コンテンツの制作

「情報II」第3章

Contents

SAMPLE

クラス：　　番号：　　氏名：

コンテンツ制作のプロセスと要件定義

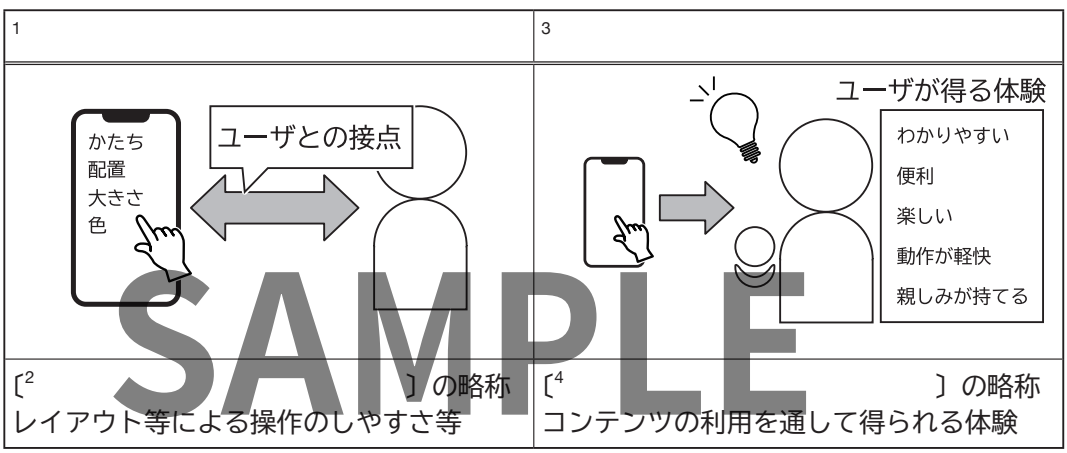
Webサイトなどのコンテンツ（情報の中身）は、表現メディアの特性を上手く組み合わせることによって、より効果的にコミュニケーションの目的を達成させることができます。ここではどのようにしてコンテンツの構成を考えるかについて学びます。

（教科書II：p.28 – p.31）

■ コンテンツ制作のプロセス

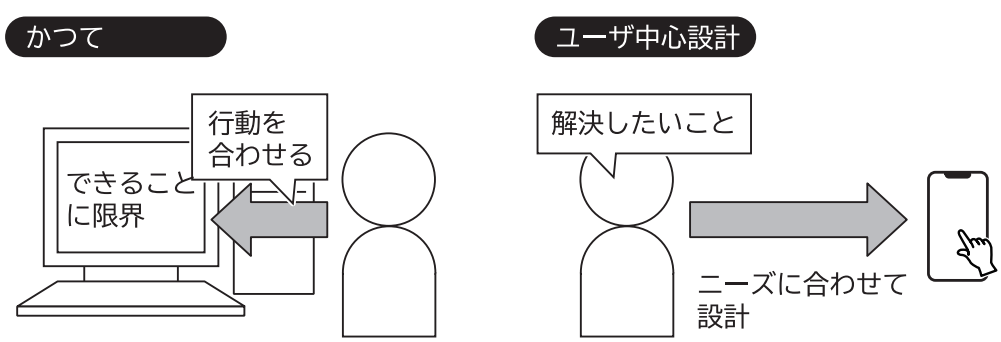
ユーザ視点のコンテンツ制作

近年、人びとが使用する情報機器が多様化（パソコン、スマートフォン、タブレット等）
→いかに内容が優れていても、受け取る側がよいと感じなければよいコンテンツではない



