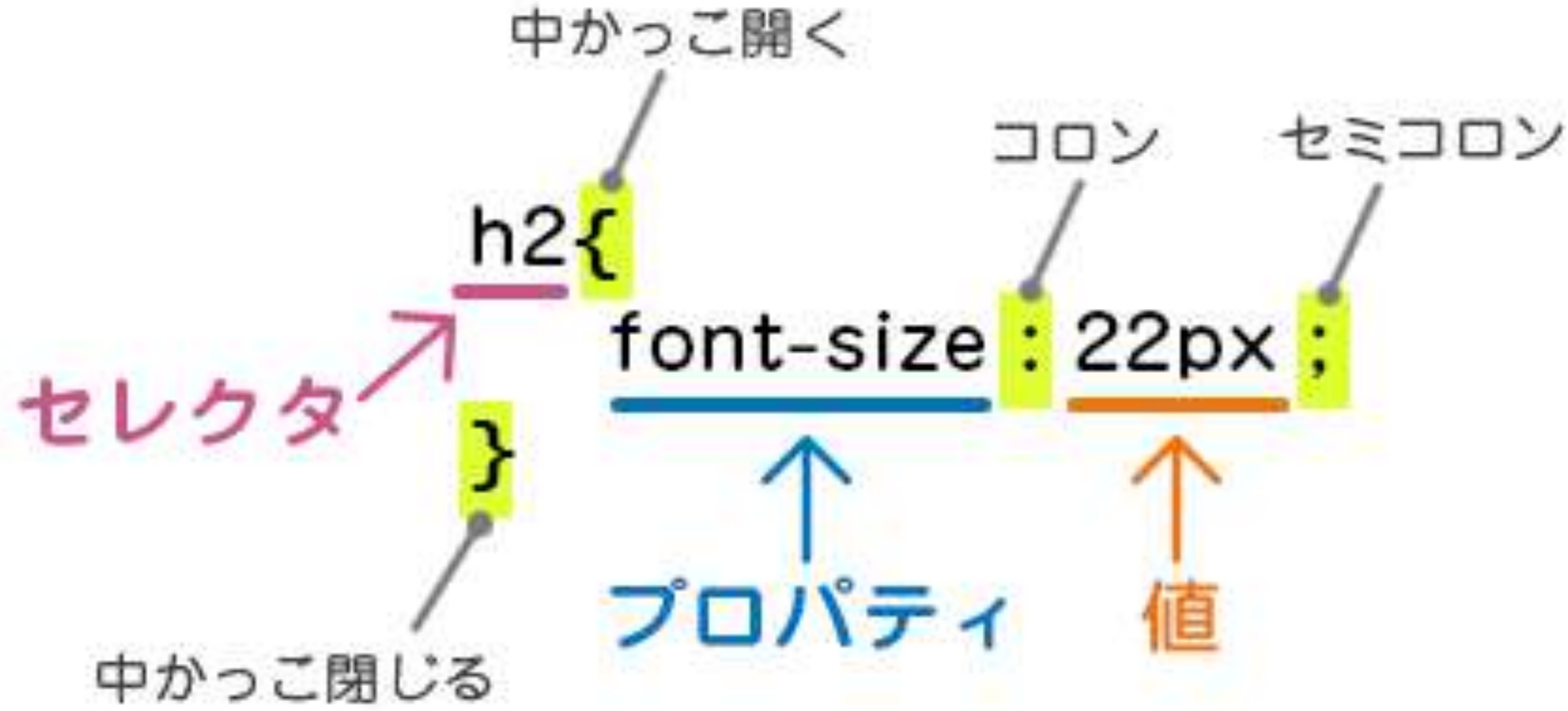
CSS基礎

STEP 2

まずは...

