

WEB フロントエンド Chapter-11

「問い合わせページの作成（CSS 編）」

< 想定開発環境 >

OS：Windows 10 または 11

エディタ：Microsoft Visual Studio Code（通称 VSCode）

ブラウザ：Google Chrome

※ 参考教科書紹介

「HTML & CSS & Web デザインが 1 冊できちんと身につく本」：技術評論社

※ 注意事項

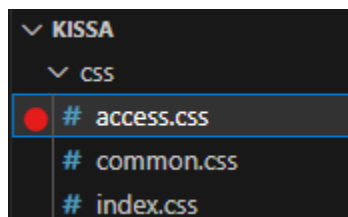
- ・ 講師の解説が始まったら私語は慎み、解説に集中してください。
- ・ 一度行った説明を再度行うケースが多く、授業進行が遅れてしまいます。
- ・ 教室にはマイクが無く、他の方の話声で解説内容が聞き取れない場合があります

① CSS ファイルの作成

まず初めに Access ページ用の CSS ファイルを準備しましょう。

次の画像の位置に access.css を**新規作成**してください。

<VSCode>



注意点ですが、html の時のように common.css を複製しないようにしてください。

access.css に記述するのはあくまで差分のみです。

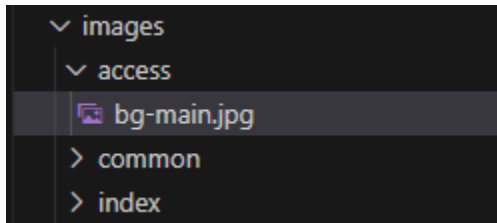
また、access.css の先頭部分には必ず「charset」の記述をしておきましょう。

```
@charset "utf-8";
```

◇ タイトル画像の追加

次の位置に配布画像を追加してください。

<VSCode>



※ images フォルダの中に access フォルダを作成し、その中に D&D します。

画像の位置と指定した path が違うと正しく表示されませんので気を付けてください。

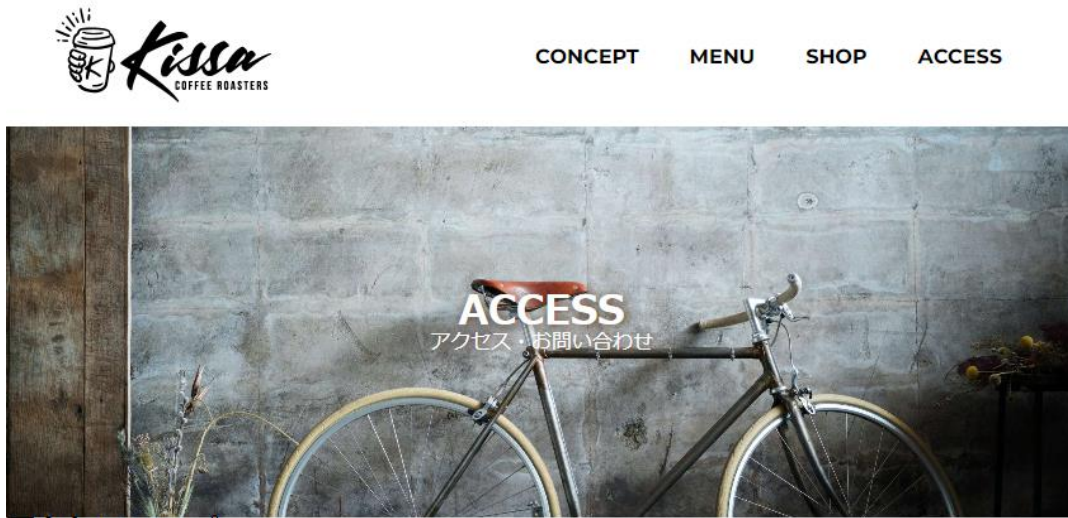
◇ タイトル部分の記述

タイトル部分にスタイルを当てます。次のセレクトアを追加しましょう。

<access.css>

```
.title {  
  height: 310px;  
  background-image: url(../images/access/bg-main.jpg);  
  background-repeat: no-repeat;  
  background-position: center center;  
  background-size: cover;  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
  align-items: center;  
  color: ■ #ffffff;  
  text-shadow: 1px 1px 10px ■ #4b2c14;  
}
```

