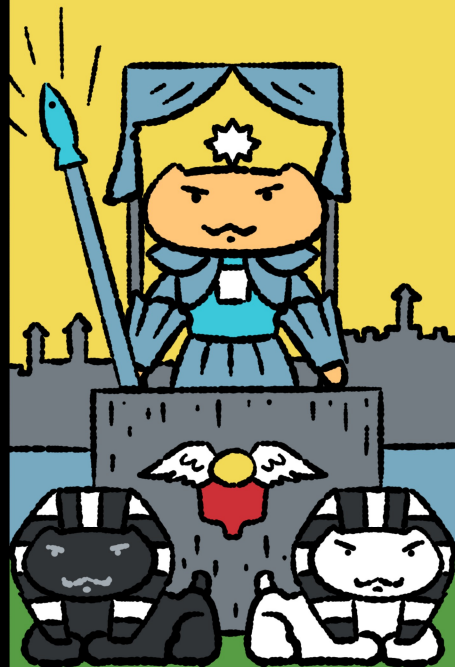


カードが動く タロット占い をつくろう

JavaScriptでアニメーションするWebアプリ

VII



THE CHARIOT.

VI



THE LOVERS.

VIII



STRENGTH.

Ruten no Oheya

Masakazu Yanai

まえがき

Web ブラウザー上で「カードが動くタロット占い」を作りたい。そうした話を聞いたのでプログラムを書いてみました。画面上にカードを並べ、シャッフルし、配るものです。最終的に表側になったカードの内容をもとに、占い結果の文章を出力します。

シンプルなものですが、似たものを作ったことがなければ、こういった技術を選べばよいのか分かりません。知っていれば簡単ですが、知らなければなかなか取っ掛かりがないものです。

「カードが動くタロット占い」を実現する方法はいくつかあります。その中で今回は、次のような方針で作ることにしました。

- canvas 要素に描画する
- Tween 方式でアニメーションする
- PC、スマホで使うことを想定して、画面サイズに合わせて初期 canvas サイズを変える
- 結果の文章は、canvas 要素の外にテキストとして出力させる
- 既存の Web サイトに占いページを追加することを想定する
- 最終的にライブラリーとしてファイルをまとめる

canvas 要素への描画は、『**Pixi.js**』を使います。ゲーム用の有名ライブラリーで、WebGL を利用して高速に描画できます。WebGL が利用できない場合は 2D で描画されます。

Tween 方式のアニメーションは『**Anime.js**』を使います。軽量で使いやすいアニメーション用のライブラリーです。DOM 要素を動かすだけでなく、オブジェクトのプロパティの値をアニメーションできます。

Tween という言葉は、in between から来ており、アニメーションの始まりから終わりの間を補完する機能のことです。こうした処理をおこなうライブラリーは複数存在しています。

『Pixi.js』『Anime.js』の2つのライブラリーは、いずれも MIT license です。費用が掛からず、安心して利用できます。

それでは Web ブラウザー上で「カードが動くタロット占い」アプリケーションを作っていきます。

2024-10 柳井政和

この本の対象者

この本は JavaScript の知識がある人向けに書いています。JavaScript のプログラムをまったく書いたことがない人は、初心者向けの本を読み、ある程度経験を積んでからこの本を読んだ方がよいと思います。

この本のプログラムでは非同期処理を多用しています。Promise や async/await に慣れていない方は、私が Amazon で公開している『マンガでわかるJavaScriptのPromise』を読むとよいです。Amazon インディーズマンガを利用して無料で公開しており、マンガで分かりやすく説明しています。

マンガでわかるJavaScriptのPromise

<https://www.amazon.co.jp/gp/product/B09XQ89M9Y>

動作確認

本書付属のサンプルをサーバー上で実行するか、下の URL から動作を確認できます。

タロット：今日の運勢

https://crocro.com/tools/item/fortune_telling_tarot_one_oracle/

同梱の「猫タロット」についての解説は、下の URL の電子書籍で確認できます。Amazon インディーズマンガを利用して無料で公開しています。

猫タロット本 Cat Tarot Book

<https://www.amazon.co.jp/gp/product/B0DHSV4PSSR>

▶ マンガ...Promise



▶ タロット：今日の運勢



▶ 猫タロット本...



