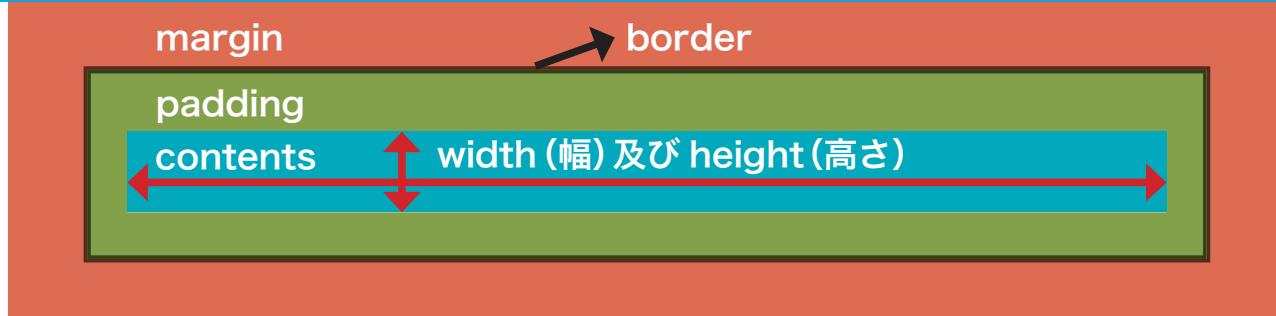
About Me エリア・スマホコーディングのポイント

- width → max-width に変更し width が端末幅を越えないようにする。
- スマホ閲覧時テキストが両端にくっつかないように、左右の padding を 10px 設定し、box-sizing:border-box を設定する。

例) max-width:560px;

- どんなに幅が大きくなっても 560px より大きくはならないですよ。
- 560px 以下の時は width:100% と同じになりますよ。

box-sizing:border-box



HTML のボックスサイズを決める・レイアウトする時には
width+border+padding の合計値で考える。



box-sizing:border-box を指定すると、HTML のボックスサイズを決める・
レイアウトをする時には、width の値だけ考えれば OK !!!
(※イメージとしては、width の中に padding と border が含まれる)



box-sizing: border-box

sanitize.css など、一部のリセット CSS にはあらかじめ全ての要素に
box-sizing:border-box が適用されているため、改めて CSS で **box-**
sizing:border-box を設定する必要がないケースもあります。
(今回はこのケースに該当)

Works & Contact エリアのスマホコーディング

Works

あけましておめでとうございます



オリジナルイラスト年賀状を作成しました！

2017/2/6

Contact

お名前

メールアドレス



Works エリア・スマホコーディングのポイント

- ・横並びの解除→縦並び
- ・スマホ閲覧時テキストが両端にくっつかないように、左右の padding を 10px 設定し、box-sizing:border-box を設定する。

左右の padding 設定は、about me エリアでやったことに習って、設定をしてみましょう！

Works & Contact エリアのスマホコーディング

Works

あけましておめでとうございます



オリジナルイラスト年賀状を作成しました！

2017/2/6

Contact

お名前

メールアドレス



横並びの解除→縦並び

PC 時) `display:flex`

SP 時) `display:block`

スマホ閲覧時に元々横並びだったものを縦並びに変更したい場合、一番シンプルな方法は、**display : block**などを適用することです。

flex-box 使用時に適用した `justify-content` プロパティなどは、`display:flex` が適用されていないと何の効果もないで、`display:block` を適用するだけで事足りてしまいます！