<ブラウザ表示>



どのプロパティも既に解説済みですので、個別の解説については割愛します。

title 要素で `display: flex;` 指定しているのは、title 要素内の `<h1>` と `<p>` を整列させて並べるためです。 `flex-direction: column;` の指定でブラウザ上でそれぞれの要素が縦並びに、 `justify-content` と `align-items` を `center` に指定しているため、上下左右ともに中央揃えになっています。

◇ タイトル文字の調整

更に見栄えを良くするため、次のスタイルを追加して修正しましょう。

<access.css>

```
.title h1 {  
  font-family: 'Montserrat', sans-serif;  
  font-size: 32px;  
  font-weight: bold;  
}  
  
.title p {  
  font-size: 14px;  
  margin-top: 15px;  
}
```

<ブラウザ表示>



こちらにも既に解説したプロパティばかりですね。フォントが google フォントから追加した montserrat に置き換わって、より統一感が出ました。

◇ メインエリアの見出し調整

アクセスマップやお問い合わせの見出しが前後とくっついていて見づらいため、その部分を調整します。

<access.css>

```
.main h2 {  
  font-size: 22px;  
  font-weight: bold;  
  line-height: 30px;  
}  
  
.main h2::after {  
  content: '';  
  display: block;  
  width: 36px;  
  height: 3px;  
  background-color: #000000;  
  margin-top: 20px;  
}
```

<ブラウザ表示>

アクセスマップ



お問い合わせフォーム

お問い合わせ

今回当てたスタイルで最も注目して欲しいのは `::after` という記述です。

リセット CSS の回で少しだけ解説しましたが、これは疑似要素と呼ばれるセレクトタです。他にも `::before` などの疑似要素があります。

今回の `::after` の場合は、**指定したセレクトタの後に記述したスタイルが付加**されます。ブラウザ表示上で言うと、地図の左上に見える黒い棒線の部分が疑似要素で記述された箇所です。

画面表示はされているものの、この部分専用の html タグはありません。実際に `access.html` を確認してみるとわかると思います。

疑似要素を使うメリットとしては、記述を `css` だけで完結出来る点にあります。

通常のスタイルは html 上に配置した要素に対して `css` を書く必要がありますが、疑似要素の場合、付加する対象の要素が既にあれば html に追記する必要はありません。`css` だけ記述すれば OK ですし、見た目を変更する場合もその部分だけ書き換えれば良いので、メンテナンス時に有利な場合もあります。

今回のように線を表示したり、簡単な表示を行う場合に利用すると良いでしょう。因みに `content` プロパティの値に文字を設定すれば、その文字が画面に表示されます。

◇ マップエリアの調整

マップ表示エリアのスタイルを記述します。

ここではまず領域確保をしている親要素の記述をしています。

<access.css>

```
.map {  
  width: 930px;  
  max-width: 90%;  
  margin-top: 75px;  
  margin-left: auto;  
  margin-right: auto;  
}
```

上の CSS では単に領域を確保しているだけなので、ブラウザ上の表示はほとんど変わりません。マップ自体は<iframe>というタグに記述されていますので、次のような指定で地図を親要素の幅いっぱいに表示させます。

<access.css>

```
.map iframe {  
  display: block;  
  width: 100%;  
  height: 320px;  
  margin-top: 25px;  
}
```

<ブラウザ表示>

アクセスマップ



地図がサイトの中央に収まり、見栄えが良くなりました！

以前も少しだけ解説しましたが、HTML 上でどんなタグ構造作ればよいかわからない場合、次のような考えで作成するとレスポンス対応しやすい構造に出来ます。

- ・まず初めに、表示させたい要素（例えば画像など）を配置する為のエリアを確保する。レスポンス対応は基本的にここで行う。
- ・表示させたい要素は、上で準備したエリアの子要素として配置する。