目次

- まえがき
 - 目次
- 第1章 開発するプログラム
 - 1-1 画面の遷移
 - 1-2 ファイル構成
- 第2章 開発の準備
 - 2-1 開発環境の準備
 - 2-2 Pixi.js
 - 2-3 Anime.js
 - 2-4 画像とフォント
 - 2-5 タロット占いのデータ data-tarot.js
- 第3章 アプリケーションの入り口
 - 3-1 HTMLファイル index.html
 - 3-2 CSSファイル
 - 3-3 エントリーポイント main.js
 - 3-4 縦横サイズ size.js
 - 3-5 リソースの読み込み preload.js
- 第4章 タイトル画面
 - 4-1 【進行】アプリの進行 proc.js
 - 4-2 【進行】タイトル画面 proc-01-title.js
 - 4-3 【UI】テキストボタンを作る ui-button-text.js
 - 4-4 【UI】フェードアウトとフェードイン ui-fade.js
- 第5章 カード操作
 - 5-1 【進行】カードを積み重ねる proc-02-stack.js
 - 5-2 【UI】メッセージボックスを作る ui-message-box.js
 - 5-3 【UI】スプライトボタンを作る ui-button-sprite.js
 - 5-4 【進行】シャッフル画面 proc-03-shuffle.js
 - 5-5 【進行】カードの展開 proc-04-spread.js
- 第6章 占いの結果
 - 6-1 【進行】カードの選択 proc-05-select.js
 - 6-2 【進行】カードの開示 proc-06-open.js
 - 6-3 【進行】結果の表示 proc-07-result.js
- 第7章 公開の準備
 - 7-1 Web フォントの軽量化 fontmin
 - 7-2 ライブラリーの出力 vite
- あとがき
- 宣伝
- 奥付

第1章

開発するプログラム

1-1 画面の遷移

開発するプログラムは、Web ページ上で、カードがアニメーションするタロット占いです。このアプリケーションがどのように動作するのかを確認します。

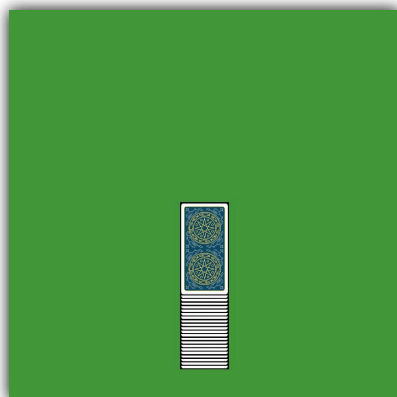
まずはタイトル画面です。右から左にカードがスクロールします。また、上部にタイトル文字を、下部にボタンを表示します。

▶ タイトル画面



次はカードの積み重ねです。少しずれた状態から始まり、カードがスライドして積み重なります。そして上部にメッセージが表示されます。カードの1番上にマウスを載せると明るくなり、クリック可能なことを伝えます。クリックすると先に進みます。

▶ カードの積み重ね1



▶ カードの積み重ね2



▶ カードの積み重ね3



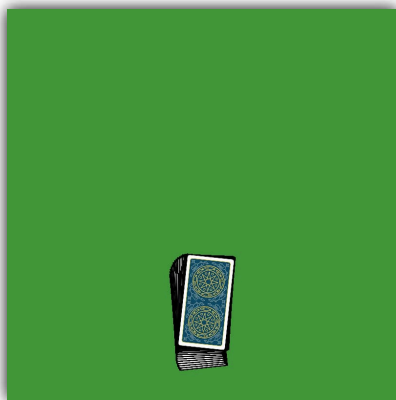
続いてカードのシャッフルです。カードをドーナツ状に散布したあと回転させます。下部にボタンを表示します。ボタンをクリックすると先に進みます。

