WEB フロントエンド Chapter-08
「フルスクリーンレイアウト」

＜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

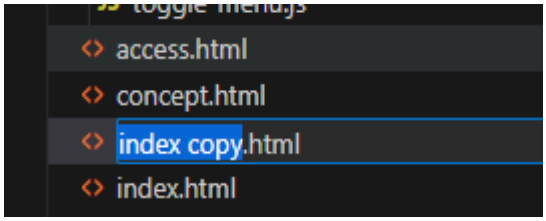
※ 注意事項

・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

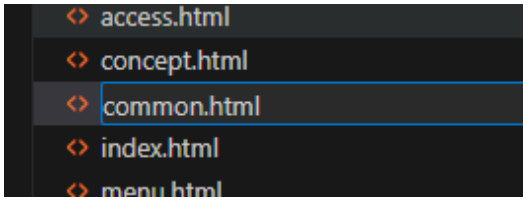
①　index.html を複製し、共通ページとして保存する

前回までで基礎部分は完成しましたので、今回からは各ページを作っていきます。
index.html はサイトの TOP ページになりますが、このまま変更を加える前に、まずは index.html を複製し、common.html として保存しておきましょう。他のページはこの common.html を元に作っていったほうが楽だからです。
複製の手順は次の通りです

・index.html をクリックして選択します。
・ctrl+c でクリップボードにコピーします
・ctrl+v でその場に貼り付けます。すると次のようなファイルが出来るはずです



複製して出来た index copy.html を「common.html」に rename します



common.html は現状、今まで書いてきた index.html と全く同じものです。
今後 common.html には変更を一切加えず、必ずコピーして使っていきます。

②　フルスクリーンレイアウトの作成

index.html はサイトのトップページ（本来はこのページをホームページと呼んでいました）になりますので、まずページ概要を修正します。
単なる文章なので、下のコードから**コピー＆ペーストして構いません。**

<index.html>

```
<meta name="description" content="自家焙煎したこだわりのコーヒーと、思わず長居したくなるような居心地の良い空間を提供するカフェ「KISSA」のウェブサイトです。">
```

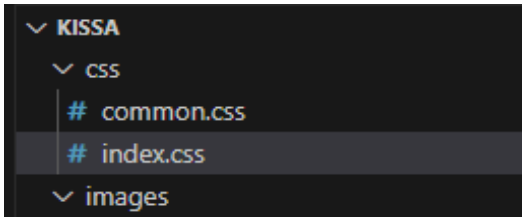
◇　CSS ファイルの追加

今まで CSS は「common.css」に書いてきましたが、これには各ページに共通する部分だけを記述して来ました。
しかしこれからはページごとに異なる部分、いわば差分を記述していきますので、css にも差分が必要になります。
今回は index.html を編集しますので、その差分の css として「index.css」を作成し、link 要素で読み込んで使用します。

<index.html>

```
<link href="./css/common.css" rel="stylesheet">
<link href="./css/index.css" rel="stylesheet">
```

<VSCode のエクスプローラー>



ファイルの保存位置とファイルパスに気を付けてください。

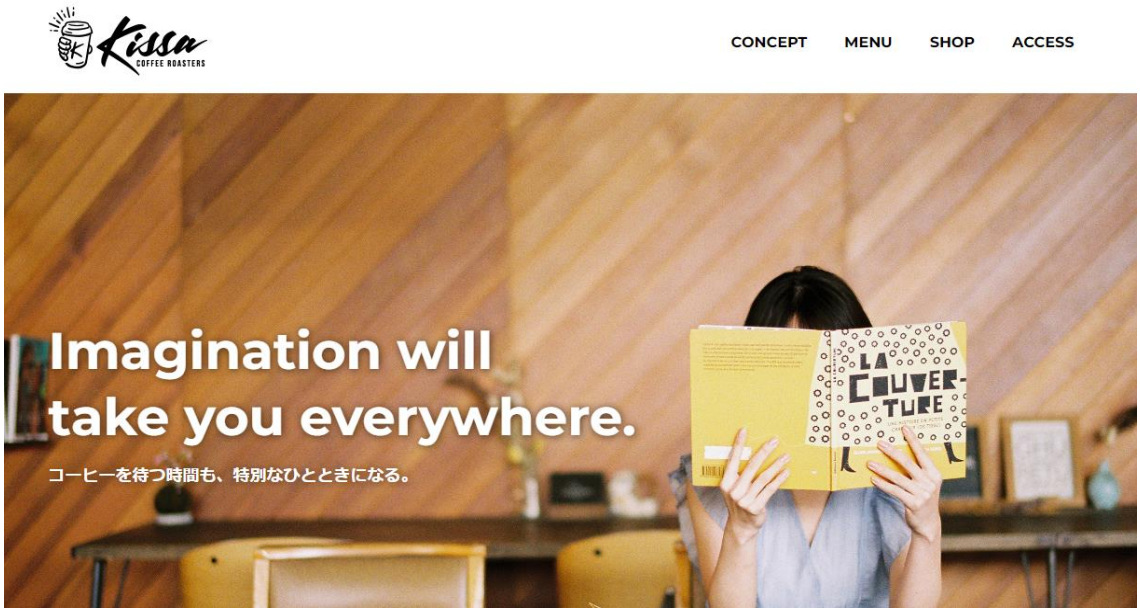
ファイルの保存場所、path 指定については今まで何度も説明してきましたので、今後は基本的に詳しい解説は行いません。css ファイルは必ず css フォルダ内に保存していきますので、今のうちに覚えておきましょう。