前回のおさらい

CSSの書き方



CSSの書き方

index.html

```
<h2>小田島</h2>
```

style.css

```
h2{
    font-size: 22px;
}
```

h2で囲まれたテキストが

font-sizeが**22**pxになる

CSSの書き方

これは？

CSSの書き方

「 タグ」を、

「 を 」

「 を 」にする

```
p {
    color: blue;
    line-height: 1.5;
}
```

CSSの書き方

「Pタグ」を、
「文字色を青」
「行間を1.5」にする

```
p {
    color: blue;
    line-height: 1.5;
}
```

CSSのプロパティの種類

背景とボーダー・色

- color　文字色
- background　背景色
- background-img　背景画像
- border　線

透過

- opacity　透明度

テキストの指定

- font-family　フォントの種類
- font-size　文字サイズ
- font-weight　文字のウェイト
- line-height　行間
- text-align　左揃え、中央揃え、右揃え

CSSのプロパティの種類

リスト

- list-style-type

  リストの種類

  →点を消したりできる

レイアウトの指定

- width	要素の横幅指定
- height	要素の高さ指定
- margin	要素の外側の余白指定
- padding	要素の内側の余白指定
- display	要素の属性指定
- position	位置を絶対指定
- flex	要素内の要素間隔を指定

idとclass

id

1ページ内で同じ名前を1回しか使用できない
ページ内で唯一意味を持たせたいときに使用

class

1ページ内で同じ名前を何度も使用できる
複数に同じ意味を持たせたいときに使用

idとclass

- class

index.html

```
<h1 class="logo">
    <a href="">Web Designer Academy</a>
</h1>
```

style.css

最初にカンマ！

```
.logo{
    font-weight: 900;
    font-size: 24px;
    text-align: center;
}
```

idとclass

- id

index.html

```
<h1 id="logo">
    <a href="">Web Designer Academy</a>
</h1>
```

style.css

最初にシャープ！

```
#logo{
    font-weight: 900;
    font-size: 24px;
    text-align: center;
}
```

idとclass

命名規則のルール

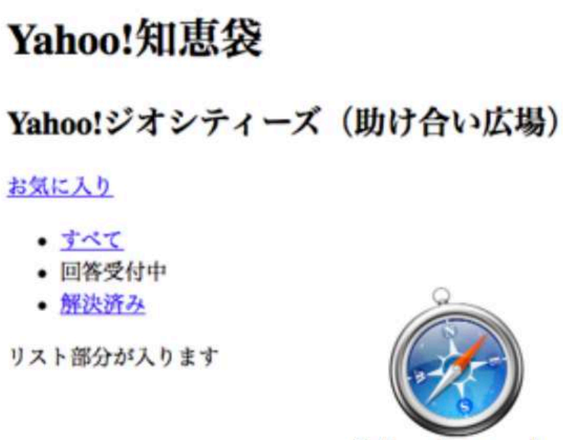
- 半角英数字、アンダースコア(_)、ハイフン(-)を使用します。
- 数字、アンダースコア、ハイフンで始まる名称は使用しません。
- idやclassが使われている場所の情報やその内容を表す名前を付けます。

ブラウザごとの差分問題

Webブラウザ（Chrome、Firefox、IEなど） は、それぞれCSSのタグ毎にデフォルト値を独自に持っているので、例えば何もせずデフォルトのままでHTMLファイルを表示するとブラウザ毎に文字サイズが変わってしまいます