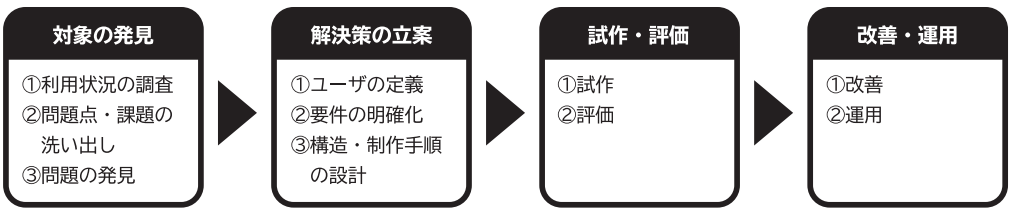
ユーザ中心設計（人間中心設計）

ユーザ中心設計＝ ユーザのニーズや特性を理解し、よりよいUXを目指す設計思考



※かつては、コンピュータにできることに人間が行動を合わせていた
※試作品を作り、ユーザの行動を観察することを繰り返しながらユーザ中心の視点で設計

コンテンツ制作のプロセス



STEP① コンテンツ対象の発見

コンテンツを伝える対象となるユーザの利用状況を調査

→何のための調査か、何を知りたいのかを明確にした上で調査を行う

ユーザの視点に立って現状の問題点や課題を洗い出す

→コンテンツを活用することで解決したい問題を発見する

STEP② 解決策の立案

問題解決の仮説を作成するために、具体的にユーザを定義

→ユーザがコンテンツを利用するうえでの要件を明確化

コンテンツの要件をもとに、コンテンツの構造や制作手順を設計

→コンテンツの構造や手順を具体的に設計

STEP③ 制作・評価

設計に従って、 コンテンツの制作を行う

試作品（プロトタイプ）をつくる

→ユーザ視点によるコンテンツの優先順位、デザインの確認

STEP④ 改善・運用

評価により明らかになった問題点を改善し、コンテンツの運用を行う

評価・改善の一連の作業を繰り返すことで、ユーザによりよいコンテンツを提供

■ 解決策立案のための仮説生成

解決策立案のための仮説生成

問題の解決策を立案するためには、ユーザが切実に解決して欲しいと思う問題点を整理
→問題解決の仮説を作成
ユーザを取り巻く環境や人物像を仮想的ながらも具体的に記述すること
→ユーザが本当に求めていることが明らかになっていく

ペルソナ法

ペルソナ法＝ 具体的なユーザ像を設定することでユーザのニーズや特性を理解する手法

※漠然としたイメージだけでなく、実際に氏名を設定し、可能な限り詳しく属性を定義
→性別、年齢、住所、職業、趣味、好きなもの、家族構成、行動の背景となる要因など

実習1

高校生を想定して、ペルソナを具体的に設定してみよう

<table>
<tr><td rowspan="4">顔・容姿</td><td>氏名</td></tr>
<tr><td>性別・年齢</td></tr>
<tr><td>居住地</td></tr>
<tr><td>家族構成</td></tr>
<tr><td colspan="2">性格・価値観など</td></tr>
<tr><td colspan="2">学校生活（学習状況や進路希望など）</td></tr>
<tr><td colspan="2">好きなもの・趣味など</td></tr>
<tr><td colspan="2">アクティビティ（部活動やその他取り組んでいる活動など）</td></tr>
</table>

実習2

［実習1］で作成したペルソナが抱える学校生活上のニーズを列挙してみよう

ここで挙げた学校生活上のニーズをグループでシェアしよう
→出し合ったニーズの中で、アプリを作ることで解決できそうなことはどれだろう？
→どのようなアプリを作るとよいかを導こう

要件定義

要件定義＝ ユーザの問題を解決するために必要となる情報や機能が何かを明らかにする

実習3

考えたアプリに必要な情報や機能をリストアップしよう

振り返り

次の各観点が達成されていれば□を塗りつぶしましょう。
□コンテンツ制作のプロセスについて理解できた
□ペルソナを詳細に設定することができた
□ペルソナを設定することにより、ユーザのニーズを引き出すことができた
□要件定義を考えることができた

今日の授業を受けて思ったこと、感じたこと、新たに学んだことなどを書いてください。

HTML と CSS

Webサイトは、複数のWebページから構成されたWebページの集合体のことをいいます。一つ一つのWebページは、HTMLとCSSによってつくられています。ここでは、HTMLとCSSでどのようにWebページがつくられているかについて学びます。