◇　html タグの記述

index.html はこのサイトのトップページに当たる部分です。

まずは PC 表示（フルスクリーンレイアウト）の完成イメージを見てみましょう。

<index.html ブラウザ表示　上部>



<index.html ブラウザ表示　下部>

「想像力はあなたをどこにでも連れて行ってくれる」
注文を待つ間に広げた、一冊の本の中に見つけたことば。
ゆったり流れる時間の中で、想像をふくらませる楽しさを思い出す。
そんな時間を過ごすとき、おいしいコーヒーがあるとうれしい。

CONCEPT

　実際の業務ではデザインが決まっている場合、デザインを見ながらどのような要素を追加すればそのレイアウトを実現出来るかを考えます。
　しかしそれがスムーズに出来るようになるのは、制作に慣れてからです。後からいくらでも修正出来るものなので、初めは「ボックスモデル」を意識して、上から順番に要素を書いて行きましょう。
　そうやって数々の修正していくうちに、自然と力が身に付くようになるものです、

　それではまずは index.html の main 要素内に記述していきしょう！
　スタイルを当てていないのでごちゃごちゃな表示になりますが、今は自分の想定通りに並んでいるかどうかを気にしていれば OK です。

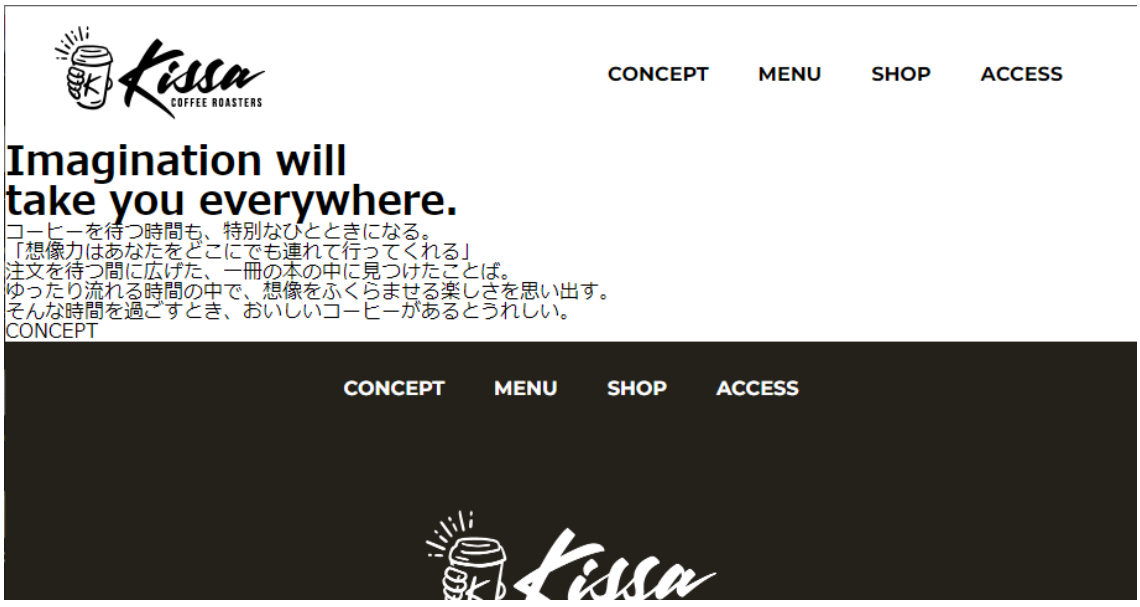
　lead クラス中のメッセージ分は長いので、次の文をコピペして OK です。

「想像力はあなたをどこにでも連れて行ってくれる」注文を待つ間に広げた、一冊の本の中に見つけたことば。ゆったり流れる時間の中で、想像をふくらませる楽しさを思い出す。そんな時間を過ごすとき、おいしいコーヒーがあるとうれしい。

<index.html>

```
<main class="main">
    <div class="first-view">
        <div class="first-view-text">
            <h1>Imagination will <br>take you everywhere.</h1>
            <p>コーヒーを待つ時間も、特別なひとときになる。</p>
        </div>
    </div>
    <div class="lead">
        <p>「想像力はあなたをどこにでも連れて行ってくれる」<br>注文を待つ間に広げた、一冊の本の中に見つけたことば。<br>ゆったり流れる時間の中で、想像をふくらませる楽しさを思い出す。<br>そんな時間を過ごすとき、おいしいコーヒーがあるとうれしい。</p>
        <div class="link-button-area">
            <a class="link-button" href="./concept.html">CONCEPT</a>
        </div>
    </div>
</main>
```

<ブラウザ表示>



p 要素はブロックレベル要素なので、br 要素で改行するようにしています
また lead 要素内はお店のコンセプトに繋がる導入的な内容が書かれています。そのため、すぐ下に配置したリンクボタンは、最も関連の深い concept ページに遷移するようにしています。

◇　CSS の初めに必ず書く事

html タグは準備出来ましたので、今度は index.css に記述していきましょう。まだ作成出来ていない方はこの pdf の 3 ページ目に css の作成場所が説明されていますので、そちらを参考に作成してください。