▶ シャッフル

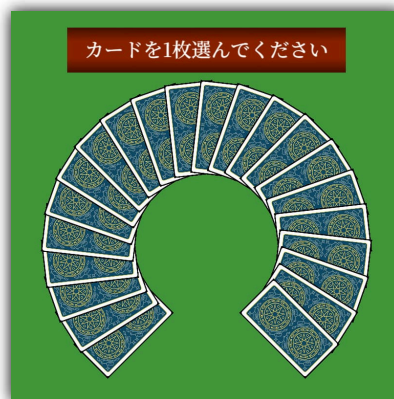


シャッフルが終了すると、カードをいったん集めたあと扇状に並べます。また、上部にメッセージを表示します。カードにマウスを載せると明るくなり、クリック可能なことを伝えます。カードを1枚選んでクリックすると先に進みます。

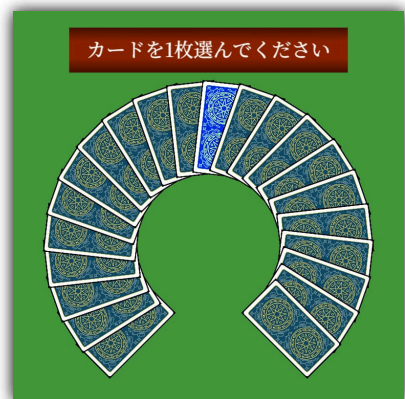
▶ カードの集合



▶ カードの展開1

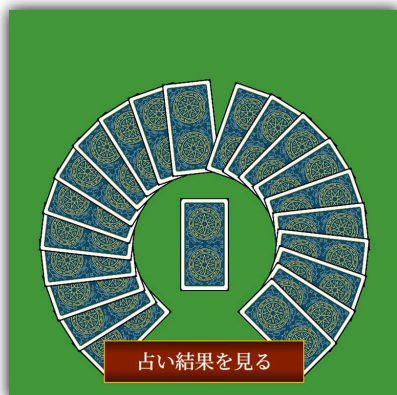


▶ カードの展開2

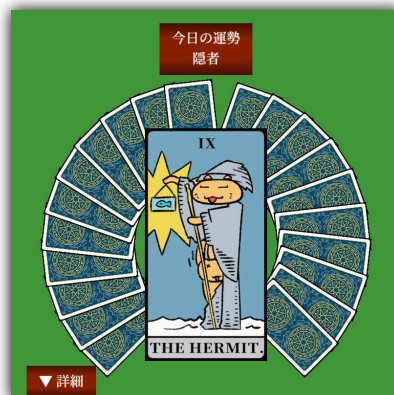


選んだカードが中央に移動します。下部にボタンを表示します。ボタンをクリックすると、カードが拡大して表側になります。また、canvas 要素の外に、今日の運勢をカード付きで表示します。

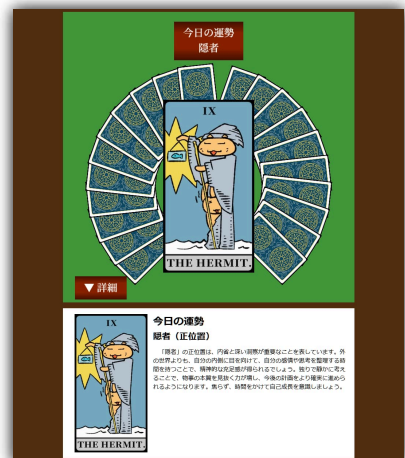
▶ 占い結果1



▶ 占い結果2



▶ 占い結果3



1-2 ファイル構成

開発するプログラムのファイル構成は、次のとおりです。今回作成するプログラムは「js-my/main.js」から始まります。細かな処理は「js-my」ディレクトリー内に入っています。

