レスポンシブ

今回はレスポンシブについて学びます。

ただガッツリ課題をやると言うより、概要を知ってもらう課題です。

レスポンシブを踏まえたコーディングは別の課題で出します。

レスポンシブもWeb制作で重要なので研修を通してしっかり身に付けてください。

レスポンシブとは

正式にはレスポンシブデザインと言います。

スマートフォンやタブレットの普及により今までPCでしか閲覧できなかったWebサイトがそれらのデバイスでも閲覧できるようになりました。

ただ、PC用に作ったものをそのままスマートフォンなどに表示するとWebページ全体を表示させるか、ものすごい拡大して表示されるなどが起こりユーザーにとって利用し難いWebサイトとなってしまいます。

なのでスマートフォンなどでも使いやすいように画面サイズに応じてWebサイトのデザインを切り替える必要があります。

画面サイズが認識される仕組み

Webサイトなどを利用するにあたって、ユーザーエージェントと言うものが裏で動いています。

Webサイトにアクセスした際、Webサイト側にこちらの情報を送っています。（使用している端末、OS、ブラウザなど）

主には送られてきた端末の情報を元に、それに応じたWebサイトを表示しています。

レスポンシブの対応方法

方法は2つあります。

1. 1ページに対して1つのHTMLを用意して画面幅に応じてメディアクエリで表示を切り替える（メディアクエリについては後程説明します）

＊ メリット
サイト自体はひとつのため、管理や変更の手間は少ない

PCサイトとスマホサイトのURLが同じ（URLをシェアしやすい）

＊ デメリット
余分なファイルの読み込み等があり、負荷がかかる

デザインの制約が生まれる

2. 1つのページに対して2つHTMLを用意して画面幅に応じて表示するHTMLを切り替える。

```
* メリット
  別のサイトを用意する為スマホ用に特化できる

  PCサイトも閲覧可能

* デメリット
  サイトが2つになるため、管理や変更の手間が増える

  サーバーなどの構成によってはPCサイトとスマホサイトのURLが異なる
```

ユーザーエージェントを使用したレスポンシブの対応は色々と設定をする必要があるので研修では説明のみとします。

メディアクエリの使用方法

メディアクエリどうやって使うの？と思いますが、CSSで使用します。

CSSファイル内で下記のように書きます。

```
@media screen and (max-width: 1100px) {}
```

`(max-width: 1100px)`画面幅の指定を行なっています。複数指定することも可能で上から順に読み込まれる。

`max-width`が画面幅の最大値、`min-width`が画面幅の最小値となっています。

例

```
/* 画面幅が最大1100pxの時に下記のメディアクエリが読み込まれる */
@media screen and (max-width: 1100px) {}

/* 画面幅が最小値768pxから最大値1000pxの時に下記のメディアクエリが読み込まれる */
@media screen and (min-width:768px) and ( max-width:1000px) {}

/* 画面幅が最大値767px以下の時に下記のメディアクエリが読み込まれる */
@media screen and (max-width: 767px) {}
```

`{}`の中に、その画面幅になった時のCSSを書いて行きます。

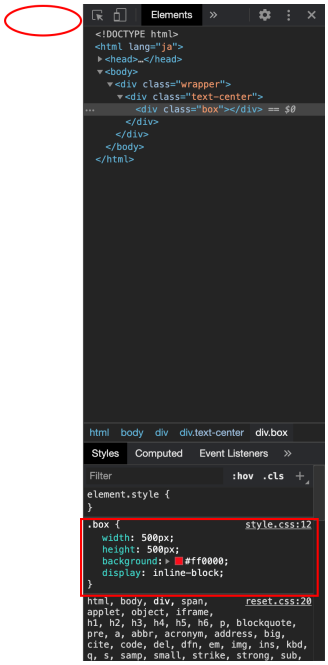
例

- HTML

```
<!-- 赤い色の箱を作っている要素 -->
<div class="box"></div>
```

- CSS

```
/* 赤い色の箱を作っているCSS */
.box {
  width: 500px;
  height: 500px;
  background: #ff0000;
  display: inline-block;
}
```