Windows

Windows



Macintosh

ブラウザごとの差分問題

CSSでリセットできる

reset.css
https://webdesign-trends.net/entry/8137

Step 1

1. CSSの書き方
2. プロパティの種類
3. idとclass
4. リセットCSS

Step 1

1. CSSの書き方
2. プロパティの種類
3. idとclass
4. リセットCSS

Step 2

1.セレクタ

2.CSSの優先順位

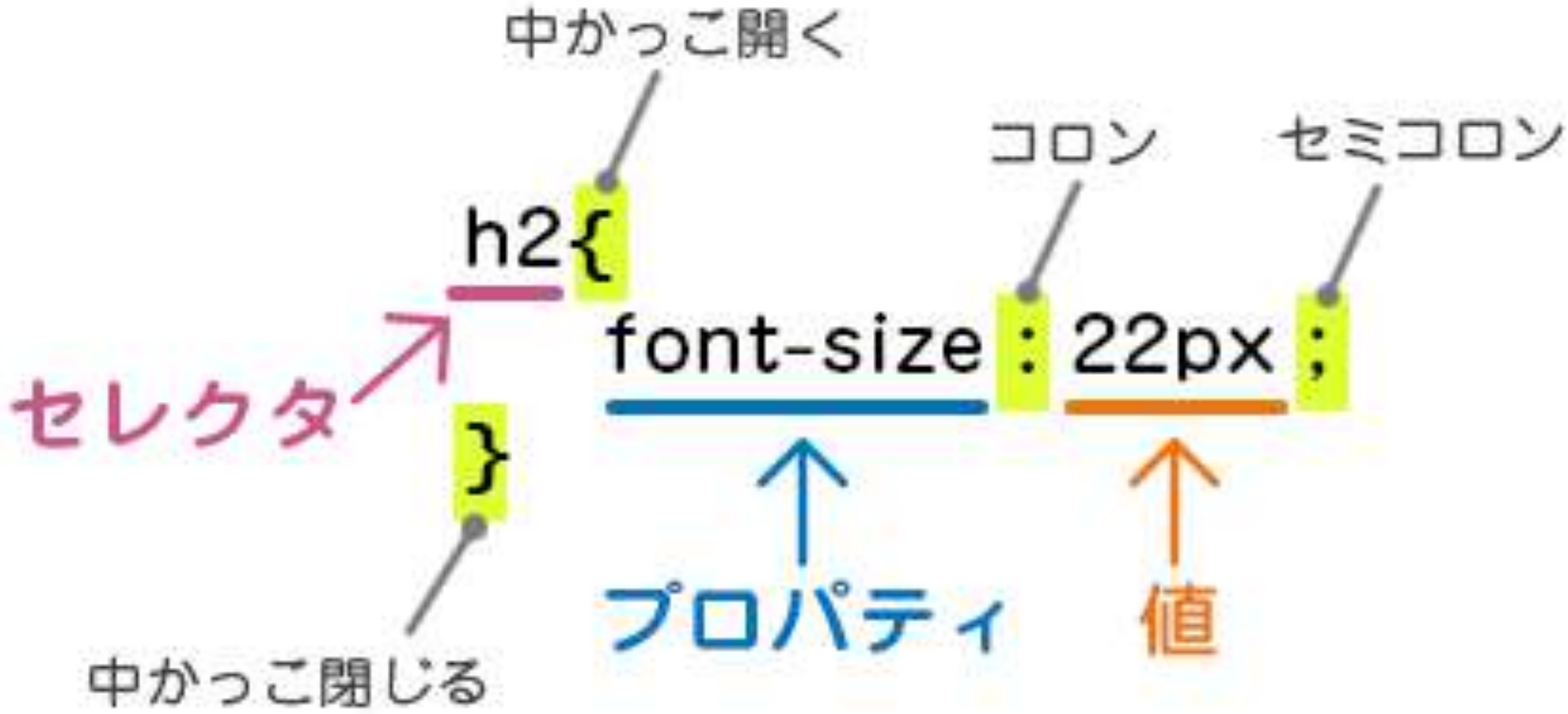
3.ベンダープレフィックス

4.単位

5.BEM

6.Sass

セレクタ



セレクタ

実は...