- index.html
- css/main.css …… アプリを設置するWebサイト側のスタイル
- css-my/ …… アプリ固有のスタイル
 - tarot.css …… このファイルから、同じ階層の各ファイルを読み込む
 - fit-ele.css …… アプリを外側の要素にフィットさせるスタイル
 - output.css …… 占い結果のテキスト出力欄のスタイル
- font/NotoSerifJP-Medium.ttf …… フォント
- image/card/
 - 00-TheFool.jpg …… カードの画像
 - ⋮
- js-lib/
 - anime.min.js …… Anime.js
 - pixi.min.js …… Pixi.js
- js-my/
 - main.js このファイルから、同じ階層の各ファイルを読み込む
 - data-tarot.js …… 占い結果のテキストデータ
 - preload.js …… 画像やフォントの読み込み
 - proc.js …… 後続の proc-～.js を順に実行
 - proc-01-title.js …… 【進行】タイトル
 - proc-02-stack.js …… 【進行】カードの積み重ね
 - proc-03-shuffle.js …… 【進行】カードのシャッフル
 - proc-04-spread.js …… 【進行】カードの展開
 - proc-05-select.js …… 【進行】カードの選択
 - proc-06-open.js …… 【進行】カードの開示
 - proc-07-result.js …… 【進行】占いの結果
 - size.js …… サイズ自動調整
 - ui-button-sprite.js …… 【UI】スプライトボタン
 - ui-button-text.js …… 【UI】テキストボタン
 - ui-fade.js …… 【UI】フェードイン、アウト
 - ui-message-box.js …… 【UI】メッセージボックス

第2章

開発の準備

2-1 開発環境の準備

Visual Studio Code

開発は『Visual Studio Code』（VSCode）でおこないます。下のサイトからダウンロードしてインストールします。

Visual Studio Code - Code Editing. Redefined

<https://code.visualstudio.com/>

Live Server

作成するプログラムは、Web サーバー経由でファイルを読み込まなければ動作しません。そのため、VSCode の拡張機能『Live Server』をインストールします。

Live Server - Visual Studio Marketplace

<https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>

インストール後、VSCode の Explorer（ファイル一覧）で HTML ファイルを右クリックします。メニューに「Open with Live Server」が追加されています。この「Open with Live Server」を選ぶと、ローカルサーバーが起動して Web ブラウザーでファイルを確認できます。

導入したあと、サンプルプログラムの `index.html` を「Open with Live Server」で実行してください。上手く Web ブラウザーで開くことができたなら先に進みます。

『Live Server』の設定について触れます。『Live Server』は、プロジェクト内のファイルを保存するたびに、Web ブラウザーの表示を強制的にリロードさせます。このリロード機能を切りたい時には、以下の設定をおこないます。

プロジェクトのルートに `.vscode` ディレクトリを作り、その中に `settings.json` を作ります。そして、以下の設定を書き込みます。

`/.vscode/settings.json`

```
001 {  
002   "liveServer.settings.ignoreFiles": [ "**/*" ]  
003 }
```

既に `settings.json` がある場合は、2行目の設定を追加します。

Node.js

作成したプログラムは、最後にライブラリー化します。またフォントの縮小化もおこないます。それらの処理を『Node.js』でおこないます。まだ導入していない場合はインストールしてください。

Node.js — Run JavaScript Everywhere

<https://nodejs.org/>

Promiseとasync/await

この本のプログラムでは非同期処理を多用しています。Promise や async/await に慣れていない方は、私が Amazon で公開している『マンガでわかるJavaScriptのPromise』を読むとよいです。Amazon インディーズマンガを利用して無料で公開しており、マンガで分かりやすく説明しています。

