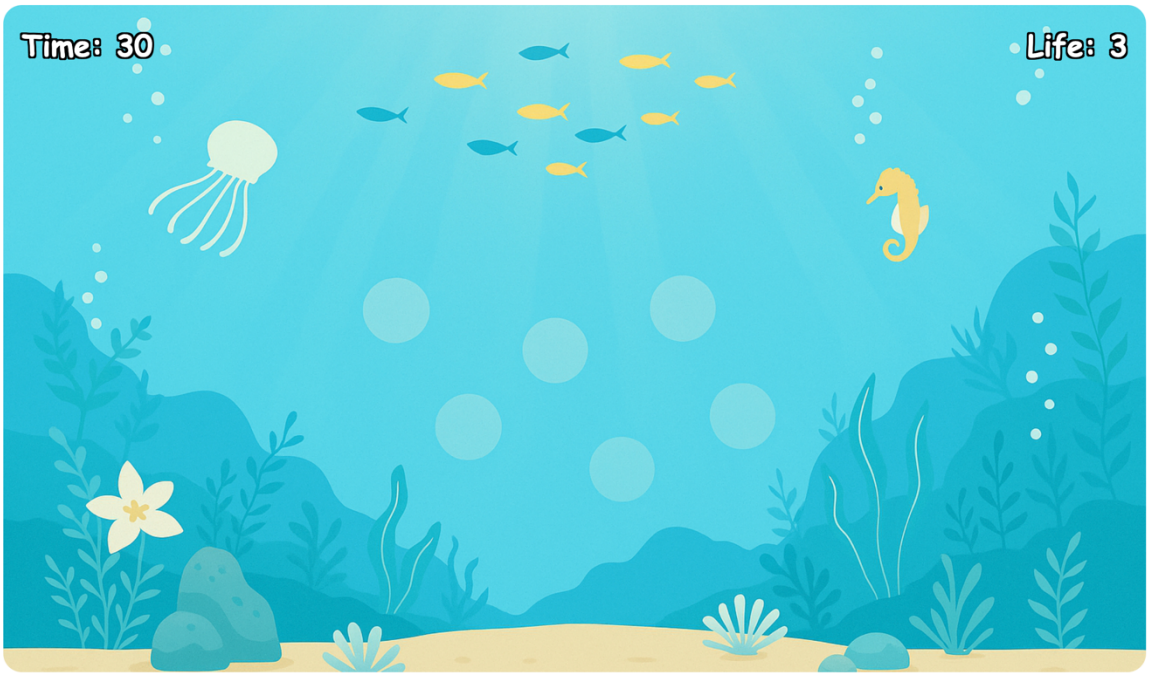
プログラミング基礎演習 / 説明資料 / G2 / screen

目標：

ゲームの画面（スクリーン）として、「スタート画面」「プレイ画面」「ゲームオーバー画面」「クリア画面」の 4 つの画面を作成します。



準備：

```
g2/
└ 4_screen/
   └ step1/
     ├ css/      ・・・ CSS ファイルを格納
     ├ js/       ・・・ JavaScript ファイルを格納
     └ img/      ・・・ 画像ファイルを格納
```

開発手順：

Step1：　HTML で各画面の文章を作成する

Step2：　CSS で画面を装飾する 1（共通部）

Step3：　CSS で画面を装飾する 2（見出し、制限時間、ライフ）

Step4：　JS で画面切替機能を実装する

Step5：　ブラッシュアップ

G2-4-Step1：HTML で各画面の文章を作成する

スタート画面（#start）、プレイ画面（#play）、ゲームオーバー画面（#end）、クリア画面（#clear）を 1 つの HTML に記述します。各画面のメッセージは、自由に書いて構いません。以下はサンプルです。ただし、HTML タグと id と class は、CSS の装飾指定や DOM 操作を行うため、必ず、以下の通り指定してください。