（教科書II：p.36 , 教科書I：p.107 – p.109）

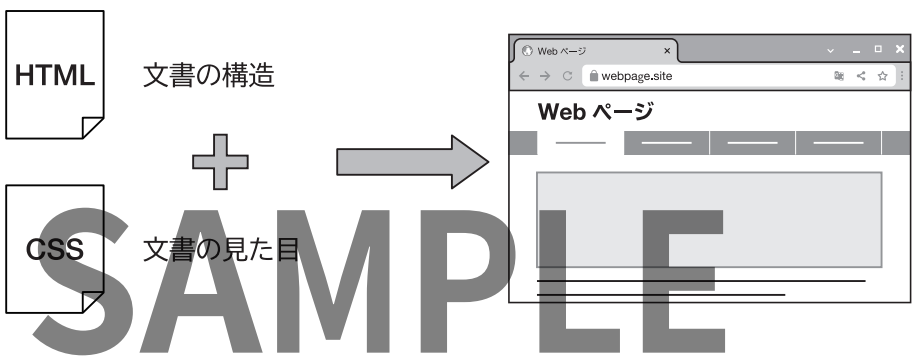
■ HTMLとCSS

Webページをつくる言語

WebページはHTMLとCSSによって構成されている

→HTMLが文書構造を、CSSが見た目やスタイルを指定する

Webページに動きをつけるには、更にJavaScriptという言語を使う



※HTML ＝ Hyper Text Markup Language

※CSS ＝ Cascading Style Sheets

Markup Language

HTML ＝ Hyper Text Markup Language

Hyper Text ＝ 別の文書に移動するしくみを備えた文書のこと

Language ＝ コンピュータに指示するための言語

では、Markupとは？

markupの語源は、marking upという出版業界の用語がもとであった

marking up ＝ 原稿用紙の余白に印刷に関する指示記号を描き加えること

→コンピュータに書体や文字サイズなどの書式情報を指示する言葉に転じた

→Markup Language ＝ コンピュータで書式を指示する言語という意味に

HTMLの基本

タグ

```
<タグ名>コンテンツ</タグ名>
```
（<タグ名>：開始タグ　</タグ名>：終了タグ）

上記のように開始タグ <タグ名> と終了タグ </タグ名> で挟む
→内容に文書の構造や制御情報を埋め込むことができる
※画像を表示させる <img> タグのように、終了タグのない単体使用のタグも存在する

例）ブラウザのタイトルバーにページのタイトルを表示させるタグ

```
<title>ページタイトル</title>
```

タグには、**属性**を追加することで様々な設定を追加することができるものもある

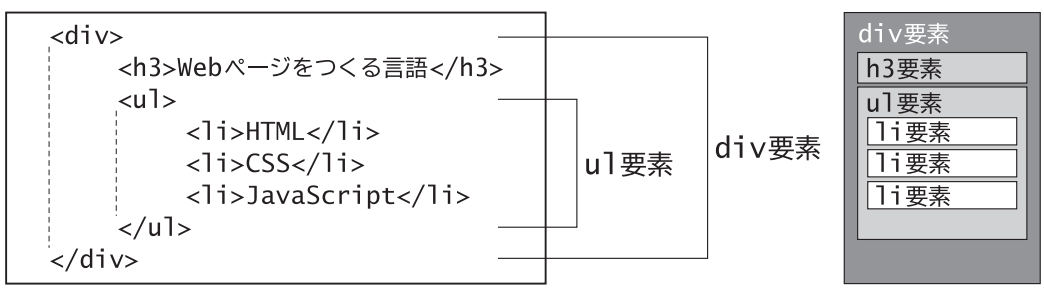
```
<a href="betsu.html">別のページへ</a>
```
（href：属性名　"betsu.html"：属性の値）

上記の例は、「別のページへ」という文字列をクリックすると"betsu.html"へ移動
→移動先のWebページを指定するためにhref属性をaタグに追加

HTMLの入れ子構造

大きなものの中に小さなものが入っているような構造を「入れ子」構造という
開始タグと終了タグはしっかりと対応させておかなければならない
入れ子構造がわかりやすいように、字下げを使うと便利（字下げは使わなくてもよい）

```
<div>
    <h3>Webページをつくる言語</h3>
    <ul>
        <li>HTML</li>
        <li>CSS</li>
        <li>JavaScript</li>
    </ul>
</div>
```



※段落タグpで見出しタグh1 ～ h6 を囲むことはNGのように入れ子のルールも存在
　→ルールをすべて覚えることは難しいが、意味がわかってくれば判断できるようになる
開始タグと終了タグの順番をクロスさせてはならない

```
<p>段落内に<strong>強調文字</p>を入れたい</strong>
```

開始タグと終了タグがクロスしている → **NG**

CSSの基本

CSSの基本書式

```
セレクタ {
    プロパティ : 値;
    プロパティ : 値;
    ........
}
```

セレクタ	HTMLのどの要素にスタイルを適用させるかを指定する
プロパティ	サイズや色など、設定するスタイルの種類
値	プロパティに適用する数値（プロパティによって指定の仕方は異なる）

例）pタグ（段落）の文字サイズを14px、文字色を#0000FF（青色）を指定したい

```
p {
    font-size: 14px;
    color: #0000FF;
}
```

※行末の;を忘れないように気をつけよう

よく使う単位

px（ピクセル）	モニタの1ピクセルを1pxとする絶対値の単位
%（パーセント）	ブラウザのデフォルトサイズを100%としたときの割合
em（エム）	適用する要素の大文字Mのフォントサイズを1とする相対値
rem（ルートエム）	ブラウザのデフォルトのMのフォントサイズを1とする相対値