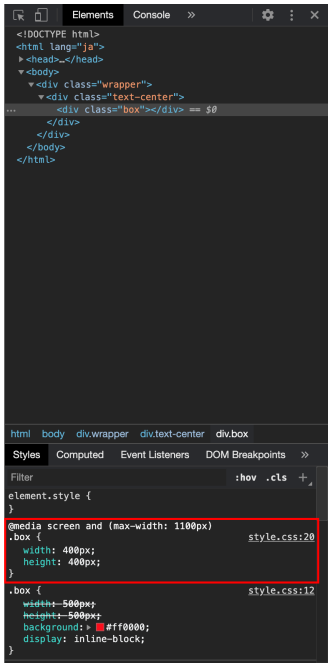
スクショの関係で写っていないのですが、HTMLを表示しているところとディベロッパーツールの境目にマウスカーソルを持っていくと両端に矢印のついたアイコンが表示されるのでそれで画面幅を調整できます。

その時に画像の赤丸で囲ってある部分に現在の画面幅が表示されます。

画面幅が最大1100pxの場合

- CSS

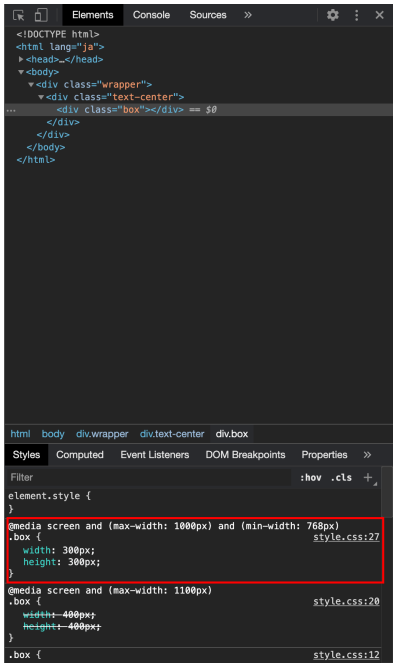
```
@media screen and (max-width: 1100px) {
  .box {
    width: 400px;
    height: 400px;
  }
}
```



画面幅が最小値768pxから最大値1000pxの場合

- CSS

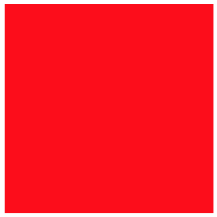
```
@media screen and (min-width:768px) and ( max-width:1000px) {
  .box {
    width: 300px;
    height: 300px;
 }
}
```

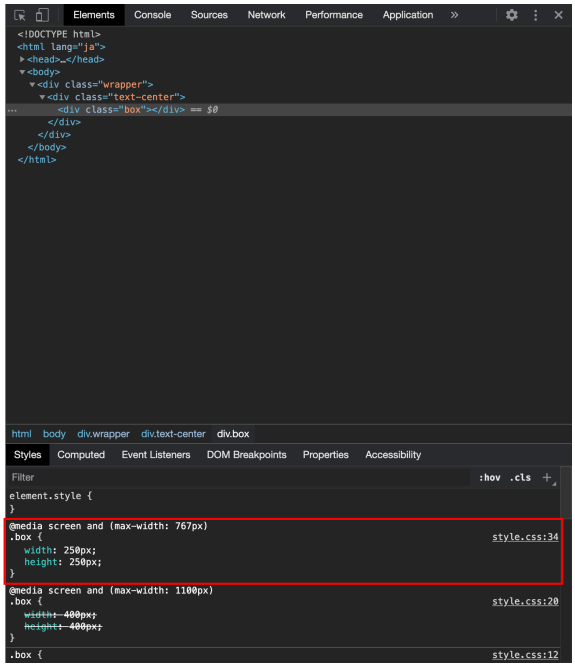


画面幅が最大値767pxの場合

- CSS

```
@media screen and (max-width: 767px) {
  .box {
    width: 250px;
    height: 250px;
  }
}
```





このようにして画面幅が変わるとメディアクエリ内に書いたCSSが優先して適用されます。

メディアクエリの外で書いたCSSのプロパティと中で書いたCSSのプロパティに差異がありますが、メディアクエリ内で書かれなかったCSSのプロパティはメディアクエリが適用されていても適用されます。