セレクタの書き方には種類がある

セレクタ

要素型セレクタ	要素名	要素名で指定した要素	p {color:blue;}
全称セレクタ	*	すべての要素	* {color:blue;}
classセレクタ	要素名.class名	クラス名を付けた要素	p.sample {color:blue;}
idセレクタ	要素名#id名	id名を付けた要素	div#sample {color:blue;}
擬似クラス	要素名:link	未訪問のリンク	a:link {color:blue;}
	要素名:visited	訪問済のリンク	a:visited {color:blue;}
	要素名:hover	カーソルが乗っている要素	a:hover {color:blue;}
	要素名:active	クリック中の要素	a:active {color:blue;}
	要素名:first-child	カーソルが乗っている要素	p:first-child {color:blue;}
	要素名:last-child	クリック中の要素	p:last-child {color:blue;}

セレクタ

要素型セレクタ	**要素名**	**要素名で指定した要素**	**p {color:blue;}**
全称セレクタ	*	すべての要素	* {color:blue;}
classセレクタ	要素名.class名	クラス名を付けた要素	p.sample {color:blue;}
idセレクタ	要素名#id名	id名を付けた要素	div#sample {color:blue;}
擬似クラス	**要素名:link**	**未訪問のリンク**	**a:link {color:blue;}**
	要素名:visited	訪問済のリンク	a:visited {color:blue;}
	要素名:hover	**カーソルが乗っている要素**	**a:hover {color:blue;}**
	要素名:active	クリック中の要素	a:active {color:blue;}
	要素名:first-child	**カーソルが乗っている要素**	**p:first-child {color:blue;}**
	要素名:last-child	**クリック中の要素**	**p:last-child {color:blue;}**

セレクタ

擬似要素	要素名:before	要素の直前	li:before {content:"』";}
	要素名:after	要素の直後	li:after {content:"』";}
属性セレクタ	要素名[属性名~= "属性値"]	属性値候補と一致した要素	p[class~="sample"] {color:blue;}
複数のセレクタ	セレクタ,セレクタ	複数のセレクタ	h1, h2 {color:blue;}
子孫セレクタ	セレクタ セレクタ	下の階層の子孫要素	p strong {color:blue;}
子セレクタ	セレクタ>セレクタ	直下の階層の子要素	p > strong {color:blue;}
隣接セレクタ	セレクタ+セレクタ	直後に隣接している要素	h1 + p {color:blue;}

セレクタ

擬似要素	**要素名:before**	**要素の直前**	**li:before {content:"』";}**
	要素名:after	**要素の直後**	**li:after {content:"』";}**
属性セレクタ	要素名[属性名~= "属性値"]	属性値候補と一致した要素	p[class~="sample"] {color:blue;}
複数のセレクタ	セレクタ,セレクタ	複数のセレクタ	h1, h2 {color:blue;}
子孫セレクタ	セレクタ セレクタ	下の階層の子孫要素	p strong {color:blue;}
子セレクタ	**セレクタ>セレクタ**	**直下の階層の子要素**	**p > strong {color:blue;}**
隣接セレクタ	セレクタ+セレクタ	直後に隣接している要素	h1 + p {color:blue;}

Step 1

1. CSSの書き方
2. プロパティの種類
3. idとclass
4. リセットCSS

Step 2

1.セレクタ

2.CSSの優先順位

3.ベンダープレフィックス

4.単位

5.BEM

6.Sass

CSSの優先順位

Point 1

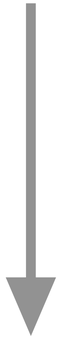
```
p {color:red;}
p {color:yellow;}
p {color:blue;}
```

このCSSの時、
ブラウザの表示はどうなる…？

CSSの優先順位

Point 1

上から下へ、順に読み込む



```
p {color:red;}        /*赤色*/
p {color:yellow;}     /*黄色で上書き*/
p {color:blue;}       /*青色で上書き、この値が有効となる*/
```

CSSの優先順位

Point 2

最終奥義	color: #d9534f !important;
html中のstyle属性	テキスト
ID	#id
class	.class
要素名	pタグとかh1タグとか