さて、今回は css ファイルを新規に作成しました。
一番初めに記述するコードは common.css と同じです。

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

これですね！　忘れずに記述しましょう。

③　見出し・キャッチコピー部分（first-view クラス）の記述

first-view エリアは背景画像の上に文字を表示するという構造になっていますが、可能であれば端末サイズの高さに合わせてピッタリ表示させたい所です。
今回はブラウザの表示領域を扱う事の出来る vh(View port Height)という単位を使います。index.css に次のスタイルを追加しましょう。

<index.css>

```
.first-view {
  height: calc(100vh - 110px);
}
```

上のキャッチコピーと下のメッセージとの間に、かなり広い間隔が出来たと思います。

【vw : Viewport Width】
【vh : Viewport Height】単位
　CSS 上で要素や文字のサイズを表す時は px（pixel）が一般的ですが、px のような固定値だとレスポンシブ対応時では都合の悪い場合があります。
　特に広い領域を持つ要素の場合にはそれが顕著で、ブラウザをウィンドウ表示させるとレイアウトが崩れてしまうなど、設定値に苦労したりします。
　そこでよく使われるのが vw と vh です。
　viewport、つまり各端末のブラウザの表示領域を 100%として利用出来る単位で、100vw なら横幅一杯、50vh なら高さの半分を指定する事が出来ます。
　今回は 100vh と指定していますが、これはブラウザの表示領域と同じ高さを表します。

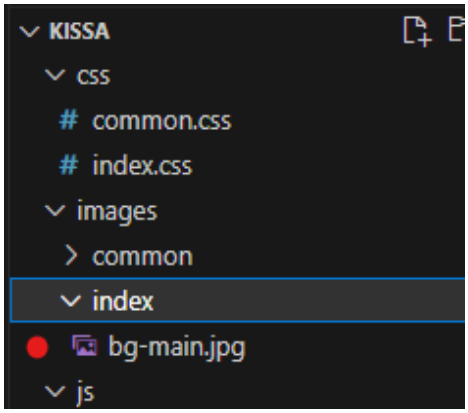
【calc】値
　calc は calculate（計算する）の略で、文字通り計算した結果を反映できる、とても特殊な値です。
　今回の例では 100vh（端末の表示部分の高さ）から 110px を引いています。
　なぜ 110px 引いているかというと、ヘッダーメニューの高さが 110px だからです。
　実際に検証ツールで確認してみると良いでしょう。

　calc と vw, vh の組み合わせはとても便利ですので、覚えておくと良いでしょう。

◇　背景画像の表示設定

　背景には画像を設定しますのでまず images の中に index フォルダを作り、今回配布する bg-main.jpg をその中に配置します。VSCode 上では次の位置になります。

<VSCode：エクスプローラーでの表示>

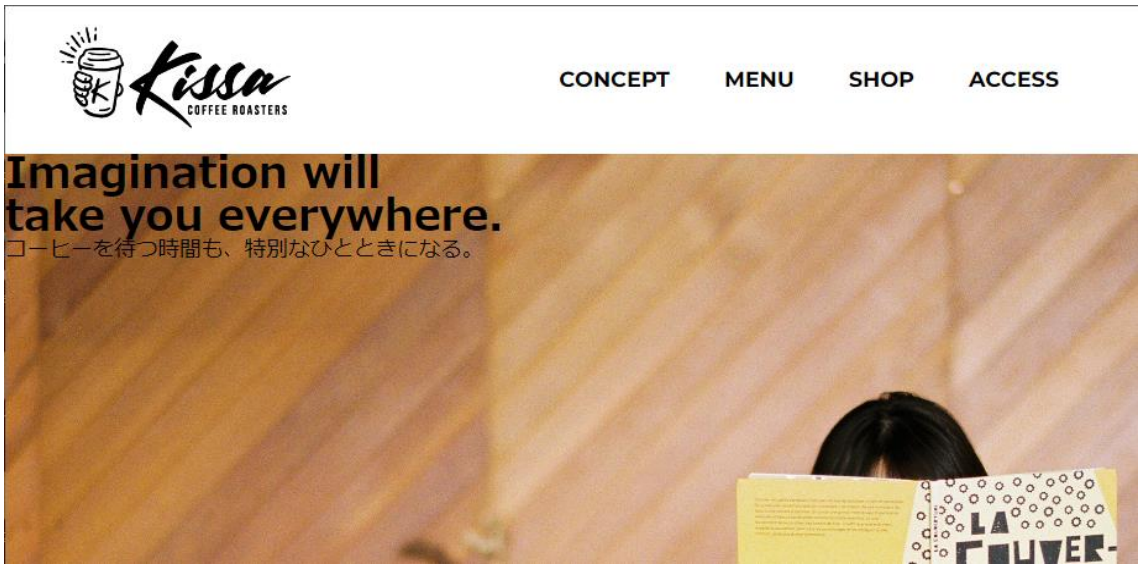


画像を配置し終えたら、first-view セレクタにスタイルを追加していきましょう。

<index.css>

```css
.first-view {
  height: calc(100vh - 110px);
  background-image: url(../images/index/bg-main.jpg);
  background-repeat: no-repeat;
  background-position: center center;
  background-size: cover;
}
```

<ブラウザ表示>



背景画像が表示されました。ぐっと完成イメージに近づきましたね。
画像が表示されない方は、

・画像ファイルは正しいフォルダに収められているか？
・指定した画像パス(URL)は正しいか？
・セレクタ名は正しいか？　html ファイルに書いたクラス名と違っていないか？