マンガでわかるJavaScriptのPromise

<https://www.amazon.co.jp/gp/product/B09XQ89M9Y>

▶ マンガでわかるJavaScriptのPromise



2-2 Pixi.js

公式サイトの説明

『Pixi.js』は、OpenGL を利用してゲームの描画をおこなうライブラリーです。OpenGL が利用できない場合は、ピクセルベースの 2D Canvas で描画をおこないます。『Pixi.js』は実績が豊富なライブラリーです。MIT license なので費用が掛からず、安心して利用できます。

公式サイトは次の場所です。

PixiJS | The HTML5 Creation Engine | PixiJS

<https://pixijs.com/>

基本的には「Examples」のページを見ると、必要な処理の例がまとまっています。それぞれ、サンプルコードと実行例が1つのページに入っています。それほど時間が掛からないので、全てを見ておくと、どんなことができるのかが分かります。ほとんどの機能は、この「Examples」からのコピペで使えます。

Examples | PixiJS

<https://pixijs.com/8.x/examples>

APIの細かな挙動を知りたい時は、「Documentation」のページを見ます。クラス、メソッド、プロパティーなどの情報がまとまっています。踏み込んだプログラムを書きたい場合には、これらの情報を活用します。

PixiJS API Documentation

<https://pixijs.download/release/docs/index.html>

ファイルの入手

この本のサンプルには `pixi.min.js` を同梱しています。この `pixi.min.js` の入手方法を書きます。

GitHub の「Releases」のページから、ビルド済みの `pixi.js` や、縮小化済の `pixi.min.js` を入手できます。またこのページには、ES modules として読み込める `pixi.mjs` や `pixi.min.mjs` もあります。用途に合わせて使い分けるとよいです。

この本を書いている時点では、『Pixi.js』のバージョンは「8.4.1」です。そのためこの本では、バージョン「8.4.1」を前提として説明します。

コード例1 .mjs

シンプルなコード例を示します。カードの画像を1枚読み込んで、Webページの中央でゆっくりと回転させるものです。

まずは、ES modules として読み込む方法を掲載します。HTML、CSS、JavaScript のファイルを示します。

sample/sample-pixi-mjs/index.html

```
001 <!DOCTYPE html>
002 <html lang="ja">
003   <head>
004     <meta charset="utf-8">
005     <meta name="viewport" content="width=device-width, user-scalable=0">
006     <title>Sample</title>
007     <link rel="stylesheet" type="text/css" href="./main.css">
008     <script type="module" src="./main.mjs"></script>
009   </head>
010   <body></body>
011 </html>
```

sample/sample-pixi-mjs/main.css

```
001 html, body { margin: 0; padding: 0; }
002 canvas { position: absolute; }
```

sample/sample-pixi-mjs/main.mjs

```
001 import { Application, Assets, Sprite } from './pixi.min.mjs';
002
003 // Init Pixi App
004 const app = new Application();
005 await app.init({ background: '#1099bb', resizeTo: window });
006 document.body.appendChild(app.canvas); // Append App
007
008 // Init Sprite
009 const texture = await Assets.load('./00-TheFool.jpg');
010 const sprite = new Sprite(texture);
011 app.stage.addChild(sprite);
012
013 // Anchor and Position
014 sprite.anchor.set(0.5);
015 sprite.x = app.screen.width / 2;
```


| | |
|-----|---|
| 016 | sprite.y = app.screen.height / 2; |
| 017 | |
| 018 | // Loop Animation |
| 019 | app.ticker.add(time => { |
| 020 | sprite.rotation += 0.01 * time.deltaTime; |
| 021 | }); |

JavaScript ファイルの説明をおこないます。

1行目では、`./pixi.min.mjs` から `Application`, `Assets`, `Sprite` をインポートします。

4～6行目では、『Pixi.js』の `Application` を初期化したあと、`document` に追加します。

9～11行目では、`Sprite` を初期化して、`Application` に追加します。

14～16行目では、アンカー（基準点）を中央にして、X位置、Y位置を描画領域の中央にします。

19～21行目では、アニメーションのフレームごとに回転を加算します。`time.deltaTime` は、60fpsの時には `1` の値が入っています。60fpsよりも遅延する際には、その比率に応じて大きな値が入っています。