HTML / <body> 変更

```
---略---
<body>
  <div id="start" class="screen active">
    <h1>Marimo Game</h1>
    <p>マリモを操作してボールを避けよう！</p>
    <p class="next">スペースキーを押してゲームを開始してください</p>
  </div>
  <div id="play" class="screen active">
    <div id="time">Time: 30</div>
    <div id="life">Life: 3</div>
    <div id="marimo"></div>
    <div id="ball"></div>
  </div>
  <div id="end" class="screen active">
    <h1>Game Over</h1>
    <p class="next">スペースキーを押すとゲームの開始画面に戻ります</p>
  </div>
  <div id="clear" class="screen active">
    <h1>Clear!</h1>
    <p>おめでとうございます！ボールを避け切りました！</p>
    <p class="next">スペースキーを押すとゲームの開始画面に戻ります</p>
  </div>
</body>
---略---
```

結果：

4 つの画面が縦並びに表示され、各画面内に文章が表示されたら成功です。

解説：

1）`class="screen active"` について

以前に演習した「プレゼントボックス」と同様に、class を 2 つ指定しています。

`screen` はすべての画面に同じ CSS を適用するため、`active` は有効にしたい画面のみ付与する予定です。

現時点では、CSS で装飾を行うため、すべての画面に `active` を付与して開いた状態にしています。
最終的には、JS から表示したい画面のみに `active` を付与します。

G2-4-Step2：CSS で画面を装飾する 1（共通部）

画面の基礎を作ります。4 つすべての画面が同じ指定になるため、「.screen」に対して記述します。
以下のように CSS（全文）を記述してください。

CSS / 全文 / 変更

```css
body {
  background-color: #fff;
}

.screen {
  position: relative;
  margin: 50px auto;
  width: 80vw;
  height: 80vh;
  background-color: #fff;
  border-radius: 20px;
  background: url('../img/bg.png') center/cover no-repeat;

  display: none;
}


.screen.active {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}
```

結果：

4 つの画面に背景が表示され、各画面に文章が上下中央寄せで配置されたら成功です。

解説：

セレクタ	内容
background:	background プロパティをまとめて書いた ショートハンド（省略形） です。

	background-image: url('../img/bg.png'); => 画面に表示したい画像ファイルを指定 '../img/bg.png' は「一つ上のフォルダにある img/bg.png」という意味 background-position: center; => 背景画像を横・縦の両方で中央に配置する background-size: cover; => 要素の大きさに合わせて「はみ出してもいいから画面を隙間なく埋める」 background-repeat: no-repeat; => 画像をタイル状に並べず、1 枚だけ表示する
.screen	各画面の共通になる背景などを指定しています。 display:none; を指定して非表示にしています。これは、実際のゲームを動かす際に、常に 4 つの画面を出すのではなく、1 つの画面だけを表示するためです。
.screen.active	.screen かつ.active の場合に適用されます。 display:flex;を指定することで表示状態にしています。 現時点では、すべて付与することで全画面を確認できるようにしています。

G2-4-Step3：CSS で画面を装飾する 2（タイトル、制限時間、ライフ）

タイトル（h1）、制限時間（#time）、ライフ（#life）の装飾を行います。
以下の通り、CSS の最下部に追加してください。

CSS / 最下部 / 追加

```
h1 {
  font-family: 'Comic Sans MS', sans-serif;
  padding: 20px;
  font-size: 3em;
  font-weight: bold;
  color: white;
  -webkit-text-stroke: 2px black;
}

#time {
  position: absolute;
  top: 20px;
  left: 20px;
  font-family: 'Comic Sans MS', sans-serif;
  font-size: 2em;
```

```
  font-weight: bold;
  color: white;
  -webkit-text-stroke: 2px black;
  z-index: 100;
}

#life {
  position: absolute;
  top: 20px;
  right: 20px;
  font-family: 'Comic Sans MS', sans-serif;
  font-size: 2em;
  font-weight: bold;
  color: white;
  -webkit-text-stroke: 2px black;
  z-index: 100;
}
```

結果：

プレイ画面の左上に「Time: 30」、右上の「Life: 3」と表示されたら成功です。

解説：

プロパティ	解説
font-family:	フォント（文字の見た目）を選ぶプロパティ 最初の 'Comic Sans MS' が指定フォント、そのフォントが使えないときの 予備として sans-serif を使います。いくつでも指定ができます。 左から順にフォントが指定されます。
-webkit-text-stroke:	文字の外側に線（ストローク）をつけるプロパティ 線の太さ 2px で色は黒に指定しています。 -webkit-*** は、標準化されていないが、利用されています。
z-index:	どの要素を前面に表示するかを決めるプロパティ 数字が大きいほど前面（一番上）に表示されます。 未指定の場合は、auto となり、HTML の書かれた順で重なりが決まります。今回、100 にしていますが、50 でも 999 でも構いません。他の要素に隠れないよう確実に指定することが目的です。

