**html CSS**

【ショートカット】

Ctrl + /　　　　　　　　　　コメントアウト

　　Shift+Alt+F　　　　　 コードの整形

　　Alt+↓　　　　　　　　　　　行を下へ移動

　　Alt+↑　　　　　　　　　　　行を上へ移動

Alt+Shift+↓　　　　　 行を下へコピー

Alt+Shift +↑　　　　　行を上へコピー

Ctrl + Shift + K 　　行を削除

　　Ctrl + D　　　　　　　　　　選択範囲と同じ文字列の複数指定

【自動補完】

　Emmet　VScode内に固定でバンドルされている。

　ミスを減らすために出来るだけ自動補完機能を使用してコードを作成する習慣をつける。

※実例

　div.hoge　 <div class="hoge"></div>

　div#hoge　<div id="hoge"></div>

div\*3　 <div></div><div></div><div></div>

div>h2　 <div><h2></h2></div>

【ページの動作確認】

　VScodeの拡張機能[Live Server]をインストールしておくことで、

　Htmlファイル上で右クリックしたときに、OpenWithLiveServerが表示され、実行結果がブラウザに自動で表示されるようになる。

【Sassファイルの利点】

1. 要素の階層構造を表現することができる。

⇒&が親要素を表す

1. ファイル内に変数を定義することができて、保守性の高いコーディングが可能。

⇒＄で変数を定義する

拡張機能[Live Sass Compiler] Sassファイルを解析してＣＳＳファイルを作成する。

親要素(&) > 子要素 { 子要素のみのスタイル適用 }※孫要素には適用されない

　　　　　　　& 子要素　アンパサンドと子要素の間に半角スペースを空ける場合はすべての子要素

　　　　　　　&子要素　アンパサンドと子要素の間に半角スペースを空けない場合は親要素と子要素すべてを満たす要素を所持したセレクタにスタイルが適用される。

【インライン要素とブロック要素】

span インライン要素

div ブロック要素

要素に対するスタイルは適用できる、

それぞれの要素が存在するため注意

displayスタイルにより、

タグの要素種類が変更できる。

（インライン/ブロック要素の切替）

【Styleの適用】

１．<head>内に以下のタグを記載することで外部ファイル（CSS）を読み込み

　　<link rel=”stylesheet” href=”style.css”>

　　CSS

　　セレクター { プロパティ：値; }

・Classに対しての適用　　　　　.(Class名) { プロパティ：値; }

・Idに対しての適用　　　　　　　　#(Id名) { プロパティ：値; }

　　詳細度

　　　同じ要素に対して、複数スタイルが定義されていた場合、適用される優先度は詳細度という概念により決定する。

Class内セレクタ > Id > Class > セレクタ

デバッグとして、Cromeの開発者ツールを使用して、該当要素にどのスタイルが適用されているか確認する方法がある。（stylesタグを確認する）

【要素の修飾】

**背景色**　　　　Background-color

**文字色**　　　　color

**枠線**　　　　　　border

**要素内余白**　padding

**要素外余白**　margin (0 auto) ブロック要素の中央揃え※widthが指定されている要

**文字太さ**　　　font-weight

**マウスカーソル種類**　cursor

**影をつける**　box-shadow

**角を丸める**　border-radias

**位置を移動する**　transform

　※アニメーションを利用する場合は、top leftではなくこの関数を使用する。

　⇒translate(Ｘ軸移動距離 , Y軸移動距離)

　⇒rotate(回転する度数)　transform-origin(Ｘ軸 ,Y軸)支点の位置

　⇒scale(X軸何倍?, Y軸何倍?)

**位置を指定する**　position

　デフォルトでは全要素にstatic；でプロパティ値が指定されている。

　親要素にrelative, 子要素にabsolute top leftで親要素からの相対位置で位置指定をする。absolute指定をした場合、親要素は子要素の高さと幅を含まなくなるため注意。

　他の子要素にrelativeを指定することで解決する。

　Fixedを指定することで、画面にスクロールが走ったときにも固定表示されるようになる。

　ｓtickyを指定することで、親要素に固定表示されるようになる。（Fixedが画面全体、stickyは親要素）

　⇒親要素にoverflow: hidden;が指定されていると、画面の固定表示が付いてこなくなる。

**要素の重なり**　　　z-index

　Positionで要素を指定した時に、子要素が重なることがある。

　先頭に表示するには、z-indexに大きな値を格納することが重要となる。

**枠外の値は非表示**　　　overflow: hidden;　※表示はvisible

【スタックコンテキスト】

　Positionがstatic以外 かつ z-indexに数値指定がある。

　スタックコンテキストが形成されていると呼称する。

　・ルートスタックコンテキスト

　　上記の条件を満たす、ローカルスタックコンテキスト内に存在しないスタックコンテキスト。

　・ローカルスタックコンテキスト

　　他のローカルスタックコンテキストに含まれている要素で、position、z-indexを指定したとしても、ローカルスタックコンテキスト内で入れ子のスタックコンテキストを形成することとなる。