クラス

同じタグを複数箇所で使用するが、スタイルをそれぞれ変更したい場合、クラスを指定

HTML側では、

```
<p class="red">ここは赤い段落</p>
```

CSS側では、クラス名の先頭に.を付けてセレクタとして指定

```
.red {
    color: #FF0000;
}
```

■ Webページのレイアウト

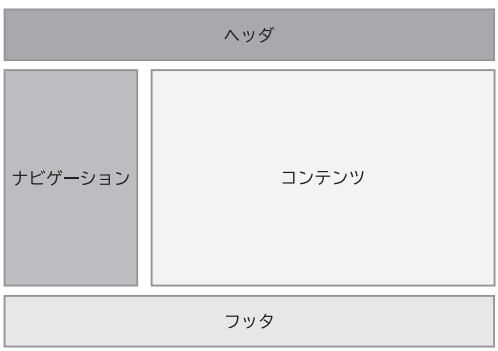
Webページのレイアウト

Webページのコンテンツをつくる前に、ページのレイアウトを検討する

→各ページに表示するメニュー（ナビゲーション）とコンテンツの位置関係を考える

例

ここでは、例として次のようなレイアウトのWebページをつくってみよう



Webページのつくり方

TextアプリでHTML文書を作成し、ファイル名の拡張子に「.html」を指定すればよい

例題1

基本的なWebページの構造をつくろう（ファイル名：webpage.html）

```
 1 <!DOCTYPE html>                              HTML文書と宣言
 2 <html>                                       HTML文書開始
 3     <head>                                   設定等を書くところ
 4         <meta charset="utf-8">               文字コードをUTF-8に
 5         <title>Webページのタイトル</title>
 6     </head>                                  設定等ここまで
 7     <body>                                   HTML文書本体ここから
 8         ここに書いたものがブラウザに表示される
 9     </body>                                  HTML文書本体ここまで
10 </html>                                      HTML文書終了
```

Webページの表示

ファイルアプリで［例題1］で保存した「webpage.html」を開く

→ブラウザにページが表示される

※書き換える度に、再読み込みすることで、最新のものを表示できる

ブロック要素

divタグを使うと、文書をいくつかのブロックに区分することができる

→どのようなブロックであるかを、class属性を使って指定する

```
<div class="クラス名">コンテンツ</div>
```

例題2

前ページの4つのレイアウトブロックをdivタグを使って区分してみよう

```
 8          <div class="header">
 9              ヘッダ部分
10          </div>
11          <div class="navigation">
12              左のナビゲーション部分
13          </div>
14          <div class="main">
15              メインコンテンツ部分
16          </div>
17          <div class="footer">
18              フッタ部分
19          </div>
```


CSSの作成

HTML文書同様、CSSもTextアプリにて作成、拡張子に「.css」を指定する

HTML文書内でのCSSの指定

HTML文書内のhead要素内にlinkタグでCSSのファイル名を指定する

```
<link rel="stylesheet" href="style.css" type="text/css">
```

※href属性で指定したファイル名でCSSファイルをつくる必要がある

→今回の場合であれば「style.css」

例題3

「webpage.html」内にCSSへのリンクを設定しよう

```
 6      <link rel="stylesheet" href="style.css"
                                        type="text/css">
```

※これをしても、現時点では画面上の変化はなにもない

例題4

「style.css」ファイルを作成しよう

```
body {                              表示される領域全体を設定
    background-color: #ffffff;      背景色を白に
    color: #000000;                 文字色を黒に
    margin-right: auto;             右側の余白を自動で設定
    margin-left: auto;              左側の余白を自動で設定
    width: 80%;                     幅をブラウザの幅の80%に
    font-size: 14px;                フォントサイズを14pxに
}

.header {                           headerクラスを設定
    background-color: #0000ff;      背景色を青色に
    color: #ffffff;                 文字色を白に
    height: 90px;                   高さを90pxに
    font-size: 36px;                フォントサイズを36pxに
    line-height: 90px;              1行の高さを90pxに→上下中央に
    padding: 20px;                  囲み内の余白を20pxに
}

.navigation {                       navigationクラスを設定
    background-color: #ffffff;      背景色を白に
    color: #000000;                 文字色を黒に
    width: 25%;                     幅を現在の横幅の25%に
    float: left;                    別の要素の左側に配置
}

.main {                             mainクラスを設定
    background-color: #ffffff;      背景色を明るいグレーに
    color: #000000;                 文字色を黒に
    width: 70%;                     幅を現在の横幅の70%に
    float: right;                   別の要素の右側に配置
}

.footer {                           footerクラスを設定
    background-color: #aaaaaa;      背景色をグレーに
    color: #000000;                 文字色を黒に
    height: 50px;                   高さを50pxに
    clear: both;                    floatの配置を解除する
}
```

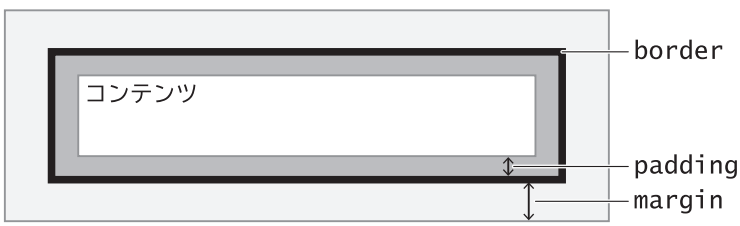
例題5

ナビゲーションメニュー側のレイアウトをしていこう

「webpage.html」の「navigation」ブロックにメニューを並べてみよう

```
        <div class="navigation">
            <div class="left-menu-head">コンテンツ</div>
            <div class="left-menu">メニュー 1</div>
            <div class="left-menu">メニュー 2</div>
            <div class="left-menu">メニュー 3</div>
            <div class="left-menu">メニュー 4</div>
        </div>
```