G2-4-Step4：JS で画面切替機能の実装する

JS で、画面を切替える機能を実装します。以下のコードを app.js に記述してください。

JS / 新規

```
// 現在の画面を格納する変数
let mode; // start, play, end, clear

// 各画面の DOM 取得
const startScreen = document.querySelector("#start");
const playScreen = document.querySelector("#play");
const endScreen = document.querySelector("#end");
const clearScreen = document.querySelector("#clear");

// 画面切替
function showScreen(screenName) {

  // すべての画面から active クラスを除去
  startScreen.classList.remove("active");
  playScreen.classList.remove("active");
  endScreen.classList.remove("active");
  clearScreen.classList.remove("active");

  // 指定された画面だけ active クラスを追加
  if (screenName === "start") {
    startScreen.classList.add("active");
  } else if (screenName === "play") {
    playScreen.classList.add("active");
  } else if (screenName === "end") {
    endScreen.classList.add("active");
  } else if (screenName === "clear") {
    clearScreen.classList.add("active");
  }

  // モードを更新
  mode = screenName;
}
```

```
// 初期画面を表示
//showScreen("start");
showScreen("play");
//showScreen("end");
//showScreen("clear");
```

結果：

`showScreen("***");` の `***` を、`start / play / end / clear` を指定して、
それぞれの画面が表示されたら成功です。

解説：

コード	解説
`.classList.remove("active");`	各画面要素に class=”active”がある場合は、削除しています。 これにより一度すべての画面を非表示状態にします。
`.classList.add("active");`	指定された画面要素のみに class=”active”を付与します。
`mode = screenName;`	現在表示中の画面が何かわかるように、mode 変数にセットしています。現時点では利用しませんが、「ゲームオーバー中はスペースキーを押したら・・・」などの判定に使う予定です。

G2-4-Step5：ブラッシュアップ

・「スペースキーを押す」メッセージを目立たせるようにする
・フォントを変更する
・背景画像を自作して画面イメージを変える
・Time や Life の表示位置、サイズなどを変更する
etc…

例）「スペースキーを押す」メッセージを目立たせるようにする

CSS / 最下部 / 追加

```css
p.next {
  color: #e74c3c;
  font-weight: bold;
  animation: blink 1.5s ease-in-out infinite;
}

@keyframes blink {
  0%, 100% {
    opacity: 1;
  }
  50% {
    opacity: 0.4;
  }
}
```

G2-4-Step6：画面サイズを取得する

各画面のサイズを取得するコードを記述します。app.js に以下のコードを記述してください。

1）変数に画面サイズの幅と高さの変数を宣言します（黄色箇所）

JS / 変数 / 追加

```
// 現在の画面を格納する変数
let mode; // start, play, end, clear
let maxX; // 画面サイズ(幅)
let maxY; // 画面サイズ(高さ)
```

2）各画面を active した後に画面サイズを取得するコードを記述します（黄色箇所）

JS / showScreen() / 追加

```
function showScreen(screenName) {
・・・略・・・
  // 指定された画面だけ active クラスを追加
  if (screenName === "start") {
    startScreen.classList.add("active");
    maxX = startScreen.clientWidth;
    maxY = startScreen.clientHeight;
  } else if (screenName === "play") {
    playScreen.classList.add("active");
    maxX = playScreen.clientWidth;
    maxY = playScreen.clientHeight;
  } else if (screenName === "end") {
    endScreen.classList.add("active");
    maxX = endScreen.clientWidth;
    maxY = endScreen.clientHeight;
  } else if (screenName === "clear") {
    clearScreen.classList.add("active");
    maxX = clearScreen.clientWidth;
    maxY = clearScreen.clientHeight;
  }
  // 画面サイズ出力(デバッグ)
  console.log(maxX, maxY);
・・・略・・・
```

結果：

showScreen("***"); の *** には、start / play / end / clear のいずれかを指定してください。

DevTools のコンソールに「幅, 高さ」として画面サイズが表示されれば成功です。

※ 各画面（start / play / end / clear）のサイズはすべて同じです。