　　※z-indexを記載しない or プロパティ値にautoを指定すると

【擬似要素】

　::before context=””　を指定することで、からのspan要素が追加されているとみなして動作を行うことができる。1要素につき、::before ,::after指定は1つまでとなる。優先度はより詳細度が高い方となる。

（補足）スタイルの適用のためだけに空のspan要素を追加することは要素の無駄遣いとなるため、このような記法が用いられる。

【トランジション】

　Transition を　定義することでアニメーションを指定することができる。

　アニメーションの変化の移り変わりの時間を指定　transition 要素　変化秒数　遅延秒数

　　➡第三引数にeaseを指定して、画面の検証モードからアニメーションの検証を行うことで、

　　　　Cubic-bezier()の指定を行うことができる。横軸が変化速度、縦軸は変化量

　マウスカーソルが重なったとき　hover

【位置の調整】

　（横方向の調整）

　inline要素は基本文字の要素を表す。

　<div>子要素にdisplay:inline-block;を適用する。

　親要素にtext-align: ~ ; 文字配置を指定

　　➡子要素間にスペースが空いてしまうが、letter-spacing: -6px;と定義することでスペースがなくなる。

　Block要素 or inline-block要素に対して、margin: 0 auto;

　　で左右に均等な余白を作ることができる。　※widthが指定されていることが前提。

　（縦方向の調整）

１．Position: relative と absoluteを使う方法

　　　　要素の上に要素が重なるようなレイアウトの作り方を行う場合に利用する方法

1. Display:flex;を利用する方法

要素が横並びになる。

Flex-direction: column;

要素が縦並びになる。

Justify-content: strat, center , end ; 左　真ん中　右

align-items: flex- strat, center , end ; 上　真ん中　下

※Flex-direction: column;を指定していると、逆転する。

　子要素に対して、flex-grow: 比数; 親要素に対して、指定の比率で子要素を並べる。

　それぞれの子要素ごとに比率を指定することで、思い通りの比率で子要素を並べることができる。

大体はflexでレイアウトを作れるので、調べてみて。

【表現】

　●を生成

　　  width: 18px;

    height: 18px;

    background-color: black;

    border-radius: 50%;

【アニメーション】

　@keyframes キーフレーム名 { アニメーションの内容 }

　Animation: キーフレーム名 変化時間 継続タイプ;

(例)animation: sk-bouncedelay 1.4s infinite;

　アニメーション開始時間を変更する

　　Animation-delay: (-は早く +は遅く);

　〇点滅

　@keyframes キーフレーム名 {

0% {opacity: 0;}

100% { opacity: 1;}

@keyframes sk-bouncedelay {

  0% { opacity: 0; }

  100% { opacity: 1; }

}

（ローダーを作ってみよう）

〇収縮↔拡大

　@keyframes キーフレーム名 {

0% {scale: 0;}

100% { scale: 1;}

  & div{

    display: inline-block;

    width: 18px;

    height: 18px;

    background-color: $cBlack;

    border-radius: 50%;

    animation: sk-bouncedelay 1.4s infinite;

    &.bounce1 {

      animation-delay: -0.32s;

    }

    &.bounce2 {

      animation-delay: -0.16s;

    }

  }

}

@keyframes sk-bouncedelay {

  0% { transform: scale(0); }

  40% { transform: scale(1); }

  80% { transform: scale(0); }

  100% { transform: scale(0); }

}

ローダーのテクニック

　アニメーションの開始(animation-delay)は0より小さくしておく

　　⇒画面が描画されたタイミングで最初の要素が変化し始めるため見え方が自然

　アニメーションの動作は大体0.1s刻みくらいにしておくと滑らかな動作となる。

様々なプロパティ値

  animation: sk-bouncedelay 1.4s;

  animation-timing-function: ease;

  animation-delay: 3s;

animation-iteration-count: 2;

animation-direction: reverse;

animation-fill-mode: forwards;

animation-iteration-count　何度アニメーションの変化を行うか指定できる

　　infiniteは無限に繰り返す　数値では指定回数

animation-direction　　　　　　　変化の方向を指定できる

　　reverse 100%　⇒　0%

alternate 0%　⇒　100%　と　100%　⇒　0%を交互に繰り返す

animation-fill-mode　　　　　　　アニメーション終了時点の状態を適用

animation-play-state: paused;　アニメーションを停止することができる。

　※基本的にJSと一緒に使用する　イベントを受け取ったらこのプロパティが適用される実装

【アニメーション設定のプロパティ値を管理】

@mixinを使用することで、プロパティ値を変数のように変化させることが可能。

また複数プロパティを指定することも可能であり、一部プロパティはデフォルト値を指定する方法がよく使用される。

変数により設定　　　　　　 name

デフォルト値により設定　duration

@mixin animation (

  $name ,

  $duration: 1s

) {

  animation-name: $name;

  animation-duration: $duration;

}

@include animation(

  $name : sk-bouncedelay

);