なので画面サイズが変わった時に適用したいプロパティのみをメディアクエリ内に書きましょう。

どの画面幅で切り替えるのかを**ブレイクポイント**と呼びます。

今回解説の為にコードを分けていますが、同じファイルの中で書いてます。

作業環境や、案件にもよりますが、同じファイル内にレスポンシブ対応を行う場合と、別ファイルに分けて対応する場合があります。

課題

問題を7問出すので指定通り作成してください。

作成したものはすべてて中央に寄せてください。（一部場外有）

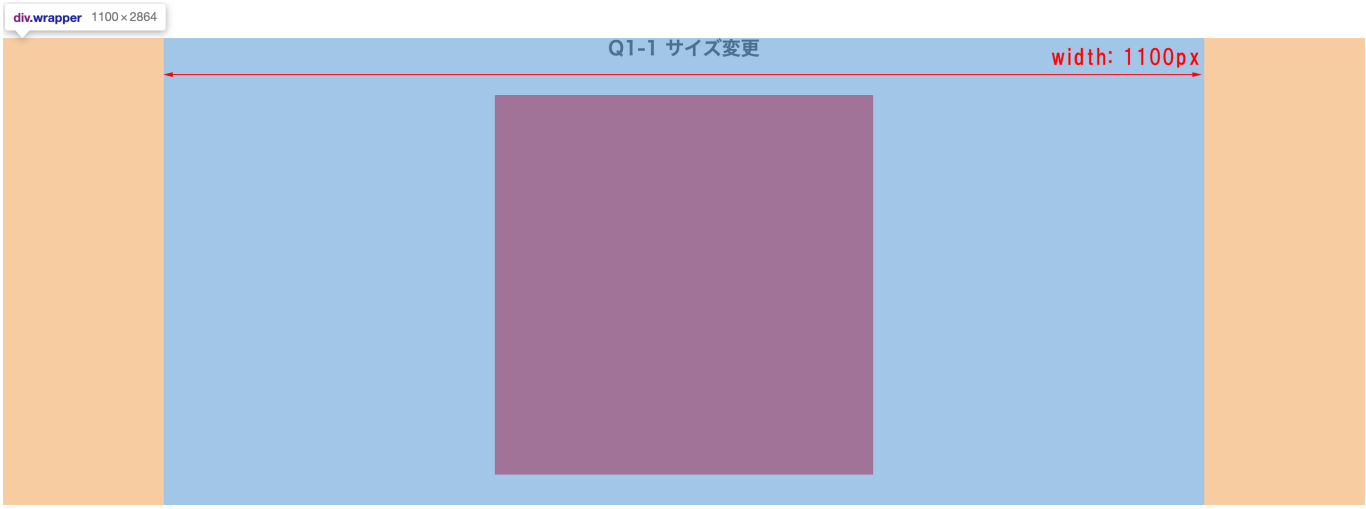
中央に寄せる方法はお任せします。ただ、あまりにも無茶な方法を取っていた場合は修正をしていただきます。