CSSの優先順位

Point 2

最終奥義	color: #d9534f !important;
html中のstyle属性	テキスト
ID	#id
class	.class
要素名	pタグとかh1タグとか

CSSの優先順位

Point 2

1. Class
2. 要素名

```
.fugafuga {
  background: blue;
}
p {
  background: red;
}
```

このCSSの時、
ブラウザの表示はどうなる…？

CSSの優先順位

Point 2

1. Class
2. 要素名

```
.fugafuga {
  background: blue;
}
p {
  background: red;
}
```

背景は青色になる

Class > 要素名

CSSの優先順位

Point 2

最終奥義	color: #d9534f !important;
html中のstyle属性	テキスト
ID	#id
class	.class
要素名	pタグとかh1タグとか

CSSの優先順位

Point 2

強い

弱い

最終奥義	color: #d9534f !important;
html中のstyle属性	テキスト
ID	#id
class	.class
要素名	pタグとかh1タグとか

CSSの優先順位

Point 2

1. id
2. Class

```
#fugafuga {
  background: green;
}
.hogehoge {
  background: red;
}
```

このCSSの時、
ブラウザの表示はどうなる…？

CSSの優先順位

Point 2

1. id
2. Class

```
#fugafuga {
  background: green;
}
.hogehoge {
  background: red;
}
```

背景はみどり色になる

Id > Class > 要素名

CSSの優先順位

Point 2

強い

弱い

最終奥義	color: #d9534f !important;
html中のstyle属性	テキスト
ID	#id
class	.class
要素名	pタグとかh1タグとか

CSSの優先順位

うまくCSSが効かない時は確認してみる

Step 1

1. CSSの書き方
2. プロパティの種類
3. idとclass
4. リセットCSS

Step 2

1.セレクタ

2.CSSの優先順位

3.ベンダープレフィックス

4.単位

5.BEM

6.Sass

ベンダープレフィックス

border-radius: 20px;

20pxの角丸をつける

ベンダープレフィックス

border-radius: 20px;

20pxの角丸をつける

Chromeでは表示されるけど、IEでは表示されない…

ベンダープレフィックス

Q, なぜブラウザで違う表示に？

A. 最新のCSSは（CSS3と言う）
全てのブラウザが対応しているわけではない

ベンダープレフィックス

Q, どうすれば良い…？

A. そんなときは

CSSにベンダープレフィックスを記述する

ベンダープレフィックス

接頭辞	対応ブラウザ
-webkit-	Google Chrome、Safari
-moz-	Mozilla Firefox
-ms-	Microsoft Internet Explorer
-o-	Opera

ベンダープレフィックス

```
.sample {
    border-radius: 20px;




}
```

ベンダープレフィックス

```
.sample {
    border-radius: 20px;
    -webkit-borde-radius: 20px; /* Google Chrome、Safari用 */
    -moz-border-radius: 20px;/* Mozilla Firefox用 */
    -ms-border-radius: 20px;  /* Microsoft IE用 */
    -o-border-radius: 20px;  /* Opera用 */
}
```

ベンダープレフィックス

```
.sample {
    border-radius: 20px;
    -webkit-borde-radius: 20px; /* Google Chrome、Safari用 */
    -moz-border-radius: 20px;/* Mozilla Firefox用 */
    -ms-border-radius: 20px;  /* Microsoft IE用 */
    -o-border-radius: 20px;  /* Opera用 */
}
```

ベンダープレフィックス

納品前に複数のブラウザで表示確認しましょう

もしかしたら、別のブラウザでは表示が違うかも...？

Step 1

1. CSSの書き方
2. プロパティの種類
3. idとclass
4. リセットCSS

Step 2

1.セレクタ

2.CSSの優先順位

3.ベンダープレフィックス

4.単位

5.BEM

6.Sass

単位

1. ピクセル単位（px）
2. Em単位（em）
3. Rem単位（rem）
4. パーセント（%）
5. Viewport（vw,vh）

単位

ピクセル単位（px）

font-size: 20px;