といった所を中心に確認してみましょう。

【background-repeat】プロパティ　default 値 : repeat;
　ハンバーガーボタンの際にも登場しましたが、背景画像を繰り返し表示するかどうかを設定するプロパティです。
　元の画像が大きすぎるため現状では設定してもその効果は分かりづらいと思います。
　ですが no-repeat を repeat に変更し background-size を contain に変更すれば、画像が並んで表示されているのがわかると思います。

【background-position】プロパティ　default 値 : top left;
　背景画像の表示位置を指定するプロパティです。値は 2 つまで設定可能で、第一引数（一つ目の値）は水平位置、第二引数は垂直位置を指定します。デフォルト値では親要素の左上に合わせて表示されます。
　また値を一つしか設定しなくても問題ありませんが、その場合は第二引数は自動的に center になります。つまり今回は 2 つ値を指定していますが、一つしか設定しなかったとしても全く同じ指定になります。

　エンジニアにもよりますが、個人的にはこういった場合、敢えて明示しておいたほうが良いと私は考えます。省略してしまうと、それが単に記述を忘れたからなのか、それとも自動で center が入る事を知っているから書かなかったのかの判断が付かないからです。

【background-size】プロパティ　default 値 : auto;
　このプロパティは少々わかりづらいかも知れないので、値の例を挙げて説明します。

contain　：**要素内に余白が出来たとしても元画像を全て表示**
cover　　：**画像がはみ出たとしても、要素内全てに画像を表示**
px, %　　：指定した値で表示する。値が一つ場合は width 指定で、height は auto

　px や%は指定された通りに表示されます。値を 2 つ指定した場合、元の比率と異なっていたとしても指定通りに変形して表示されます。

　contain と cover については、どちらも縦横比が維持されます。文章だけでは少し理解しづらいかと思いますが、要するに元画像を全部表示させる事を優先させるか、それとも領域内全てに画像を表示させる事を優先させるかの違いです。
　今回のようにあくまでイメージ画像としての利用の場合は cover のほうが都合が良いでしょう。

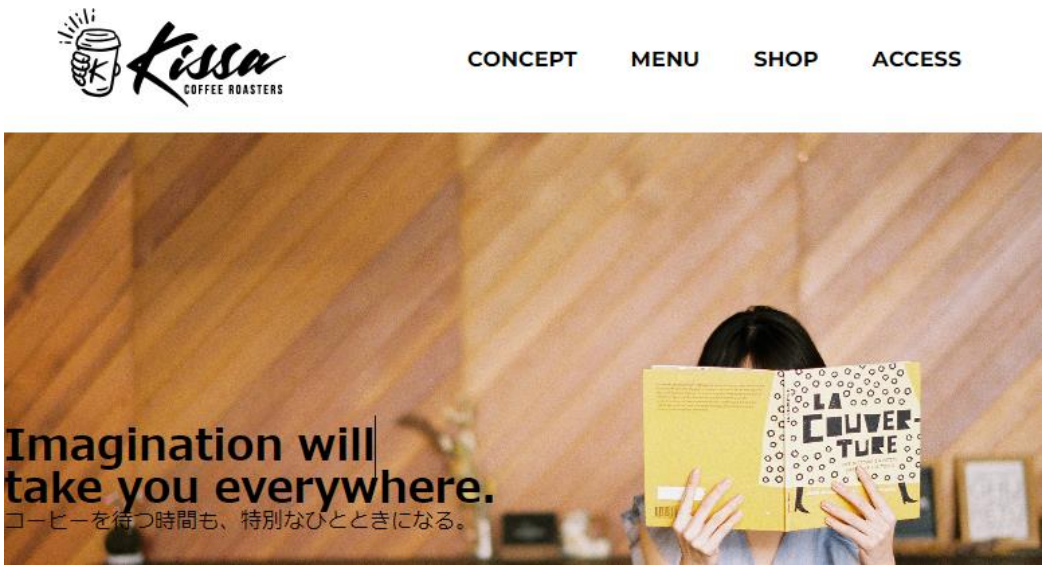
◇　見出しとキャッチコピーの表示位置修正

　イメージ画像の上に表示されている英文見出しとキャッチコピーの位置がバランス的にあまり良くないので、それを修正します。

<index.css>

```css
.first-view {
  height: calc(100vh - 110px);
  background-image: url(../images/index/bg-main.jpg);
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  display: flex;
  align-items: center;
}
```

<ブラウザ画面>



　一瞬 display : flex;を指定しているので横並びになるかと思うかもしれませんが、flex 指定した直下は first-view-text クラスと名付けた div 要素一つだけです。並び方(justify-content)の指定もしていないのでデフォルトの左寄せになっています。