今回もベースとなるものはGitHubに上げてますのでそちらからクローンお願いします。

URLは下記からアクセスしてください。

GitHub

HTMLファイルを開くと問題ごとのタイトルがありますがそちらも中央になるよう配置してください。



あらかじめクラス名を付けて、base.cssにスタイルを書いてます。

そちらには手を加える必要はありません。どうしてもCSSを書く必要がある場合、クラスを新たに追加してください。

その際は、base.cssには書かないでください。

全体的なイメージがつきやすいようにimgフォルダーの中に画像を入れていますのでご確認お願いします。

ブレイクポイントですが、960px未満になったらメディアクエリ内のスタイルが適用されるようにしてください。

問題の説明でPCとSPで分けます。

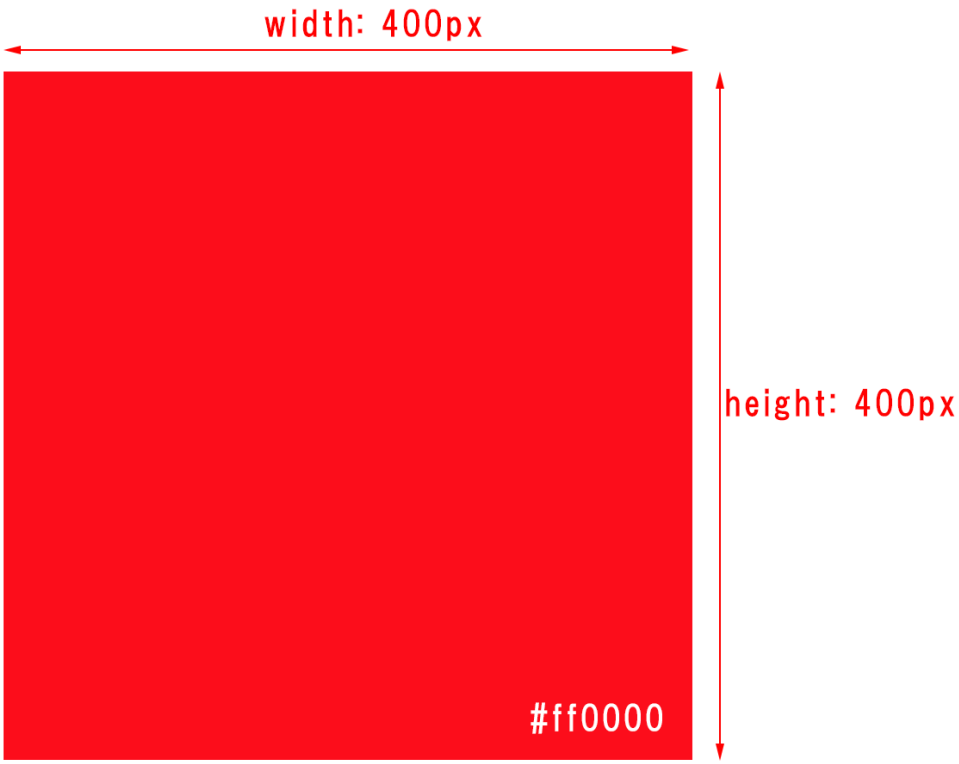
PCがメディアクエリ適用前、SPがメディアクエリ適用後の意味合いで説明します。

Q3以降箱の高さ、幅の値が書かれていませんが、Q2で書かれているサイズと同じサイズにしてください。

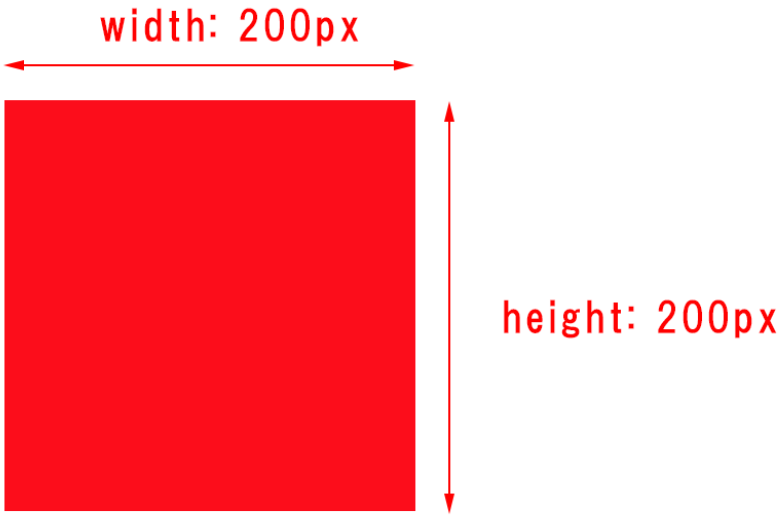
Q1

ブレイクポイントで箱のサイズの変更をお願いします。

- PC



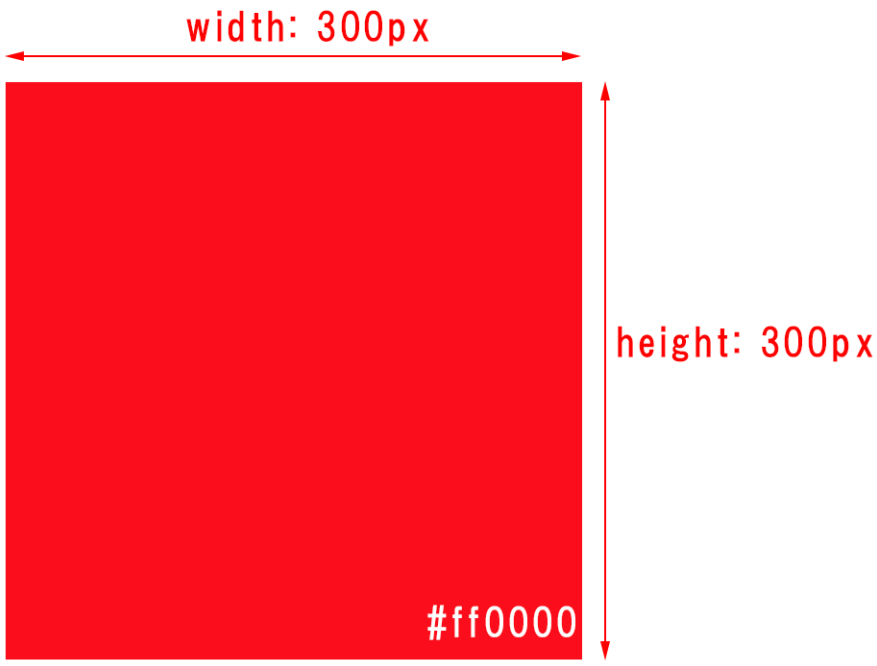
- SP



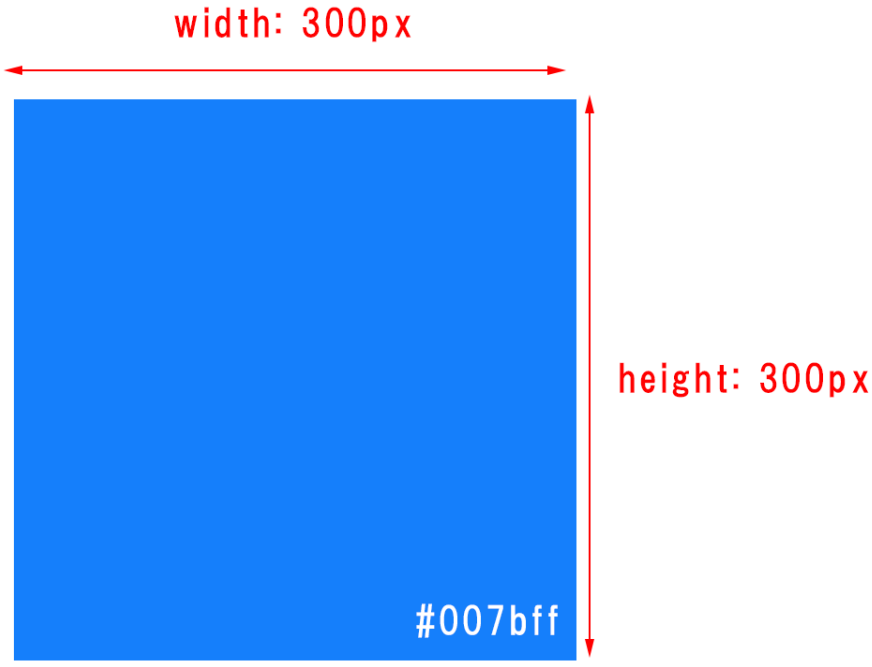
Q2

ブレイクポイントで箱の背景色の変更をしてください。