絶対単位。ピクセル

単位

Em単位（em）

font-size: 1em; *- 16px*

font-size: 2em; *- 32px*

1emの大きさはブラウザーのデフォルト値で決まります。変更しなければ、1emは16pxです。

親要素のfont-sizeを基準に大きさを計算します。

ここの記述される単位は倍数の値。

単位

emは親要素を継承する

HTML

```
<section class="oya">
    <div class="kodomo">
      子供
      <h1 class="himago">ひ孫</h1>
    </div>
  </section>
```

CSS

```
.oya {
  font-size: 32px;
}
.kodomo {
  font-size: 2em;     64px;
}
.himago {
  font-size: 0.5em;    32px;
}
```

単位

Rem単位（rem）

font-size: 1rem; *- 16px*

font-size: 2rem; *- 32px*

親要素は継承せず、ルート要素となる値を基準に

大きさを計算します。（ルート要素＝HTML）

単位

remはルート要素を継承する

HTML

```
<html>
    <div class="kodomo">
      子供

      <h1 class="himago">ひ孫</h1>

    </div>
</html>
```

CSS

```
html {
  font-size: 32px;
}
.kodomo {
  font-size: 2rem;     64px;
}
.himago {
  font-size: 0.5rem;    16px;
}
```

単位

Em単位（em）

font-size: 1em; *- 16px*

font-size: 2em; *- 32px*