　ここでは align-items で縦位置の調整だけ行っています。

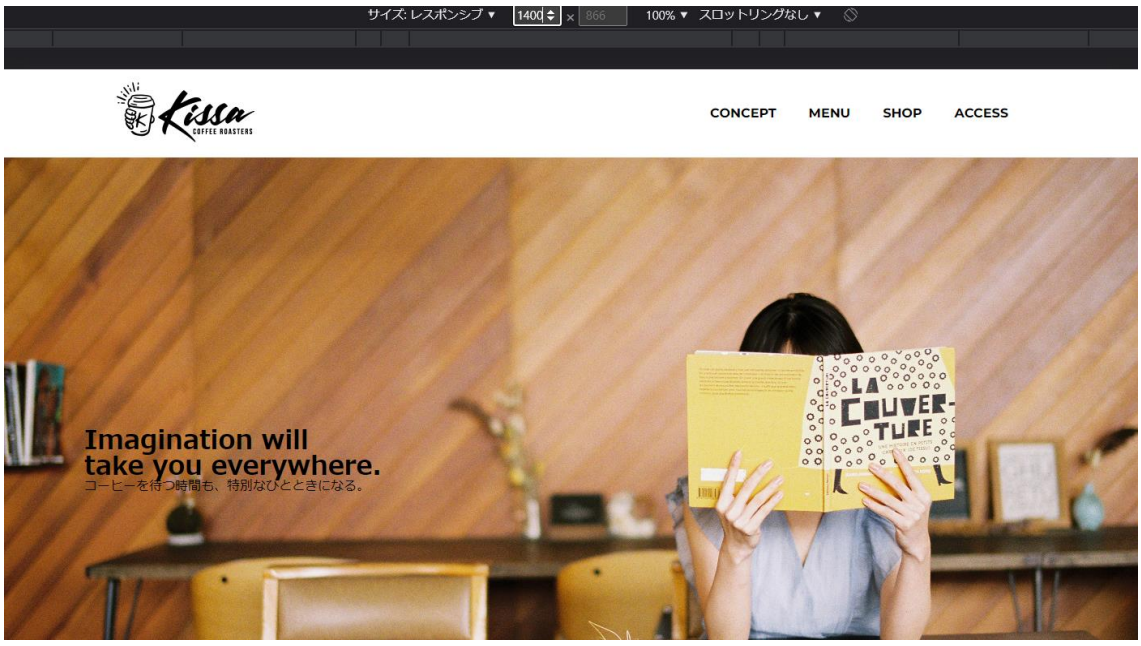
◇　見出しとキャッチコピーのスタイリング

　親要素で表示位置の調整はしましたので、今度は子要素である first-view-text クラス自体の見た目の修正をしていきましょう。
　次のセレクタを追加し、プロパティを記述してください。

<index.css>

```css
.first-view-text {
  width: 100%;
  max-width: 1200px;
  margin-left: auto;
  margin-right: auto;
}
```

<ブラウザ表示（width 1400px 時）>



　テキスト部分を max-width で 1200px に指定しているため、1200px 以下だと margin が無くなって変化が分からなくなります。変更を確認する時は上の画像のように 1200px 以上で表示してください。

書いたスタイルの割に表示的には少ししか変更されていないように見えますが、実は結構様々なスタイルが適用されていた結果、今のような記述になっています。

まず first-view-text クラスが適用されている div 要素ですが、元々ブロックレベル要素ですのでデフォルトでは親要素一杯に幅が広がっています。

しかし親要素で flex 指定された事により、width が中身依存（今回の場合は h1 や p 要素の中の文字列）に自動変更されます。こうしないと横並びに出来ないからです。

そこで今回再び width を 100%と明示して指定する事で、元のように横幅一杯まで広がるようになります。

CSS で厄介なのはデフォルト値がある事と、今回のように設定によってスタイルが自動で変わる点にあります。CSS のように設定項目が大量にあるものの場合、全ての項目を指定するのは非常に大変ですので、デフォルト値が決まっていたり、自動設定してくれるようになっています。これは 3D モデリングなどでも同じような事が言えるでしょう。

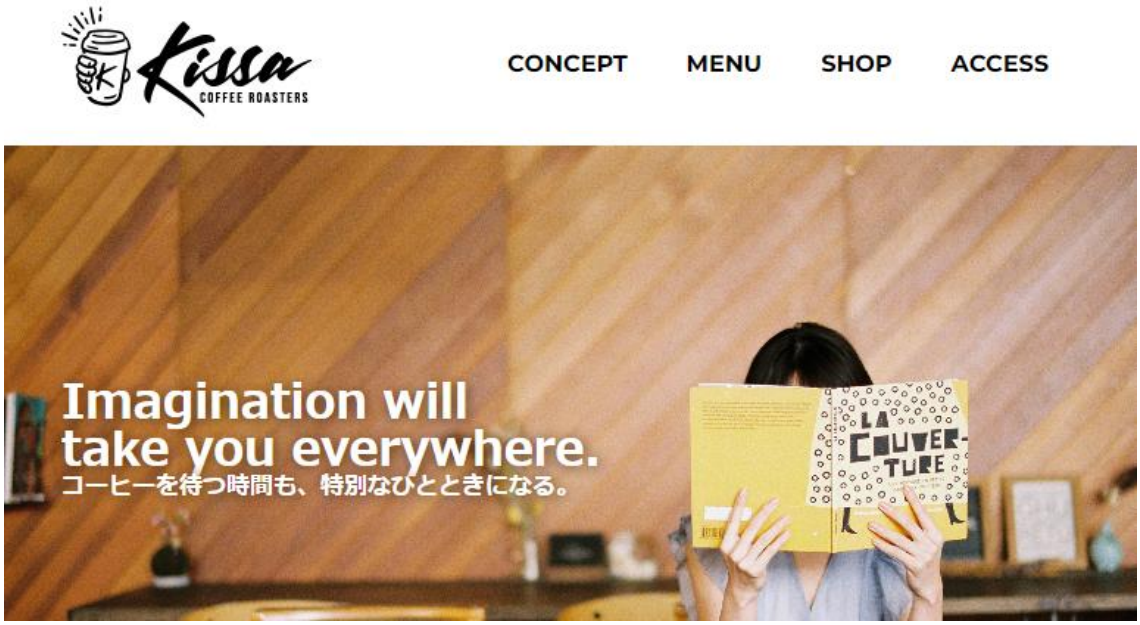
ですのでフロントエンドは仕組みを覚えようとするよりも、実際に自分で設定を色々変更し、ブラウザで確認したほうがより上達しやすいです。プロパティや値をどんどん変更して、その結果を画面で確認しながら覚えましょう
（授業プロジェクトの場合は元に戻すのを忘れずに！）