コツとしては表示させたい要素をいきなり配置するのではなく、必ず領域を確保するためのエリアを親要素で用意し、表示させたい要素は子要素として配置する事です。

というのも、スタイルには親要素からでないと指定出来ないものが沢山あるからです（例：display, justify-content, align-items, text-align などなど）

親要素の領域確保エリアでレスポンス対応をし、子要素はあくまで表示させるだけという形で実装をすると、想定した表示がしやすくなります。

◇ お問い合わせエリアの調整

前のページで説明した通り、ここでもまずはフォームを配置するエリア全体にスタイルを当てて行きます。

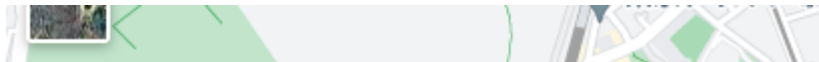
お問い合わせエリア全体には contact というクラス名が付けられているので、contact クラスをセレクトアにして記述するのですが、統一感を出す為にマップ表示エリアと全く同じスタイルを適用します。

<access.css>

```
.contact {  
  width: 930px;  
  max-width: 90%;  
  margin-top: 75px;  
  margin-left: auto;  
  margin-right: auto;  
}
```

セレクトア名以外は.map と全て同じ内容です。コピー＆ペーストしてしまいましょう。

<ブラウザ表示>



お問い合わせフォーム

お名前

メールアドレス

お電話番号

お問い合わせ種別

※ 少しわかりづらいが、マップエリアと同じ位置に配置されている

.map と.contact のように、中身が全く同じセレクトを複数用意するのはあまり良い事ではありません。こういった場合は html 側で同じクラス名を付け、セレクトは一つにまとめたほうがメンテナンス性から考えても適切です。

ただしその場合はクラス名には「map」や「contact」といった、特定のコンテンツを表す名前は相応しくありませんので、クラス名はコンテンツの内容ではなく、その役割を示したものを付けると良いでしょう。

今回のケースであれば「content-area」や「content-wrapper」などといった名称が適切かも知れません。

因みに wrapper という単語はフロントエンド開発ではよく使われます。

同じカテゴリーの部品をまとめたものを良く「コンテンツ」と呼んだりしますが、それらを wrap（包み込む）するように配置された要素なので wrapper（包み込むもの）というわけです。

今回は教材に準拠して進めておりますので、このまま別々のスタイルとして記述していきますが、自分自身でサイトを構築する場合は、その点に気を付けて記述してみてください。

◇ フォームエリアのスタイル調整

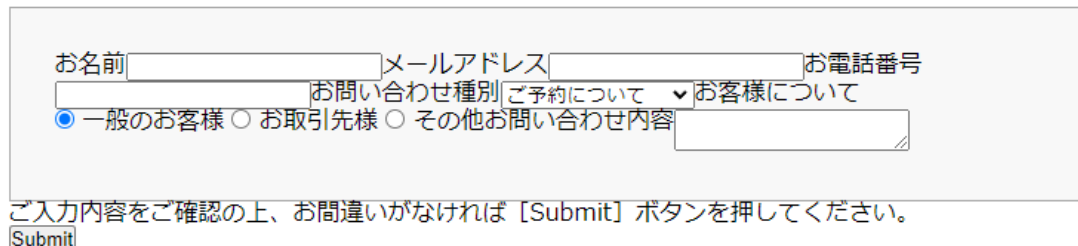
フォームの本体に背景色を付け、入力箇所がどこなのかを分かりやすくします。

<access.css>

```
.form-area {  
  background-color: #f8f8f8;  
  border: 1px solid #aaaaaa;  
  margin-top: 25px;  
  padding: 30px;  
  display: flex;  
  flex-wrap: wrap;  
}
```

<ブラウザ表示>

お問い合わせフォーム



表示が崩れていますが、これは flex-wrap: wrap; の影響によるものです。

親要素に wrap 指定すると、その子要素は横並びになり、右端で折り返して表示されるようになります。

なぜこのような記述をしているかというと、入力フォームのタイトル（お名前、メールアドレスなど）と実際の入力フォームを並べて表示させるためです。

子要素に次のようなスタイルを当てれば、想定通りに表示されると思います。

<access.css>

```
.form-area dt {  
  width: 200px;  
  padding: 15px 0;  
  font-size: 15px;  
  font-weight: bold;  
  line-height: 24px;  
}  
  
.form-area dd {  
  width: calc(100% - 200px);  
  padding: 15px 0;  
}
```