- PC



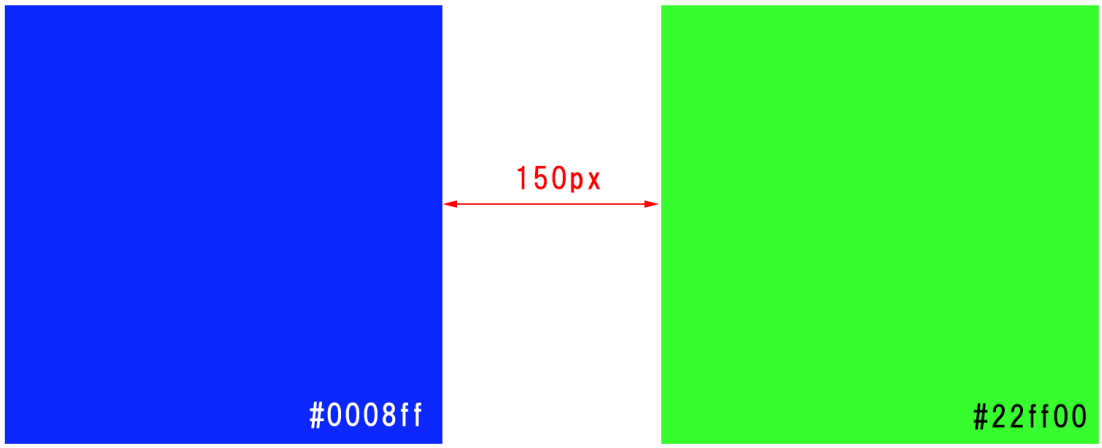
- SP



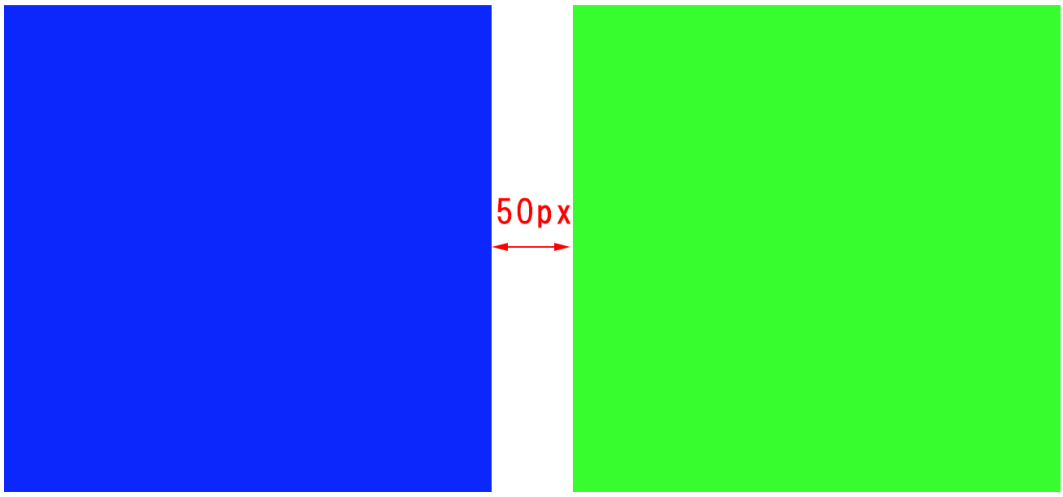
Q3

ブレイクポイントでmargin値の変更をしてください。

- PC



- SP

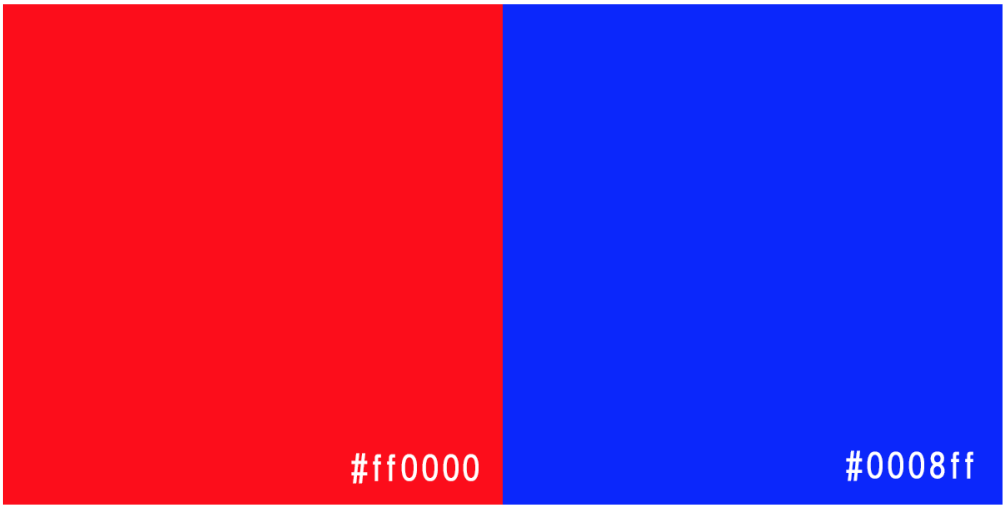


Q4

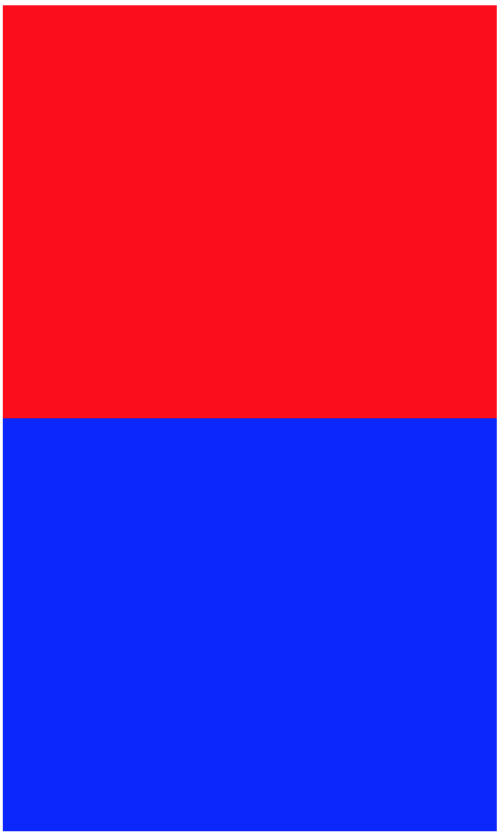
ブレイクポイントでfloatの解除を行なってください。

解除後に要素を中央に寄せる必要はありません。

- PC



- SP



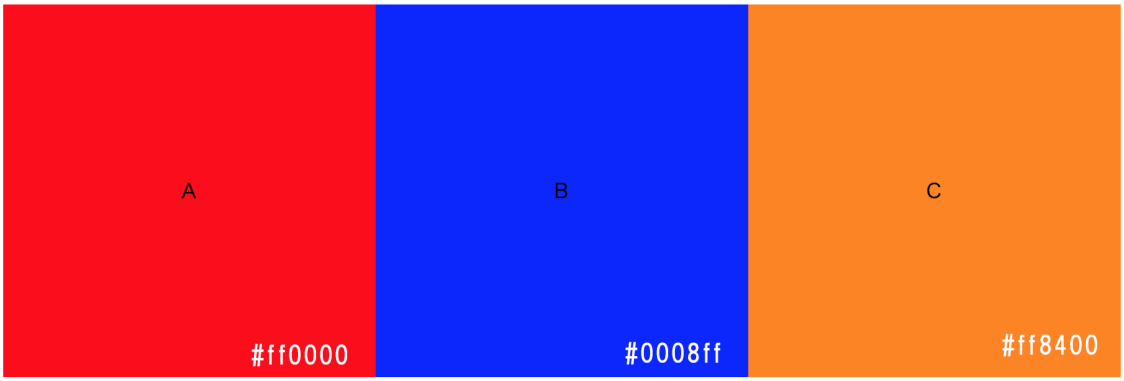
Q5