要素の余白の考え方

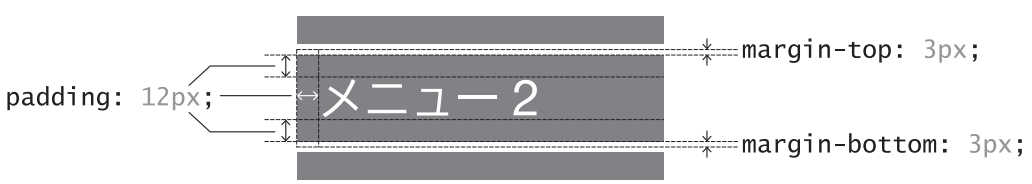


例題6

「style.css」に.left-menu-headと.left-menuセレクタを追加しよう

```
.left-menu {
    background-color: #0000ff;   背景色を青色に
    color: #ffffff;              文字色を白に
    margin-top: 3px;             要素の上に3pxの余白を空ける
    margin-bottom: 3px;          要素の下に3pxの余白を空ける
    padding: 12px;               要素の内側に12pxの余白を空ける
    font-size: 18px;             文字サイズを18pxに
}

.left-menu-head {
    background-color: #aa4400;   背景色を赤色に
    color: #ffffff;              文字色を白に
    padding: 6px;                要素の内側に6pxの余白を空ける
    font-size: 18px;             文字サイズを18pxに
}
```



メインコンテンツ部分の作成

メインコンテンツ部分は、構造化されたテキストで記述する必要がある
見出しは、レベル1がh1、レベル2がh2という具合に、レベル6のh6まで指定可能
段落はpで指定する
箇条書きは箇条書きのひとまとまりをulで指定し、箇条書きの要素をliで並べる

例題7

メインコンテンツをつくってみよう

```
19          <div class="main">
20              <h1>見出しレベル1</h1>
21              <h2>見出しレベル2</h2>
22              <p>ここが本文。</p>
23              <ul>
24                  <li>箇条書き1</li>
25                  <li>箇条書き2</li>
26                  <li>箇条書き3</li>
27              </ul>
28              <h2>見出しレベル2</h2>
29              <p>ここが本文。</p>
30          </div>
```


課題

CSSで見出しや本文のスタイルをいろいろと設定してみよう

振り返り

次の各観点が達成されていれば□を塗りつぶしましょう。
□WebページがHTMLとCSSを使って表示されていることを理解した
□HTMLタグの基本的な書き方や入れ子構造を意識しながらHTMLを書くことができた
□CSSのしくみを理解し、CSSを使ってさまざまなスタイルの設定をすることができた
□CSSにおける要素の余白の考え方を理解することができた

今日の授業を受けて思ったこと、感じたこと、新たに学んだことなどを書いてください。

Web サイトの構造

HTMLは、文書と文書を相互に関連付けることで、相互に行き来することができるのが特徴です。このしくみを用いて、いくつかのWebページをまとめたものをWebサイトといいます。ここでは、ページ間を相互に行き来するしくみを作ります。

（教科書II：p.36 , 教科書I：p.106）

■ ファイル管理

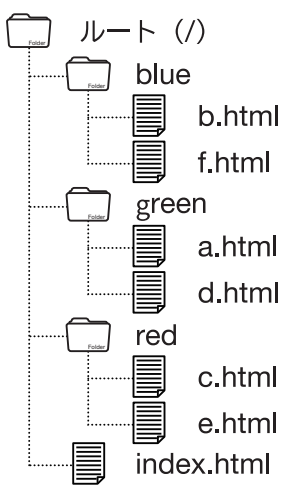
ファイルとフォルダ

フォルダ＝ ファイルを分類・整理しておくための保存場所

※フォルダの中には更にフォルダを作成し、階層的に管理することができる
→箱の中に箱を入れ、その箱の中にファイルが格納されているイメージ

ツリー構造

フォルダとファイルは次のようなツリー構造で表現
フォルダの中に入るときは、"/"で区切って表現
最上位をルート（根）といい、"/"で表す



パス

ファイルまでの経路をパス(path)という

例）`index.html`から見て`a.html`は、

1

例）`a.html`から見て`b.html`は、

2

経路を一つ前のフォルダへ戻る場合は、`..`と指定する

問題1

次のそれぞれの問のパスはどのように表されるでしょうか？

(1)	b.html → c.html	3
(2)	c.html → d.html	4
(3)	d.html → e.html	5
(4)	e.html → f.html	6
(5)	f.html → index.html	7