<ブラウザ表示>

お問い合わせフォーム

お名前	<input type="text"/>
メールアドレス	<input type="text"/>
お電話番号	<input type="text"/>
お問い合わせ種別	<input type="text" value="ご予約について"/>
お客様について	<input checked="" type="radio"/> 一般のお客様 <input type="radio"/> お取引先様 <input type="radio"/> その他
お問い合わせ内容	<input type="text"/>

一気に綺麗に並びました！

ポイントとしては、入力フォームのタイトルを 200px で固定し、残りの幅を calc で計算させている点です。

このように指定すればタイトルは全て同じ幅で固定されているので縦に綺麗に並び、残りのフォームは親要素の幅一杯まで広がります。

実際にウィンドウサイズを変更して確認してみましょう。ウィンドウサイズを変えても綺麗に並んだまま表示されているのがわかると思います。

◇ 必須入力項目に目印を付ける

html 上ではいくつかのフォームに required という属性を付けたと思います。required を付けると入力が必要になるため、その欄への入力がないとデータ送信出来なくなっているわけですが、現状ではサイト閲覧者が見ても、どの項目が必須なのかわかりません。

そこで required となっている要素に、入力必須である事がわかるように目印を付けていきます。

次のようなセレクトを追加してください。

追加する場所はわかりやすいように `.form-area dt` の後 が良いでしょう。

<access.css>

```
.form-area dt .required::after {  
  content: '必須';  
  font-size: 11px;  
  color: ■ #eb4f32;  
  margin-left: 10px;  
}
```

<ブラウザ表示>

お名前 必須	<input type="text"/>
メールアドレス 必須	<input type="text"/>
お電話番号	<input type="text"/>
お問い合わせ種別	ご予約について ▼
お客様について	<input checked="" type="radio"/> 一般のお客様 <input type="radio"/> お取引先様 <input type="radio"/> その他
お問い合わせ内容 必須	<input type="text"/>

.required の後ろに ::after という疑似要素が付けられていますので、required クラスが付いている要素のみに必須表示が追加されているのがわかると思います。

もし今後必須入力項目が追加されても、その項目に required というクラス名を追加するだけで必須の文字が表示されます。

最近では文字表記ではなく、アスタリスク（*）で入力必須であることを示す場合も多いです。あまり文字を沢山表示させたくない場合は、アスタリスク表記にしても良いでしょう。

◇ 入力フォームのサイズを変更

かなり見た目も整って来ましたが、現状だと入力欄やセレクトボックスが小さいのでクリックしづらく、あまり操作性が良いとは言えません。

入力欄とセレクトボックスのスタイルを調整しましょう。

<access.css>

```
.input-text {  
  width: 100%;  
  max-width: 280px;  
  height: 40px;  
  padding-left: 10px;  
  padding-right: 10px;  
}  
  
.select-box {  
  width: 200px;  
  height: 40px;  
}
```

<ブラウザ表示>

お名前 必須	<input type="text"/>
メールアドレス 必須	<input type="text"/>
お電話番号	<input type="text"/>
お問い合わせ種別	<div>ご予約について ▼</div>

幅や高さを少し変更しただけで、デフォルトのフォーム部品よりも随分と見栄えが良くなりました。

◇ ラジオボタン周りの調整

「お客様について」の項目はラジオボタンで選択するようになっていますが、それぞれの項目が近いので少し圧迫感もあり、クリック箇所を間違える可能性もあります。

前後に間隔を空けるという対処法もありますが、今回は要素をブロックレベル要素に変更し、縦に並べる方法を取ります。

<access.css>

```
.radio-button {  
  display: block;  
  margin-top: 20px;  
}
```

<ブラウザ表示>

お客様について

- ☒ 一般のお客様
- ☐ お取引先様
- ☐ その他

綺麗に並んだ……ように見えますが、よく見ると一番最初の項目がタイトルよりも下に表示されていて、少し不格好に見えます。これは全ての項目に margin-top が適用されているからです。