ブレイクポイントで要素の並びを逆順にしてください。

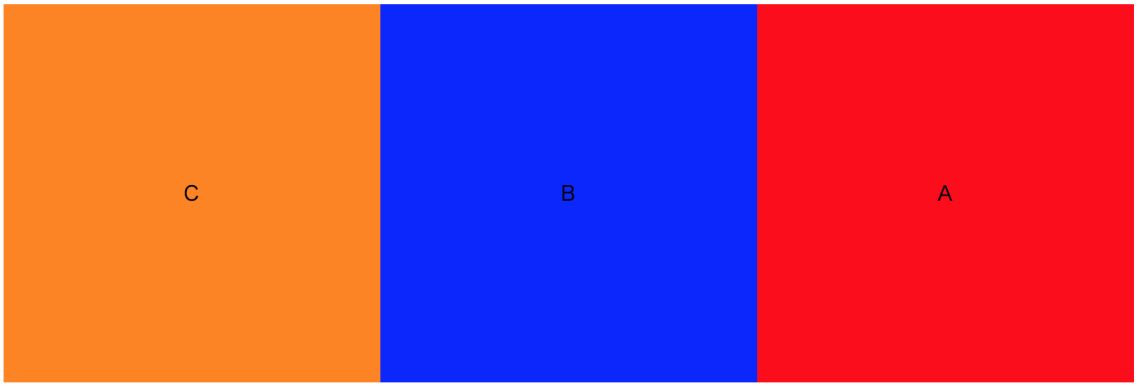
要素の横並びはflexboxを使用してください。

文字も中央で表示するようにしてください。

- PC



- SP



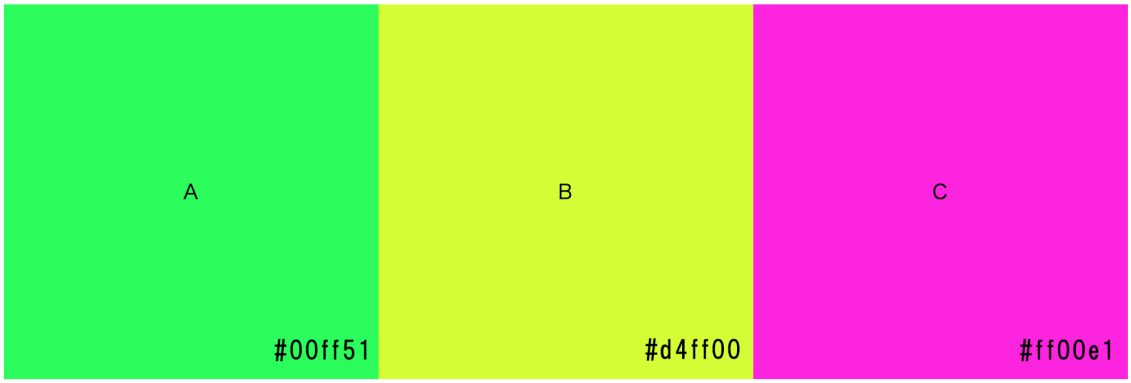
Q6

要素の並びを指定し、ブレイクポイントで入れ替わるようにしてください。

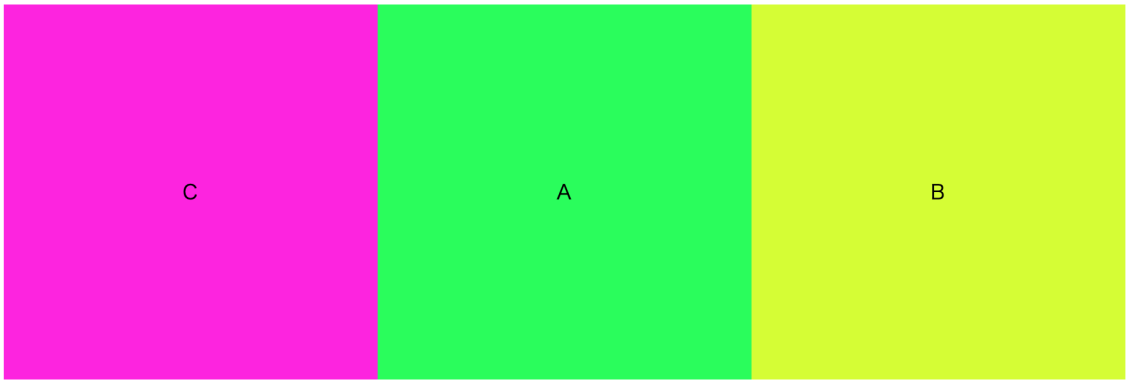
要素の横並びはflexboxを使用してください。

文字も中央で表示するようにしてください。

- PC



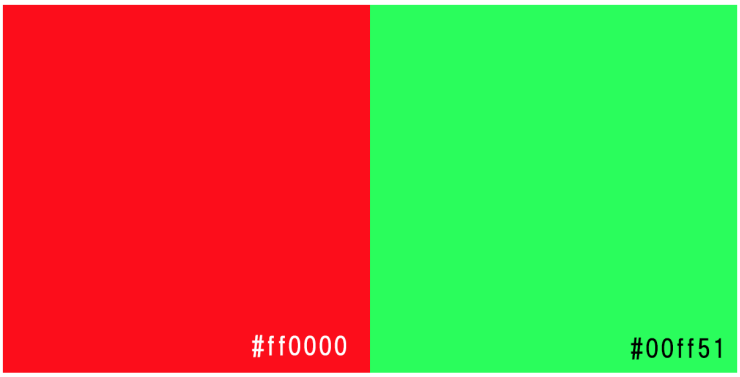