コード例2 .js

次に、`script` タグから読み込む方法を掲載します。HTML、CSS、JavaScript を示します。

| sample/sample-pixi-js/index.html | |
|----------------------------------|--|
| 001 | <!DOCTYPE html> |
| 002 | <html lang="ja"> |
| 003 | <head> |
| 004 | <meta charset="utf-8"> |
| 005 | <meta name="viewport" content="width=device-width, user-scalable=0"> |
| 006 | <title>Sample</title> |
| 007 | <link rel="stylesheet" type="text/css" href="./main.css"> |
| 008 | <script defer src="./pixi.min.js"></script> |
| 009 | <script defer src="./main.js"></script> |
| 010 | </head> |
| 011 | <body></body> |
| 012 | </html> |

| sample/sample-pixi-js/main.css | |
|--------------------------------|---------------------------------------|
| 001 | html, body { margin: 0; padding: 0; } |
| 002 | canvas { position: absolute; } |

| sample/sample-pixi-js/main.js | |
|-------------------------------|-------------------------------------|
| 001 | (async () => { |
| 002 | // Init Pixi App |
| 003 | const app = new PIXI.Application(); |

| | |
|-----|--|
| 004 | await app.init({ background: '#1099bb', resizeTo: window }); |
| 005 | document.body.appendChild(app.canvas); // Append App |
| 006 | |
| 007 | // Init Sprite |
| 008 | const texture = await PIXI.Assets.load('./00-TheFool.jpg'); |
| 009 | const sprite = new PIXI.Sprite(texture); |
| 010 | app.stage.addChild(sprite); |
| 011 | |
| 012 | // Anchor and Position |
| 013 | sprite.anchor.set(0.5); |
| 014 | sprite.x = app.screen.width / 2; |
| 015 | sprite.y = app.screen.height / 2; |
| 016 | |
| 017 | // Loop Animation |
| 018 | app.ticker.add(time => { |
| 019 | sprite.rotation += 0.01 * time.deltaTime; |
| 020 | }); |
| 021 | }()); |

先に紹介した、ES Modules は top-level await なので `(async () => { })();` を付けていませんでした。しかし、こちらでは `await` を使うために、このような記述が必要です。

また、`PIXI.Application` のように、『Pixi.js』のクラスに `PIXI.` を加えます。

この本で作成する占いアプリでは、`script` タグから読み込む方法でプログラムを書きます。これは、作成したプログラムをライブラリー化する場合に、手順を簡単にするためです。

2-3 Anime.js

公式サイトの説明

『Anime.js』は、Tween 方式でアニメーションをおこなうライブラリーです。MIT license なので費用が掛からず、安心して利用できます。

『Anime.js』は、軽量で使いやすく、DOM だけでなくオブジェクトのプロパティに対してもアニメーションをおこなえます。そのため『Pixi.js』と簡単に組み合わせることができます。

公式サイトは次の場所です。

anime.js • JavaScript animation engine

<https://animejs.com/>

基本的には「Documentation」のページを見ると、必要な処理の例がまとまっています。それぞれ、実行例とサンプルコードが横に並んでいます。それほど時間が掛からないので、全てを見ておくとなんかことができるのかが分かります。ほとんどの機能は、この「Documentation」からのコピペで利用できます。

Documentation | anime.js

<https://animejs.com/documentation/>

ファイルの入手

この本のサンプルには `anime.min.js` を同梱しています。この `anime.min.js` の入手方法を書きます。

GitHub の「Releases」のページから、ビルド済みの `anime.js` や、縮小化済の `anime.min.js` を入手できます。

Releases · juliangarnier/anime

<https://github.com/juliangarnier/anime/releases>

「Source code」をダウンロードして解凍します。lib ディレクトリー内に、`anime.js` `anime.min.js` `anime.es.js` が入っています。`anime.es.js` は、ES modules として読み込めるものです。用途に合わせて使い分けるとよいです。