◇　見出しとキャッチコピーのスタイリングその 2

<index.css>

```css
.first-view-text {
  width: 100%;
  max-width: 1200px;
  margin-left: auto;
  margin-right: auto;
  padding-left: 40px;
  padding-bottom: 80px;
  color: #ffffff;
  font-weight: bold;
  text-shadow: 1px 1px 10px #4b2c14;
}
```

<ブラウザ表示>



まず padding ですが、これは完全にバランスを取るために設定しています。画面を小さくした時に少し左を空けるのと、下に余白を作る事で、結果的に文字を少し上に移動させています。表示位置を変える方法は色々ある（例えば position : absolute;を使う方法など）のですが、個人的には margin や padding を使って調整するほうが大きく崩れたりせず、よりスマートだと考えています。

最後の text-shadow についてですが、これは以前ヘッダー部分で記述した box-shadow と設定方法は同じです。写真の上に文字を置いた場合、下地の色次第では見づらくなることがあるので、こういった調整をしています。

◇　見出しとキャッチコピーの文字のスタイリング

目標のデザインにかなり近づいてきました。あとは文字自体にのスタイルを当てれば、この部分は完成です！

見出しは h1 要素、キャッチコピーは p 要素で書かれていますので、それぞれ別のスタイルを当てます。

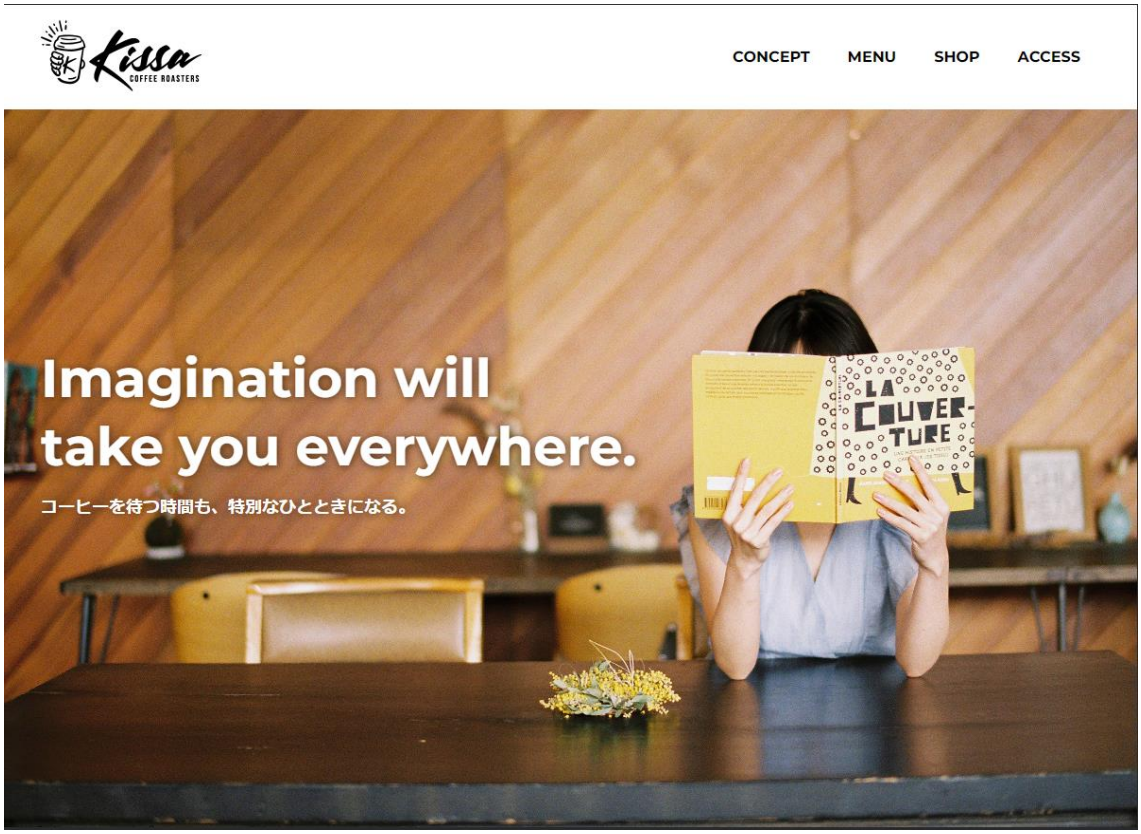
次のセレクタを新たに追加し、記述してください。

<index.css>

```css
.first-view-text h1 {
  font-family: 'Montserrat', sans-serif;
  font-size: 56px;
  line-height: 72px;
}

.first-view-text p {
  font-size: 18px;
  margin-top: 20px;
}
```

<ブラウザ表示>



プロパティについてはほとんど使った事のあるものばかりなので割愛し、ここではセレクタの指定方法について少し触れます。

今回セレクタを「.first-view-text p」のように、スペース繋ぎで 2 つ指定しています。今までも同じような何度か指定してきましたが、これは「**子孫セレクタ**」と言って、指定した要素内にあるものだけが対象になります。