- SP



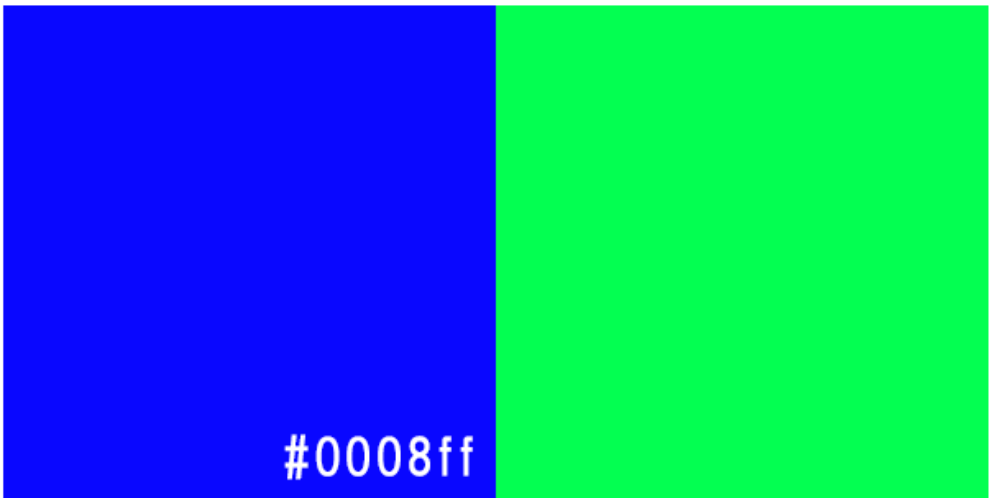
Q7

あらかじめ3つの要素の作成し、青い箱をブレイクポイント以上では非表示、ブレイクポイント未満で表示するようにしてください。

- PC



- SP



課題提出とレビュー依頼について

課題提出はまず作業が終えてからGitHubの自分のリポジトリにプッシュしてください。

GitHubのURLをClassroomにて提出してください。

レビュー依頼がありましたら、メンターがクローンして実際に確認します。

コードが正しく書けているかを確認した後、こちらからいくつか質問をするのでその答えが正しければ課題クリアとなります。

課題途中でも気になったことなどがあれば遠慮なく質問してください。