【プロパティの省略記法】

Animation-delay

Animation-name

のような「-」以降に複数プロパティを持つプロパティはまとめて記載することができる。

　Animation: {

Delay: 1s;

name: sk-bouncedelay;

timing-function: ~

}

【擬似セレクター】

同じ要素についても複数の状態をもっておりそれぞれの要素についてプロパティ値の指定をすることができる。

　:visited リンクボタン押下後の状態

　:hover 要素に対してカーソルを重ねているとき

:focus 最後にフォーカスが当たっていた要素

:active 要素をクリックしている状態

　:nth-of-type(n)　n番目の子要素に対してスタイルを適用する

　　⇒2nで偶数　、　2n+1で奇数を指定することができる。

:first-of-type 最初の子要素に対してスタイルを適用する

:last-of-type 最後の子要素に対してスタイルを適用する

　:not(セレクター) 指定セレクター以外のものについてスタイルを適用する。

【sass記法での繰り返し】

From 開始値 through 終了値

変数をセレクタの引数として指定する場合は、#{}で囲む必要がある。

　変数を用いて四則演算をする場合には、数値の単位を含めることが可能。

    @for $i from 1 through 2 {

      &:nth-of-type(#{$i}) {

        animation-delay: -0.32s / $i;

      }

    }

【ハンバーガーメニュー】

Onclickで.menu-openクラスのスタイルを呼び出すようにしている。

初めにtransformでアニメーション後の状態を確認してから、transitionに変化時間を与えてアニメーションを作りこんでいく方法が良い。

また、transformにdelayを設けて、それぞれの要素が別のタイミングで動くようにすると、アニメーションがより複雑に見える。

ボタンであれば、cursorはpointerプロパティに設定した方がわかりやすい。

.mobile-menu-icon {

  background-color: transparent;

  border: none;

  cursor: pointer;

  & > span {

    background-color: black;

    width: 35px;

    height: 2px;

    display: block;

    margin-bottom: 9px;

    transition: transform 0.7s;

    &:last-of-type{

      margin-bottom: 0;

    }

  }

}

.menu-open {

  background-color: black;

  & .mobile-menu-icon {

    & span {

      background-color: white;

      &:nth-of-type(1){

        transform: translateY(11px) rotate(135deg);

      }

      &:nth-of-type(2){

        transform: translateX(-18px) scaleX(0);

      }

      &:nth-of-type(3){

        transform: translateY(-11px) rotate(-135deg);

      }

    }

  }

}

【文字アニメーションの作成】

\_mixin.scss　先頭「\_」で始まるscssファイルは共通部品としてみなされるため、

コンパイラでcssファイルが生成されることはない。

呼出については、@import ‘mixin’;のように「\_」を除去したファイル名で参照する。

@import 'mixin';

@include animation {

}

Body { margin:0;} 初期値でbodyタグの外側に余白がある。

Height: 100vh; 画面幅いっぱいまで要素を指定する。

translateスタイルはinline要素に対してのみ指定が可能となる。

フォントの指定

https://fonts.google.com/

使いたいフォントを選んで、headタグの中にlinkをコピペする。

適用したい要素に対して、fontスタイルをコピペする。

【マップ】

Vbでいうディクショナリのようなもの。キーと値情報を保持する。

マップ記法で変数を宣言して、each文でループして使用する。

$pattern: (

  "up": translateY(6px),

  "down": translateY(-6px),

  "left": translateX(40px),

  "right": translateX(-40px)

);

@each $key, $value in $pattern {

  .appear.#{$key} {

    & .item {

      transform: $value;

    }

  }

}

【今風の画像の表示方法について】

Cssで画像を表示する方法。

divタグに対して、要素の高さを定義する必要がある。

デフォルトでは画像を繰り返すスタイルとなっているため、background-repeat:にno-repeatを適用する。

  <div class="bg-img"></div>

  <style>

    .bg-img {

      height: 500px;

      background-image: url(./images/400x250.png);

      background-repeat: no-repeat;

      background-size: cover;

      background-position: center;

    }

  </style>

Background-size: cover;はアスペクト比を固定したまま、表示領域に合わせて画像をトリムしてくれるためレイアウトが崩れにくい。imgタグではobject-fitスタイル

@media　(min-width:800px;)複数解像度の画像を容易。

Min-width指定以上で画面が表示された場合は指定のスタイルを適用する。

Background-imageスタイルでそれぞれのサイズに合った画像を適用する。

【その他テクニック】

　●画像が大きい場合に、親要素の横幅に合わせて画像を表示する。

img {

  max-width: 100%;

}

●疑似要素、::before ::afterを指定する場合は、

contentに’’を必ず指定する。擬似Spanだから。。

  &::after {

    content: '';

  }

●画像とその他ブロック要素の高さをそろえる。

デフォルトで空白が入ってしまうため。それを解消する。

  img {

    max-width: 100%;

vertical-align: bottom;

  }

●処理単位ごとにクラスを定義することが重要

●一度クラスに定義したスタイルを使いまわす場合

　　@extend class名を使用してそのクラスに定義したスタイルを継承する。

【レンダリングエンジンについて】

　Top, left より transformを利用したアニメーション

　Rgbaよりopacityを利用したアニメーションの方がリソース効率が良い。

　※Translate transform-origin(アニメーションを開始する支点)を利用して同様の実装をする。