つまり今回の例だと、「first-view-text クラス内にある p 要素」のみを指定しているので、first-view-text 以外の要素内にある p 要素には適用されません。

実際に見出し画像の下にある「想像力は〜」という文章は p タグ内にありますが、こちらにはスタイルが適用されず、何も変わりません（下の画像参照）

※ 上の赤線と下の赤い部分は両方 p 要素だが、それぞれ別のスタイルが当たっている

要素を限定して指定したい時に役に立つ書き方ですので、他の指定方法と混同しないように（ , カンマ区切りは**指定した全てに適用**です！）注意して使いましょう。

④　導入文（lead クラス）の記述

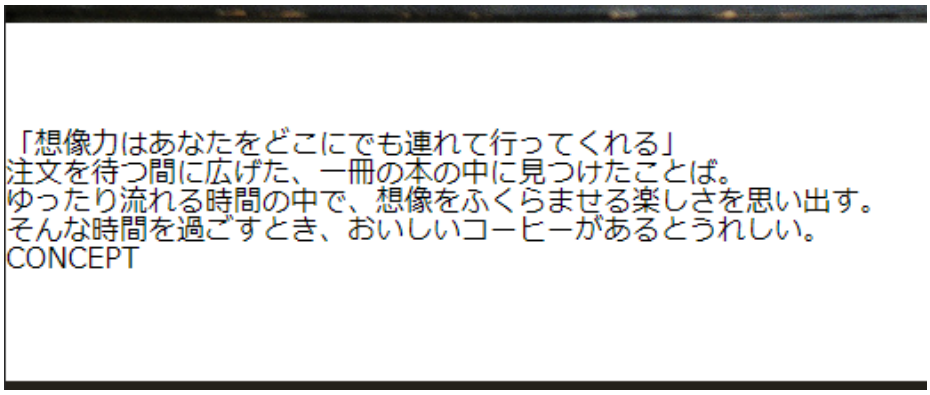
◇導入文のエリア確保

　ボックスモデルを意識して CSS を書く時、大抵はまず配置する場所を確保して記述し始めます。lead クラスというのが導入文の一番外枠のエリアになるので、このクラス名でスタイルを追加し、領域を確保します。

<index.css>

```
.lead {
  max-width: 1200px;
  margin: 60px auto;
}
```

<ブラウザ表示>



　見た目的には上下に 60px の margin が入っただけに見えます。
　ただし max-width を設定しているので、ブラウザの画像サイズが 1200px 以下では lead クラスの領域(width)が自動で縮小されます。

　また margin の指定方法についてですが、今回は値が 2 つなので、それぞれ「上下」と「左右」という指定になります。つまり上下は 60px 固定、左右はブラウザが計算し、自動で中央になるように余白を取ってくれます。

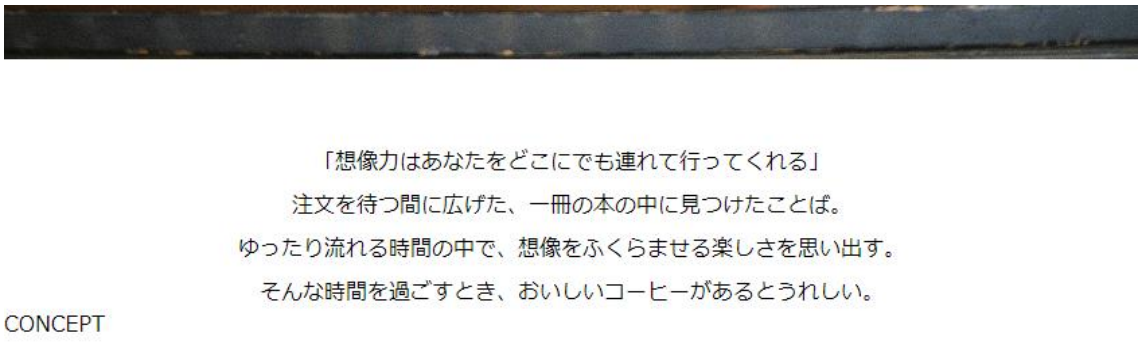
◇導入文のスタイル調整

　文章自体にスタイルを当てますので、次のようなセレクタを追加してください。

<index.css>

```
.lead p {
  line-height: 2;
  text-align: center;
}
```

<ブラウザ表示>



【line-height】プロパティ　default 値は 1.2;

　行自体の高さを設定するプロパティで、**インライン要素では指定できません。**

　私自身、以前詰まった事があったため太字で書きました。デザイナーさんから line-height 指定されたものの、指定された要素がインライン要素（確か a タグ）で全く指定が効かず焦った覚えがあります。