実習1

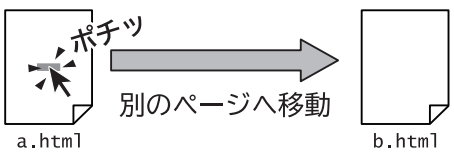
Google Classroomから「パスのしくみ.zip」をダウンロードしよう
「パスのしくみ.zip」の中にあるフォルダを「ダウンロード」フォルダへ移動させる
フォルダを開き、「パスのしくみ」内のファイルとフォルダの構成を下に書こう
ただし、「image」フォルダは省略する

8

■ ハイパーリンク

ハイパーリンク

ハイパーリンク＝ 関連する文書同士をつなげ、別の文書へ移動できるようにするしくみ



※Webサイトは、サイト内のWebページ同士を相互にハイパーリンクでつないだもの
※ハイパーリンクで相互につながる文書をハイパーテキストという

アンカータグ

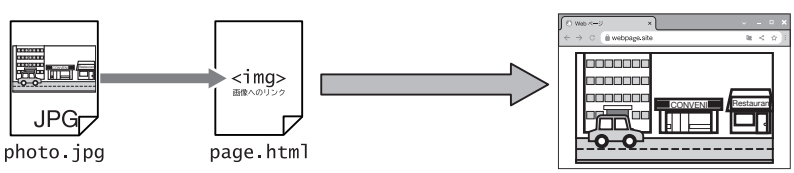
ページにハイパーリンクを設定するには、アンカータグ（a）を使う

```
<a href="リンク先のパスまたはURL">ハイパーリンクの要素</a>
```

※href属性の値にリンク先のパスかURLを指定するとハイパーリンクを設定できる

画像の挿入

Webページに画像を表示させるのは、ページに直接画像を書き込んでいるわけではない
ハイパーリンクと同様のしくみで画像ファイルの内容をページの上に載せて表示



ページに画像を表示させるにはimgタグを使う

```
<img src="画像のリンク先のパスまたはURL" alt="画像の説明">
```

※src属性の値に画像のリンク先のパスかURLを指定すると画像を表示させられる
※alt属性には、画像が表示されない場合や、視覚障害者のために画像の説明を入れる

実習2

［実習1］でダウンロードした「パスのしくみ」内にあるWebページを相互につなごう
次のようにアルファベット順に接続し、最後にindex.htmlへ戻ってくるようにしよう

```
index.html → a.html → b.html → … → l.html → index.html
```

また、各ページにはそのページに対応した画像を表示させよう

```
a.html → a.jpg , b.html → b.jpg , ……
```

絶対パスと相対パス

本節で学んだパスは、正式には**相対パス**と呼ばれるもの
→自身のファイルから見て、相手のファイルがどこにあるかを記述する

ルート（/）を基準にファイルのパスを記述する方法を**絶対パス**という
例えば、p.13の例では、どのファイルからでもa.htmlの絶対パスは

```
/green/a.html
```


振り返り

次の各観点が達成されていれば□を塗りつぶしましょう。
□フォルダを使ったファイル管理の方法を理解した
□ファイルのパスの書き方を理解できた
□パスの考え方を使ってハイパーリンクをつくることができた

今日の授業を受けて思ったこと、感じたこと、新たに学んだことなどを書いてください。

プロトタイピング

実際にWebサイトを制作する前に、画面の試作品（プロトタイプ）を作成し、使い勝手を検証することをプロトタイピングといいます。今日は、1時間目に考えた要件定義から、情報や機能同士をどのように配置し、接続するかを考えてみましょう。