この本を書いている時点では、『Anim.js』のバージョンは「3.2.2」です。そのためこの本では、バージョン「3.2.2」を前提として説明します。

コード例1 .mjs

シンプルなコード例を示します。カードの画像を1枚読み込んで、Webページの中央でゆっくりと回転させるものです。

まずは、ES modules として読み込む方法を掲載します。HTML、CSS、JavaScript のファイルを示します。

sample/sample-anime-es/index.html

```
001 <!DOCTYPE html>
002 <html lang="ja">
003   <head>
004     <meta charset="utf-8">
005     <meta name="viewport" content="width=device-width, user-scalable=0">
006     <title>Sample</title>
007     <link rel="stylesheet" type="text/css" href="./main.css">
008     <script type="module" src="./main.mjs"></script>
009   </head>
010   <body>
011     
012   </body>
013 </html>
```

sample/sample-anime-es/main.css

```
001 img { display: block; margin: auto; }
```

sample/sample-anime-es/main.mjs

```
001 import anime from './anime.es.js'
002
003 anime({
004   targets: '#card',
005   duration: 4000,
006   easing: 'linear',
007   loop: true,
008   rotate: 360
009 });
```

`anime()` 関数の引数に、設定を書いたオブジェクトを指定することで、アニメーションさせることができます。ここでの設定は、対象が `id="card"` の要素、間隔が `4000` ミリ秒、減速方法は `linear`（等速）、ループあり、回転 `360` 度になります。

『Anime.js』では、`targets` に指定したセレクターの要素や、オブジェクトに対して処理をおこないます。この `targets` には複数の要素やオブジェクトを指定することもできます。

そして、`duration` や `easing` といった既定のプロパティ以外の値（ここでは `rotate`）が、「現在の値」から「指定の値」まで変化するようにアニメーションをおこないます。

コード例2 .js

次に、`script` タグから読み込む方法を掲載します。HTML、CSS、JavaScript を示します。

sample/sample-anime-js/index.html

```
001 <!DOCTYPE html>
002 <html lang="ja">
003   <head>
004     <meta charset="utf-8">
005     <meta name="viewport" content="width=device-width, user-scalable=0">
006     <title>Sample</title>
007     <link rel="stylesheet" type="text/css" href="./main.css">
008     <script defer src="./anime.min.js"></script>
009     <script defer src="./main.js"></script>
010   </head>
011   <body>
012     
013   </body>
014 </html>
```

sample/sample-anime-js/main.css

```
001 img { display: block; margin: auto; }
```

sample/sample-anime-js/main.js

```
001 anime({
002   targets: '#card',
003   duration: 4000,
004   easing: 'linear',
005   loop: true,
006   rotate: 360
007 });
```

この本で作成する占いアプリでは、`script` タグから読み込む方法でプログラムを書きます。これは、作成したプログラムをライブラリー化する場合に、手順を簡単にするためです。

2-4 画像とフォント

画像1 猫タロット

画像はタロットカードの表面22枚と、裏面1枚の合計23枚が必要です。この本ではサンプルとして、筆者が描いた「猫タロット」の画像を同梱しています（この絵の著作権は筆者が保有しています）。

▶ 表面



▶ 裏面



同梱の「猫タロット」についての解説は、下の URL の電子書籍で確認できます。Amazon インディーズマンガを利用して無料で公開しています。一度確認しておくといいでしょう。

猫タロット本 Cat Tarot Book

<https://www.amazon.co.jp/gp/product/BODHV4PSSR>

▶ 猫タロット本 Cat Tarot Book



画像2 公開用画像

実際に公開用の占いアプリを作る際には、著作権的にフリーなタロットカードの画像一式を手するか、自作する必要があります。

筆者作の「猫タロット」で参考にした「ウェイト版タロット」は、1909年に公開され、1942年に著作者（アーサー・エドワード・ウェイト）が死去しています。そのため著作権が「著作者の死後80年以下である国・地域」においてはパブリックドメインの状態にあります。