ラジオボタンには全て .radio-button という同じクラス名が付いているが、一番上の項目だけには margin を付けたくない。

かと言ってそれぞれの項目に別のクラス名を付けてスタイルを変えるのも大変です。

こういった場合、どうしたら良いでしょうか。

実は次のようなスタイルを追加すれば解決できます！

<access.css>

```
.radio-button:first-child {  
  margin-top: 0;  
}
```

<ブラウザ表示>

お客様について

☒ 一般のお客様

☐ お取引先様

☐ その他

タイトルと先頭の項目が綺麗に並びました！

【:first-child】疑似クラス

:first-child という疑似クラスは、兄弟要素（同じ階層に並んでいる要素）の中で一番先頭にある要素に付けられるクラス名です

指定した要素が一番先頭に無い場合は、どの要素にも first-child クラスは付きません。
（文面で説明するのは大変なので、授業内でコーディングして説明します）

似たような疑似クラスに、:last-child というものもあります。

一番最後の要素に付けられる以外、利用方法は同じです。

◇ カーソルの見た目を変更

この項目は大元の教材にはありませんが、個人的には結構重要と考えていますので追加しておきます。

今の状態でセレクトボックスやラジオボタンにカーソルをマウスオーバー（上に載せること）させても、カーソルには何も変化はありません。

テキスト入力の場合はどうでしょうか？

カーソルが矢印からアルファベットの **I** のような見た目に変わったと思います。

こういった変化は閲覧者にとってみれば「ああ、ここは入力できる場所なんだな」と認識する事が出来るため、ユーザビリティの向上に役立ちます。

そこでセレクトボックスとラジオボタンに、次のプロパティを追加してみてください。

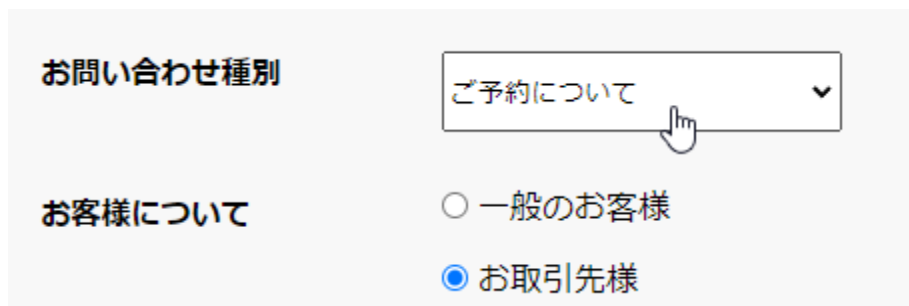
<access.css>

```
.select-box {  
  width: 200px;  
  height: 40px;  
  cursor: pointer;  
}
```

<access.css>

```
.radio-button {  
  display: block;  
  margin-top: 20px;  
  cursor: pointer;  
}
```

<ブラウザ表示>



お問い合わせ種別

ご予約について

お客様について

☐ 一般のお客様

☒ お取引先様

どうでしょう？ そここが入力フォームであると、直感的にわかると思いませんか？

このように、入力可能な要素にマウスオーバーした場合、その入力フォームに対応したカーソルに変更するようにしておくと、操作する人にとってわかりやすいUIになります。

◇ テキストエリアの調整

長文入力をするための textarea 要素ですが、今の状態では一行入力と変わらない見た目になっています。

textarea 要素には message というクラス名が付けられていますので、次のようなセレクタを追加します。

<access.css>

```
.message {  
  width: 100%;  
  height: 260px;  
  padding: 10px;  
  line-height: 1.5;  
}
```

<ブラウザ表示>

お問い合わせ内容 **必須**

特に解説の必要な所は無いと思います。line-height のデフォルト値は 1.2 程度ですが、それだと各行が詰まりすぎていて圧迫感があるため 1.5 にしています。