　値には px や%、em や rem といった単位が使えます。

　しかし公式サイトでは**単位無し指定が推奨**されていますので、ここでもそれに倣って単位を付けずに指定しています。

　今回のように長めの文章で余白を付けたい場合に便利なプロパティです。

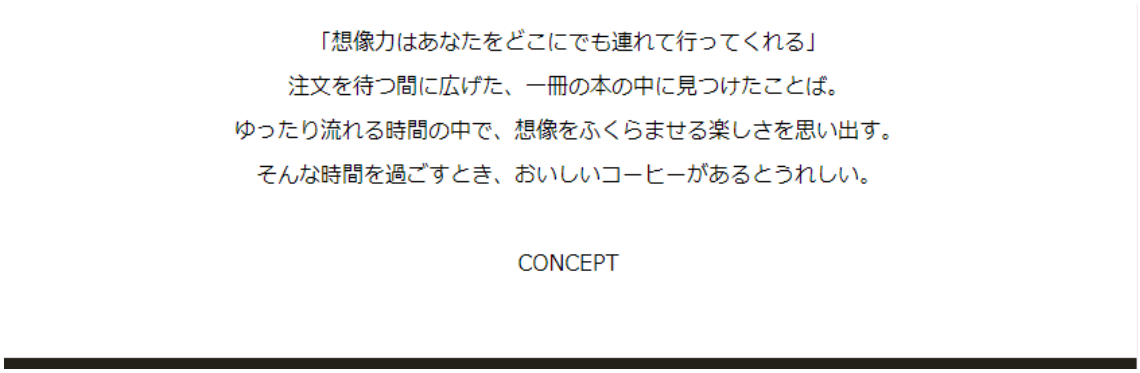
◇リンクボタンエリアの確保

今までと同様、まずは該当部分の領域確保を行います。

<index.css>

```css
.link-button-area {
  text-align: center;
  margin-top: 40px;
}
```

<ブラウザ表示>



text-align については以前解説した通り、非常にクセがありますので注意してください。

CONCEPT という文字を直接囲んでいるのは a タグですのでインライン要素です。ただし text-align は**指定した要素の中にあるインライン要素**の並び方を指定するプロパティです。ですので今回は中央表示させたい要素自体にではなく、その親要素にあたる link-button-area クラスに指定しています。

text-align については WEB フロントエンド 06 の 8 ページで説明していますので、詳しくはそちらをご参照ください。

◇リンクボタンのスタイル調整

エリアの確保は出来たので、中身のスタイルを設定していきます。

<index.css>

```css
.link-button {
  background-color: #f4dd64;
  display: inline-block;
  min-width: 180px;
  line-height: 48px;
}
```

<ブラウザ表示>

そんな時間を過ごすとき、おいしいコーヒーがあるとうれしい。

CONCEPT

display プロパティに「inline-block」という見慣れない値が設定されていますが、これはインライン要素とブロックレベル要素の両方の性質を持った要素になります。
具体的には次のような特徴があります、

・インライン要素と同様、要素の中身で横幅の初期値が決まる
・インライン要素と同様、改行されずに横並びで配置される
・ブロックレベル要素と同様、text-align や line-height などのプロパティが適用される
・ブロックレベル要素と同様、width, height, margin, padding の指定が出来る

詳しい説明が知りたい方は bing チャットで聞くか、下記のサイトを参考にしてください

[インラインブロック要素とは － 意味をわかりやすく － IT 用語辞典 e-Words]

便利な値ですが、使い方を間違えると余計なスタイルばかり増えてしまいます。
仕組みを理解し、必要に応じて使ったほうが良いでしょう。

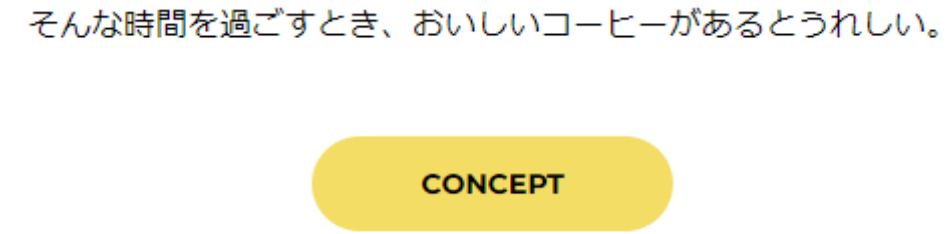
◇リンクボタンのスタイル調整その 2

　見た目を更にボタンっぽくするため、次のスタイルを追加してください。

<index.css>

```css
.link-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>



　ボタンの角が丸くなったのは、border-radius プロパティの影響です。

　radius というのは円の「半径」の事で、設定した値を半径とする円軌道になるように要素の四隅が変形します。数字が大きくなるほど丸くなる、と覚えておけば良いでしょう。

　今回は line-height の丁度半分を設定しているため、両端が完全な半円になっています。

◇リンクボタンにカーソルを乗せた際のスタイルを当てる(:hover)

html には現在の状態を表すタグ、例えば要素にマウスカーソルが乗せられているとか、既に閲覧済みのリンクという事を表すタグはありません。
しかし CSS を使うと、そういったサイトの状態を表す事が出来ます。
それが「**疑似クラス**」です（リセット CSS で出て来た疑似要素とは別物ですので注意）

使い方は簡単です。同じセレクタ名の後ろに「:hover」や「:visited」のように、**：の後に現在の状態を表す疑似クラス名を付けて表記**　すれば OK です。
それでは実際に疑似クラスを付けたセレクタを追加してみましょう！

<index.css>

```
.link-button:hover {
  background-color: #d8b500;
}
```

<ブラウザ表示>

そんな時間を過ごすとき、おいしいコーヒーがあるとうれしい。

カーソルを載せると、色が少し黒っぽく変わったのがわかると思います！

⑤　今回のまとめ

長かったですが、何とかトップページを完成させることが出来ました。
今回もまた色々なスタイルが出て来ましたが、一番難しいのは要素の幅や高さを思い通りに指定出来るになる所だと思います（12 ページあたり）。この辺はもう何度も試行錯誤しながら体で覚えないと、知識だけではすぐに抜けて行ってしまうものです。
大変だと思いますが、やっているうちに絶対に覚えていくものです。
時間を見て、是非スタイルを当てる練習をしてみてください！