px以外の値

今まではmarginなどと言った値にpxを使って来ました。

ただ他にも値があり、実務で頻繁に利用するので覚えておきましょう。

サイズ指定の種類一覧

存在するサイズ指定方法を一覧です。

値	仕様
px	サイズを絶対値指定する
%	サイズを相対値値指定する（親要素基準）
em	サイズを相対値値指定する（親要素基準）
rem	サイズを相対値値指定する（ルート要素基準）
vw	サイズを相対値値指定する（画面幅）

それぞれの詳細な仕様

絶対値とは

絶対値とは、他の要素に左右されない値のことを言います。

なので定義した値がそのまま反映されます。

相対値とは

相対値とは、他要素を参考に数値が決まる値のことを言います。

子要素に指定したサイズが親要素の値が変わるなどの要因で子要素の値を変えなくても影響が出てきます。

px

上記で説明した通り、指定した要素そのままのサイズになるということです。

たとえば以下のように、親要素がfont-sizeを10pxと指定しても、なんの影響も受けません。

```
.px-font-frame {
  font-size: 10px;
  border: 5px solid;
  width:300px;
  height: 200px;
}

.px-font {
  font-size: 20px;
}
```

```
<div class="px-font-frame">
  <div class="px-font">
      font-sizeが「20px」の場合
  </div>
</div>
```

%

%は直近の親要素に依存します。

基準はwidthやmargin、paddingなら親要素の横幅に対する割合、heightなら親要素の高さに対する割合、font-sizeプロパティなら親要素の文字サイズに対する割合となります。

```
.percent-font-frame {
  font-size: 10px;
  border: 5px solid;
  width:400px;
  height: 200px;
}
.percent-font {
  font-size: 300%;
  /* レイアウト崩れが発生するので確認用 */
  /* padding-left: 100%; */
  /* margin-top: 100%; */
}
```

```
<div class="percent-font-frame">
  <div class="percent-font">
      font-sizeが「300%」の場合
  </div>
</div>
```

今回の場合は`.percent-font-frame`（親要素）に`font-size: 10px;`と定義してありますが、子要素で`%`を使用した場合`100% = 10px`となります。

なので大きくしたい場合は100%より大きい数字、小さくしたい場合は100%より小さい数字で調整ができます。

相対値なので`.percent-font-frame`の`font-size`の値が変われば子要素のサイズも変わってきます。

em

emは至ってシンプルで。%と同じ効果です。

`100% = 1em`なので%と同じ値にするのであれば`em`とすれば同じ大きさになります。

要素自体の`font-size`値を基準として相対的な値となります。

`font-size`自体は直近の親要素に依存します。

```
.em-font-frame {
  font-size: 10px;
  border: 5px solid;
  width:400px;
  height: 200px;
}
.em-font {
  /* 親要素のfont-sizeが10pxなので10×3=30px */
  font-size: 3em;
  /* font-sizeが30pxなので30×2=60px */
  padding-left: 2em;
  margin-top: 2em;
}
```

```
<div class="em-font-frame">
  <div class="em-font">
      font-sizeが「3em」の場合
  </div>
</div>
```

rem

remは`root em`の略です。

`root em`はルート要素のことを指していて、そのルート要素はhtmlセレクターを指しています。

なのでemと違って親子関係があってもhtmlセレクターの`font-size`に依存します。

```
html {
  font-size: 20px;
}
.rem-font-frame {
  font-size: 10px;
  border: 5px solid;
  width:400px;
  height: 200px;
}
.rem-font {
  /* .rem-font-frameと親子関係だがhtml要素に依存しているのですべて20px */
  font-size: 1rem;
  padding-left: 1em;
  margin-top: 1em;
}
```

```
<div class="rem-font-frame">
  <div class="rem-font">
      font-sizeが「1rem」の場合
  </div>
</div>
```

vw

vwは「viewport width」の略称で、ユーザーが使用している端末の画面幅を100とします。

たとえば画面幅が1200pxの場合、`1vw`で`12px`となります

```
.vw-font-frame {
  font-size: 10px;
  border: 5px solid;
  width:400px;
  height: 200px;
}
.vw-font {
  font-size: 1vw;
  padding-left: 1vw;
  margin-top: 1vw;
}
```

```
<div class="vw-font-frame">
  <div class="vw-font">
      font-sizeが「1vw」の場合
  </div>
</div>
```

pxによる弊害

デザインカンプなどにはpxで書かれていることが多いですが、レスポンシブもすべてpxで作ってしまうととある弊害が起きます。

たとえばある要素のmarginを1199px～767pxまで50pxだとし、デザインカンプの横幅は1199pxだとします。

そして書いてある通りにpx指定で作成し、確認した時に767pxまでする途中でレイアウト崩れが発生しました。

原因は、デザイナーはPC、TAB、SPでデザインはするがその間のレイアウトは考慮しません。（やっていたら仕事終わらないし、実装する側も大量にブレイクポイント作ることになり仕事終わらない）

なのでそう言った部分は実装側が考慮しなけらばなりません。

pxで作ってしまうと画面サイズが変わっても必ず指定したpxを取ります。

かと言ってレイアウト崩れが発生したところでされにブレイクポイントを作るとデザイナーの指定したものと変わってしまうとの、他の要素もブレイクポイントを作って調整となり無理矢理できたとしても運用保守性がまったくありません。

なので相対的な値を使用してサイズが変わっても相対値で自動的に調整する必要があります。

またデザイナーによってはTAB版のデザインをしないこともあります。

その時はPC版の縮小版かSPの拡大版にするのかを確認して作業する必要があります。pxだと融通が効かないので相対値で対応する必要があります。

補足

Sassを使用したレスポンシブの対応でよくあるのが、htmlタグの中のfont-sizeを画面サイズに応じて調整する実装し、remを使用している部分へ影響が出るようにしています。

```
font-size: calc(100vw / 3.75);
```

この計算方法は絶対この計算方法と言うものはなく、会社、案件ごとに変わります。

なので調べてこんなやり方があると言うのをある程度把握しておいてください。

上記の計算方法は過去にやったものを参考にしています。

課題

それぞれ値を紹介しましたがどう使い分けたら良いのか値ごとにまとめてください。