ウェイト版タロット - Wikipedia

<https://ja.wikipedia.org/wiki/%E3%82%A6%E3%82%A7%E3%82%A4%E3%83%88%E7%89%88%E3%82%BF%E3%83%AD%E3%83%83%E3%83%88>

また、この画像を利用した、CC0 の画像セットも作られて公開されています。

Rider-Waite Smith Tarot Cards (CC0) by luciellaes

<https://luciellaes.itch.io/rider-waite-smith-tarot-cards-cc0>

こうしたデータを使う手もありますが、なるべく自前で用意した方がよいでしょう。

手軽に用意できない場合は、生成AIを使うのも一つの手です。「ウェイト版タロット」を入力画像にして、新たな画像を生成する方法も使えるでしょう。

フォント

「Open Font License」の `NotoSerifJP-Medium.ttf` を利用しています。Google Fonts から入手できます。

Noto Serif Japanese - Google Fonts

<https://fonts.google.com/noto/specimen/Noto+Serif+JP>

この本を書いている時点では、Web ページの右上にある「Get Font」ボタンを押したあと、「Download all」ボタンを押すと `Noto_Serif_JP.zip` という74.2MBの ZIP ファイルがダウンロードされます。この中の `static` ディレクトリーにある `NotoSerifJP-Medium.ttf` (7.66MB) を利用します。

この7.66MBというサイズは、Web で使うには少し大きいので、のちほど必要な文字データだけにして、ファイルサイズを107KBまで削減します。

2-5 タロット占いのデータ data-tarot.js

タロット占いの結果のデータは、22枚の各カードについて、正位置（上向き）と逆位置（下向き）を用意します。そのため22×2で44種類の文章が入っています。このデータが入っている `data-tarot.js` の文字数は、13,800字ほどあります。

用意するのが大変な場合は、正位置だけにするのも手です。

全てを本に掲載すると見つらくなるので、一部を抜粋して掲載します。最初のカードである「愚者」の正位置と逆位置のテキストを中心に掲載します。タロット占いのプログラムでは、こうしたテキストをカードの枚数分用意する必要があります。

テキストデータを作成するのが大変な場合は、ChatGPTなどの生成AIの手を借りるのも一つの手でしょう。

| code-one-oracle/js-my/data-tarot.js | |
|-------------------------------------|---|
| 049 | <code>export const cardDirection = {upright: '正位置', reverse: '逆位置'};</code> |
| 050 | <code>export const cardsText = {upright: {}, reverse: {}};</code> |
| 051 | |
| 052 | <code>//-----</code> |
| 053 | <code>// ##### 00 愚者 正位置: 極端な自由</code> |
| 054 | <code>cardsText.upright['00-TheFool'] = [</code> |
| 055 | <code> '「愚者」の正位置は、あなたに自由な精神と無限の可能性をもたらします。周りの常識に囚われず、自分の直感を信じて行動することが幸運を呼び込むでしょう。しかし、自由には責任が伴うことも忘れないでください。無計画な冒険は魅力的ですが、多少のリスクを考慮することで、大きな成功へと繋がるかもしれません。新しい挑戦に前向きになれる日です。',</code> |
| 056 | <code>];</code> |
| 057 | |
| 058 | <code>// ##### 00 愚者 逆位置: 見切り発車</code> |
| 059 | <code>cardsText.reverse['00-TheFool'] = [</code> |
| 060 | <code> '「愚者」の逆位置は、見切り発車や無謀な行動への警告です。計画性が欠けたまま突き進むと、思わぬトラブルに巻き込まれる可能性があります。自分の行動が本当に正しいか、もう一度冷静に見直す必要があります。気持ちは高ぶっているかもしれませんが、今は慎重な判断が求められる日です。焦らず確実な一歩を心がけましょう。',</code> |
| 061 | <code>];</code> |