Works & Contact エリアのスマホコーディング

Works

あけましておめでとうございます



オリジナルイラスト年賀状を作成しました！

2017/2/6

Contact

お名前

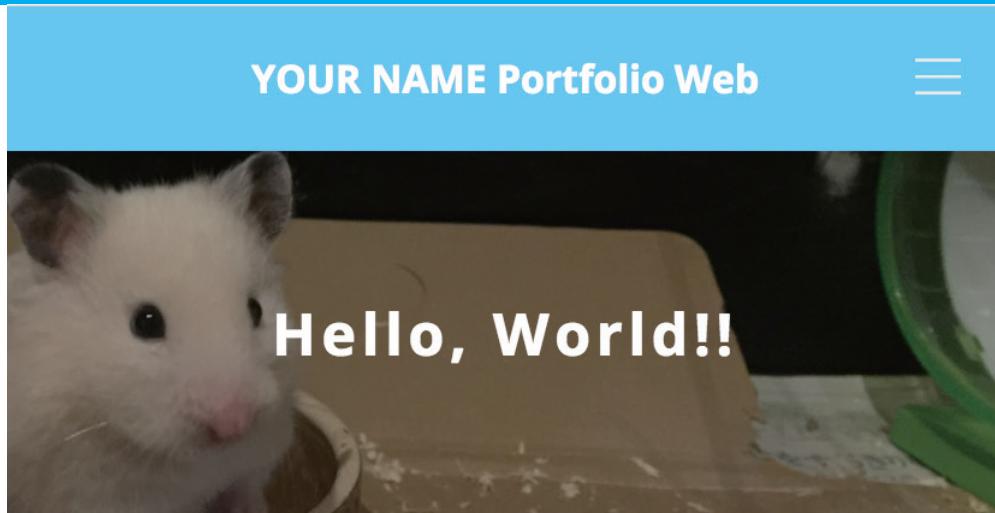
メールアドレス



Works & Contact エリアの仕上げ

あとは、margin や padding を中心とした余白やその他細かい設定を行い、両エリアのスマホコーディングを完成させていきましょう！

hero-image&header エリアのスマホコーディング



hero-image & header エリアのポイント

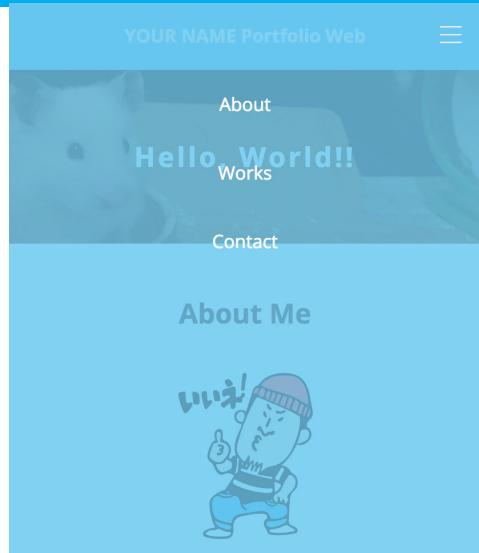
- ・ヘッダー部分の固定表示
- ・メニューボタンの作成 (button タグ) 及び固定表示

ヘッダー部分の固定表示を、position:fixed を使って実現させましょう。

button タグで表示したハンバーガーメニューボタンは、position:fixed あるいは position:relative&position:absolute の組み合わせでレイアウトを実現してみましょう。

なお、一旦 nav タグの部分は display:none で非表示にしておきましょう。

nav エリアのスマホコーディング



nav エリアのポイント

- position:fixed と z-index を使用した全画面表示
- 背景透過 (background-color:rgba 指定を使用)

position:fixed を使用して、nav エリアの全画面表示を実現しましょう。

一旦、nav エリアに設定していた display:none は消去します。

nav エリアのスマホコーディング



width:100%

height:120%

(スクロール時に下が見え
るのを避けるため)

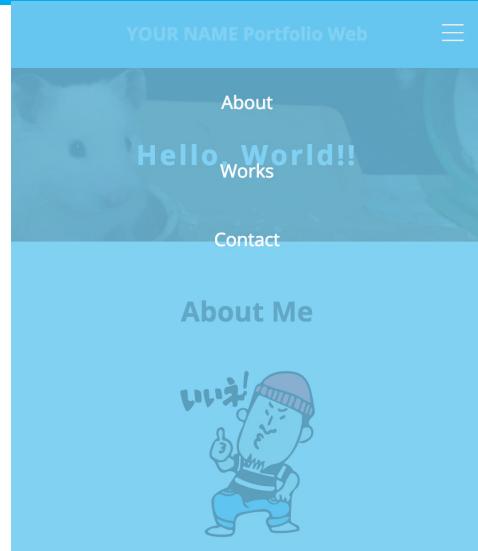


スマホでオーバーレイ表示時に注意したいこと

上記のように全画面を覆うような魅せ方を**オーバーレイ**と呼びます。

スマホでオーバーレイを実現する際は、画面の高さについては、少し多めに
とって height:120%あたりを設定するのがミソです。

z-index



z-index:100;

(-1未満にすると、隠れてしまう)



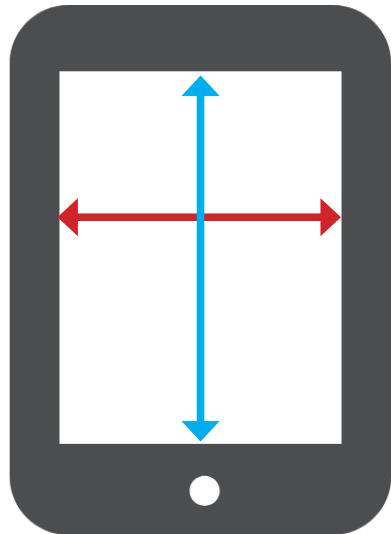
position は基本的に HTML を重ねるものです。

そのため、HTML の重なり順を指定する必要が出てくるケースも多いです。

重なり順は、**z-index** という CSS プロパティで表現することができます。

(数字が大きいほど手前に重なる)

単位：vw と vh



vw

端末 (viewport) の幅に対する割
合で表す単位

vh

端末 (viewport) の高さに対する
割合で表す単位



vw と vh

vw や vh は、Web サイトを閲覧している端末の幅や高さに対する割合でサ
イズを表す単位です。

例) 1440px の幅の PC サイズで見ているときに font-size:56px 相当

$$56 \div 1440 = 0.038888\dots \rightarrow 3.8(3.9)\text{vw}$$

(常に端末の横幅に対して 3.8% の大きさの font-size になる)

上記のように指定できます。