1emの大きさはブラウザーのデフォルト値で決まります。変更しなければ、1emは16pxです。

親要素のfont-sizeを基準に大きさを計算します。

ここの記述される単位は倍数の値。

単位

パーセント（%）

font-size: 100%; *- 16px*

font-size: 200%; *- 32px*

emと同じ

単位

Viewport

width: 100vw; - 画面幅*100%*

height: 50vh; - 画面高さの半分

ビューポートの幅 or 高さに対する割合

100vhで画面幅ぴったし

Step 1

1. CSSの書き方
2. プロパティの種類
3. idとclass
4. リセットCSS

Step 2

1.セレクタ

2.CSSの優先順位

3.ベンダープレフィックス

4.単位

5.BEM

6.Sass

BEM

CSSのclass名の理想の書き方がある

この命名規則は、あくまで推奨。

実際、自由に書いても表示はされます。

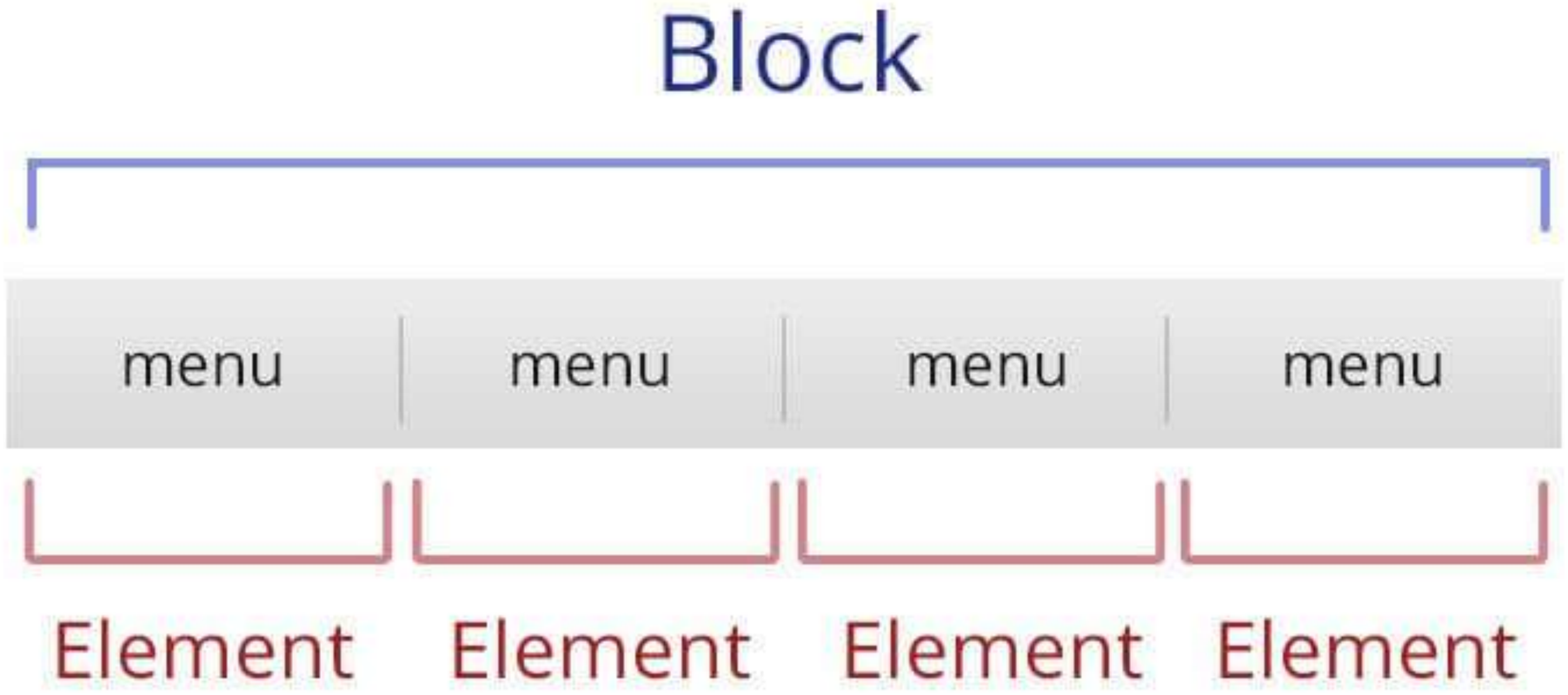
BEM

BEMというCSSの記法

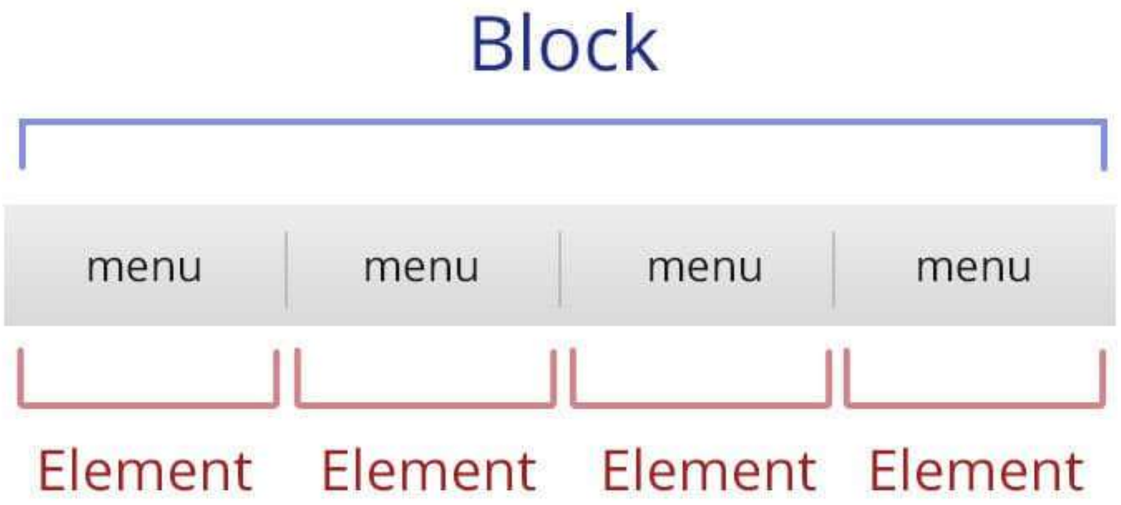
Block, Element, Modifierの

3つからなる記法

BEM



BEM



Block

構成のルートとなる要素。

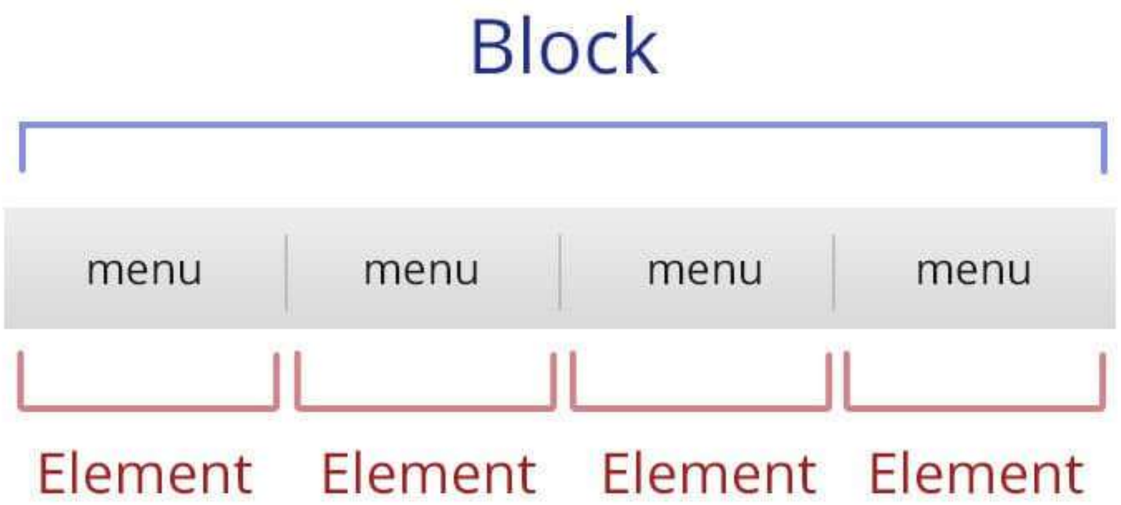
Element

Blockに所属する子要素。必ずBlockの中にいて単体では生きられない。

Modifier

元となるBlockまたはElementから変化した状態を表す要素。「--」でつなぐ。

BEM



.nav

Block

.nav__list

Block Element

.nav__list--active

Block Element Modifier

BEM

なぜ、BEMがいいの？

作業効率の向上