実際入力して確認してみて、納得のいく数値に変更するのも良いと思います。

◇ 注意書きのスタイル調整

アクセスページのスタイルも、残すところボタン周りだけとなりました。
まずは送信する際の注意書き文のスタイルを整えます。

<access.css>

```
.confirm-text {  
  font-size: 14px;  
  line-height: 22px;  
  margin-top: 30px;  
}
```

<ブラウザ表示>

ご入力内容をご確認の上、お間違いがなければ【Submit】ボタンを押してください。

Submit

line-height の設定は別に無くても大して変わらないように見えますが、ブラウザのウィンドウ幅を狭くすると文章が折り返し表示されます。

デフォルトの数値だと行間隔が狭くなって見苦しくなってしまうので、余裕をもって表示されるよう指定されています。

◇ ボタンのスタイルの流用

ブラウザの表示を見てもわかるとおり、送信ボタンがデフォルト表示になっています。ボタンのような部品の見た目はサイト内で統一されていた方が良いので、以前 index.html 上で表示させていたコンセプトへのリンクボタンのスタイルを流用しましょう。

スタイルは次の手順でコピーします（スタイルなのでコピー元は **index.css** です）

1. index.css を開き、.link-button のスタイルをセレクトごとコピー。
2. access.css の一番 (.confirm-text の) 後ろにペースト。
3. .link-button を .submit-button にリネーム。次のようになるはずです。

<access.css>

```
.confirm-text {  
  font-size: 14px;  
  line-height: 22px;  
  margin-top: 30px;  
}  
  
.submit-button {  
  background-color: #f4dd64;  
  display: inline-block;  
  min-width: 180px;  
  line-height: 48px;  
  border-radius: 24px;  
  font-family: 'Montserrat', sans-serif;  
  font-size: 14px;  
}
```

<ブラウザ表示>

ご入力内容をご確認の上、お間違い

Submit

まだこれで完全では無いので、この後スタイルをもう少しだけ調整します。

◇ 送信ボタンのスタイル修正

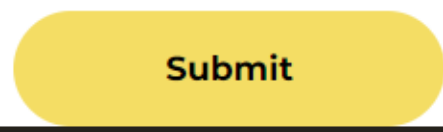
上下の余白や border の調整がまだなので、以下のプロパティを追加します。

<access.css>

```
.submit-button {  
  background-color: #f4dd64;  
  display: inline-block;  
  min-width: 180px;  
  line-height: 48px;  
  border-radius: 24px;  
  font-family: 'Montserrat', sans-serif;  
  font-size: 14px;  
  margin-top: 20px;  
  cursor: pointer;  
  border: none;  
}
```

<ブラウザ表示>

ご入力内容をご確認の上、お帰



cursor についてはラジオボタンの項で説明しました。

入力する値などによってカーソルの形を変えれば、閲覧者がどんな操作をすれば良いのかが直感的にわかるようになります。

また、今は HTML と CSS のみなのでこの程度しかできませんが、Javascript を扱うようになると更に細かい制御を行ったりします。

例えば入力フォームに必要な値が入っていない場合は、カーソルがデフォルトのまま変化せず（＝送信できない事を伝える）、なおかつボタンも実際に押せない、というような実装を良くします。

人は色や形のほうがより素早く判断出来るため、フロントエンドではビジュアル面のブラッシュアップは非常に大切です。

◇ hover 時の対応、フッターとの余白

今回はこれで最後です！ 次のスタイルを追加しましょう！

<access.css>

```
.submit-button:hover {  
  background-color: #d8b500;  
}  
  
.footer {  
  margin-top: 100px;  
}
```

フッターとの余白が開くだけなので、ブラウザ表示は割愛します。

hover については以前説明したとおり、マウスカーソルが上に来た時に付加される疑似クラスです。疑似クラスが付く前のスタイルが継承されますので、変更したいプロパティだけ指定すれば良いです。

② 今回のまとめ

入力フォームは閲覧者が実際に操作部分なので、とにかく見やすく、入力しやすく、勘違いをさせない工夫が重要です。調整に手間はかかりますが、操作性の悪いサイトはあっという間に淘汰されてしまいますので、手を抜かない事が大事です。

ITに限らず、ユーザビリティはどの業界でも非常に重要です。常に意識する癖を付けておくと良いでしょう。