（教科書II：p.40 – p.43）

■ プロトタイピング

プロトタイピング

プロトタイピング＝ 試作品を作成し、事前検証を行なうことで開発効率を高める手法

プロトタイピングツール（Figma）

Figma＝ リアルタイム協同編集にフォーカスされたプロトタイピングツール

※スマートフォンではプロトタイプを編集することはできないので注意

※スマートフォンからは、プロトタイプの閲覧と、実際に動かしてみることはできる

Figmaの起動

次のURLでプロトタイピングツール「Figma」を起動する

```
https://www.figma.com
```

ユーザ登録

画面右上の「無料で始める」からアカウントを作成する

→「Googleで続行」をすることで、学校のGoogleアカウントで利用が可能に

ファイルの新規作成

画面右上の「**＋新規作成**」から「**デザインファイル**」を選ぶと新しいファイルが作成

→この際、自分の「**Team project**」にデザインファイルを作成するようにしよう

共同編集

画面右上の「**共有**」を開き、共有するメンバーのメールアドレスを入力

権限を「**編集可**」に設定し、「**招待**」ボタンを押す

ページデザインの作成

Figmaの画面構成

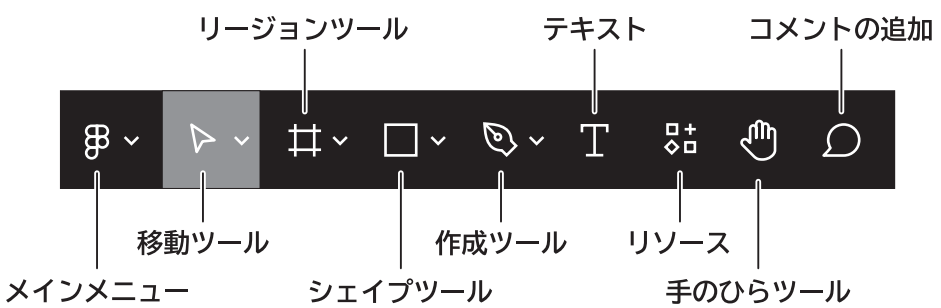
フレームの作成

ツールバーの**リージョンツール**#から、「**#フレーム**」を選ぶ

→画面右の詳細設定から、端末の名前を選択すると、デザイン画面にフレームが追加

図形 / 文字 / コンポーネントの追加

ツールバーのそれぞれの役割は、以下の通り



□	シェイプツール	デザインに四角や丸など、図形を挿入する
T	テキスト	画面に文字を配置する
	リソース	さまざまなデザインパーツをデザインに挿入することができる

※それぞれの図形等の設定は、画面の詳細設定で設定することができる

※リソース内の**コンポーネント**にあるデザインパーツはOS等に共通のパーツ

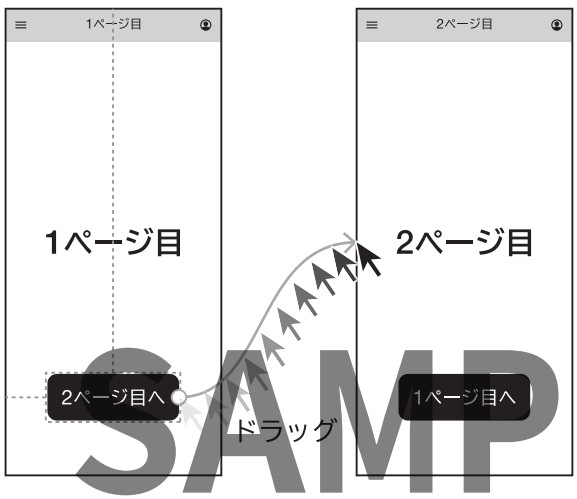
プロトタイプの作成

インタラクションの作成

画面右カラムの詳細設定で**プロトタイプ**に切り換える
→インタラクションを作成できるようになる

次に移動（ページの切り換え）

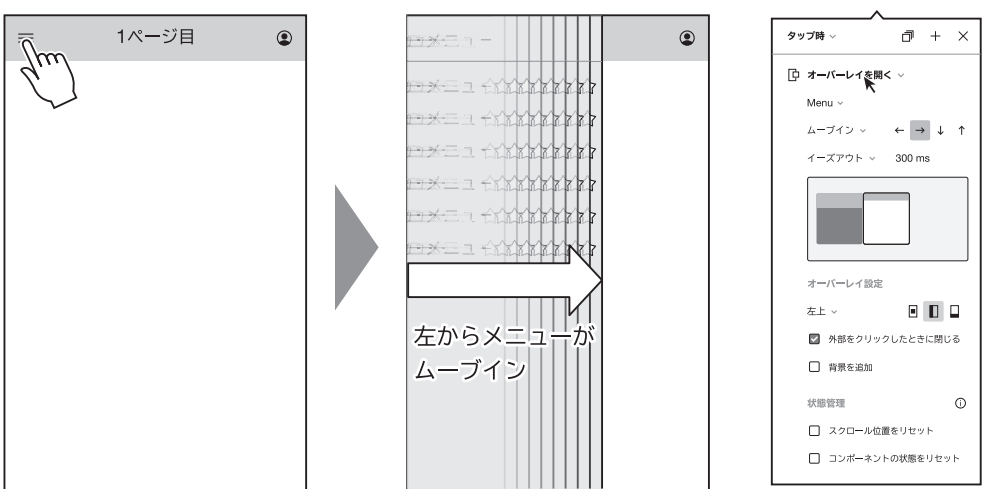
ハイパーリンクを設定するオブジェクトを選択し、
四辺に出てくる⊕をドラッグ
→リンク先のページまで矢印をのばしていく



※矢印をクリックすると、ページが切り換り方などを設定することもできる

オーバーレイ（ページの上に重ねる）

メニュー≡をタップすると、現在の画面の上にメニューが重なるような場合
→オーバーレイという手法を用いると実現できる



課題

p.04［実習3］で考えたアプリのプロトタイプをつくってみよう
グループで協力して分担しながら作成していこう
※このアプリには、ログイン画面、ユーザ登録のプロセスも必ず含めること

ヒント

コンポーネントにあるデザインパーツを積極的に使っていこう
※デザインパーツはいくつもグループ化されていることが多い
→プロトタイプ作成の際は注意が必要
→ダブルクリックしていくことでオブジェクトの階層を絞ることができる
→リンクを設定したいオブジェクトを選択できるまでダブルクリックして掘り下げよう