名前が統一されていると、自分や他人が見た時に分かりやすく、修正する時間が減り、作業効率が向上する。

コードの再利用性

名前が統一されていると、同じパーツが分かりやすくなるため、再利用したり、重複するパーツを作る可能性が減る。

Step 1

1. CSSの書き方
2. プロパティの種類
3. idとclass
4. リセットCSS

Step 2

1.セレクタ

2.CSSの優先順位

3.ベンダープレフィックス

4.単位

5.BEM

6.Sass

SASS

Syntactically awesome style sheets

Syntactically　＝　構文的に

Awesome　＝　イケてる

StyleSheet　＝　スタイルシート

SASS

まずは環境構築から

SASS

SCSS

```scss
section {
    height: 100px;
    width: 100px;

    .class-one {
        height: 50px;
        width: 50px;

        .button {
            color: #074e68;
        }
    }
}
```

CSS

```css
section {
    height: 100px;
    width: 100px;
}

section .class-one {
    height: 50px;
    width: 50px;
}

section .class-one .button {
    color: #074e68;
}
```

SASS

すごいポイント

1. **Variables**　数値を保存できる（変数）
2. **Partials**　styleファイルを分割管理できる
3. **Mixin**　CSSをグループ化して使い回せる
4. **Operators**　数値計算できる

SASS

とはいえ...

まずはCSSを書くことに慣れましょう

SASS

続きは次回