振り返り

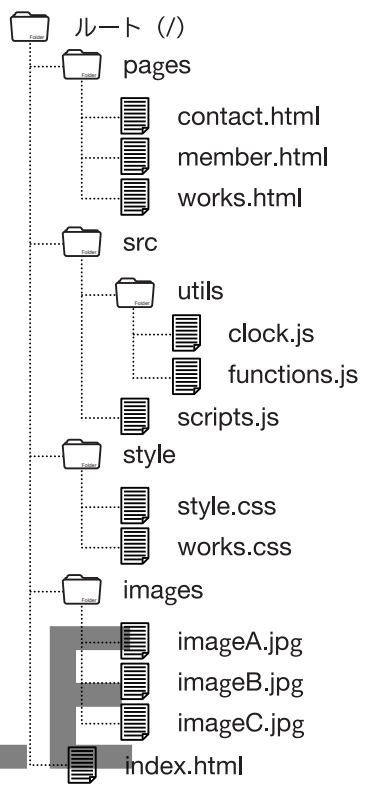
次の各観点が達成されていれば□を塗りつぶしましょう。
□プロトタイピングツールFigmaを登録し、使えるようになった
□Webページ（アプリ）の画面を設計することができた
□プロトタイピングで画面遷移を設計することができた

今日の授業を受けて思ったこと、感じたこと、新たに学んだことなどを書いてください。

章末問題

[問題]

右のようなファイルの階層があるとき、あるファイルからあるファイルを参照するときのパスを書いてください。



（1）index.html → contact.html

（2）contact.html → works.html

（3）works.html → style.css

（4）works.html → imageA.jpg

（5）index.html → scripts.js

（6）index.html → functions.js

（7）works.html → functions.js

（8）functions.js → index.html

（9）functions.js → works.html

（10）functions.js → clock.js

コラム～レスポンシブデザイン

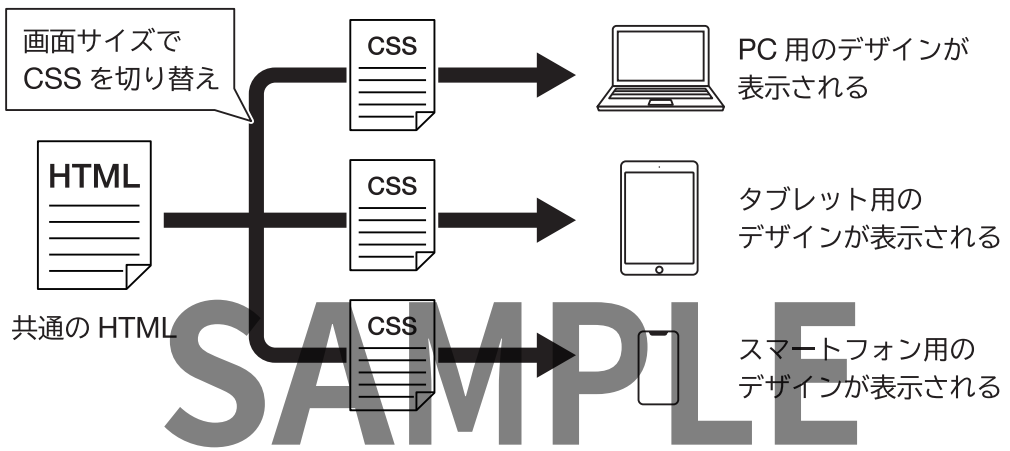
■ レスポンシブデザイン

レスポンシブデザイン

レスポンシブデザイン＝ 閲覧ユーザーの画面サイズに合わせてレイアウトを最適化

※Webサイトをどのようなデバイスでアクセスするかわからない
→それぞれのデバイスで最適な表示方法をさせる考え方

※1つのHTMLだけで、CSSを切り換えることにより見せ方を変える考え方



1つのページのHTML・URLだけを管理するだけでよくなるためたいへん効率的！

レスポンシブデザインのデメリット

<u>モバイル版の表示に時間がかかってしまう</u>

1つのHTML・URLで異なる見せ方をするデザインである
→モバイル版のページでは表示させない情報も読み込むことになる
→ページによっては全てのコンテンツが表示されるまでに時間がかかってしまう場合あり

<u>表示するデザインが似通ってしまう</u>

形や大きさは画面サイズに最適化されるが、画像やフォントそのものは変えられない
→それぞれのデバイスで表示されるデザインが似通ってしまうデメリットがある

PC版ページとモバイル版ページでそれぞれに最適なデザインを個別に制作したいとき
→それぞれのページを個別に制作するほうがよい（ただし更新は非効率化する）

SAMPLE

2025年度「情報Ⅱ」ワークブック見本
CQ出版社