

# じゃんけんゲームを創ろう ウェブアプリ 簡単入門

アトリエ未来【著】

第六版

HTML / CSS / JavaScript の  
基礎を習得、じゃんけんゲームを創ります。  
サーバーに公開、iPhoneで楽しく遊びます。

ウェブ制作の基礎を學べる一冊です

# ウェブアプリ簡単入門

— じゃんけんゲームを創ろう —

[著] アトリエ未来

「技術書典 14 新刊」  
令和五年五月五日

## ■免責

本書は情報の提供のみを目的としています。

本書の内容を実行・適用・運用したことで何が起きようとも、それは実行・適用・運用した人自身の責任であり、著者や関係者はいかなる責任も負いません。

## ■商標

本書に登場するシステム名や製品名は、関係各社の商標または登録商標です。

また本書では、™、®、©などのマークは省略しています。

# 始めに

楽しいプログラミングの世界へようこそ。情報技術「IT」を日々利用して暮らしているわたくしたち。多くの先人が築いた歴史の上に今日があります。<sup>こんにち</sup>

本書では、「じゃんけんゲームを創ろう ウェブアプリ簡単入門」として、HTML / CSS / JavaScript の概要を学び、じゃんけんアプリを作っていきます。簡単なじゃんけんゲームですが、iPhone で楽しく遊べるものとなっています。

巻末には、今後の成長へつなげて欲しいとの願いから、参考書籍、参考リンク集、HTML / CSS / JavaScript の簡易まとめを付けました。

現代の魔法、それがプログラミングです。自由自在にコンピュータを操って、幸せな未来へと大きく羽ばたいていってください。

## ♣ 対象読者

コンピュータの基本操作ができる初心者の方を想定しています。中高生から社会人まで、ウェブサイトを作成みたい方や、プログラミングに興味があって、なにかアプリを作成みたい人の最初の一歩になればと願っています。<sup>\*1</sup>

また、誰かに教える方が学習希望者にお渡しするテキストとしても、お読み頂ければ幸いです。

## ♣ プログラムのダウンロード

この本で作成する「じゃんけんゲーム」のプログラムは、GitHub<sup>\*2</sup> で公開しています。

また、完成したじゃんけんゲームは <https://joyful-janken.netlify.app> で公開しておりますので、お楽しみください。

## ♣ 謝辞

Re:VIEW Starter<sup>a</sup>を用いて、快適に執筆することができました。作者の kauplan さんに厚く御礼申し上げます。

---

<sup>a</sup> <https://kauplan.org/reviewstarter/>



<sup>\*1</sup> コンピュータとして Mac をお使いの方を対象に執筆しておりますので、Linux や Windows をお使いの方は、キーボード操作等、一部異なる箇所かもしれません、適宜読み替えていただければ幸いです。

<sup>\*2</sup> <https://github.com/Atelier-Mirai/joyful-janken>

また、表紙絵の女の子は、千葉県松戸市在住のフリーランス SD イラストレーター 早瀬ひろむ<sup>a</sup>さんの作品です。素敵なイラストを描いてくださいり、ありがとうございます。

---

<sup>a</sup> <https://hiromu-hayase.tumblr.com>



早瀬ひろむ さん

背景の桜と青空の絵は、くらうど職人<sup>a</sup>さんの作品です。桜咲く青空で心も明るくなります。

---

<sup>a</sup> <https://www.photo-ac.com/profile/590911>



くらうど職人 さん

また、信頼できる文献に触れて欲しいとの思いから、MDNやウィキペディア等、多くの文献より引用させていただいております。貢献に感謝するとともに、厚く御礼申し上げます。

## ♣ 著者紹介



卓越した技能を有する者として認められる国家資格「応用情報技術者」を保持。平成 30 年より「アトリエ未来<sup>a</sup>」を創業。HTML 講座や Ruby 講座などプログラミングの個人指導や、IT パスポート講座等の資格講座の開催、ウェブサイト作成等を受注している。

趣味の将棋は、日本将棋連盟より三段の免状を允許。日本の美しい自然や豊かな精神性を宿す熊野古道の探訪や、たくさんの花に囲まれた日々を愛している。

---

<sup>a</sup> <https://atelier-mirai.net/>

## 【コラム】金の延棒クイズ

二進数の不思議を感じる、ちょっと豊かな気分に成れるクイズです。



七日の給料の支払いとして金の延べ棒が一本あります。これを使って仕事をしてくれる方への日当を支払いたいと思います。

六回鋏を入れ七等分すれば日払いできますが、金の延べ棒を六回も切り取るのは大変です。

必要最小限の二回切り取ることをしたいですが、どことどこを切れば良いでしょうか？

# 目次

## 始めに

i

<b>第1章 開発環境の準備</b>	<b>1</b>
1.1 コミュニティ主導の高性能エディタ <small>パルサー</small> Pulsar	1
1.2 セキュリティ重視のブラウザ <small>ファイヤフォックス</small> Firefox	6
1.3 基本的な開発方法	6
<b>第2章 初めての HTML</b>	<b>11</b>
2.1 ウェブサイトの三つの言語 HTML / CSS / JavaScript	11
2.2 学習の為の参考サイトや書籍のご紹介	12
2.3 HTML の基本	13
2.4 じゃんけんゲームの土台を作る	17
<b>第3章 初めての JavaScript</b>	<b>21</b>
3.1 初めての JavaScript	21
3.2 変数	23
3.3 定数	25
3.4 以前使用されていた変数宣言のキーワード	26
3.5 命名規則	26
3.6 演算子	27
3.7 関数	28
3.8 JavaScript から HTML を操作する為の仕組み - DOM	33
3.9 イベントリスナと関数	35
3.10 開始ボタンを押すと応答するプログラム	35
3.11 繰り返し処理	37
3.12 条件分岐	39
3.13 亂数の利用	42
3.14 入れ子になった if 文	43
3.15 リファクタリング（再構成）	45
3.16 じゃんけんのアルゴリズム	46
<b>第4章 初めての CSS</b>	<b>51</b>
4.1 ユーザインターフェース (UI) と 利用者体験 (UX)	51
4.2 完成形の HTML	53

---

4.3	CSS の基本	55
4.4	CSS グリッドレイアウト	61
4.5	iPhone での表示結果を確認する	69
4.6	完成形の CSS	71
<b>第 5 章</b>	<b>じゃんけんゲームの完成</b>	<b>75</b>
5.1	プレイヤーの手の取得と コンピュータの手の表示	75
5.2	アニメーション機能と勝敗更新機能	79
5.3	じゃんけんプログラム完成	82
<b>第 6 章</b>	<b>ウェブサイトの公開</b>	<b>85</b>
6.1	Netlify でウェブサイトを公開する	85
6.2	Netlify に会員登録する	86
6.3	Netlify の利用方法	89
<b>付録 A</b>	<b>珠玉の名著のご紹介</b>	<b>105</b>
<b>付録 B</b>	<b>付録: 参考リンク集</b>	<b>111</b>
<b>付録 C</b>	<b>HTML / CSS / JavaScript 簡易まとめ</b>	<b>113</b>
C.1	HTML 簡易まとめ	113
C.2	CSS 簡易まとめ	118
C.3	JavaScript 簡易まとめ	122
<b>終わりに</b>		<b>127</b>

# 第 1 章

## 開発環境の準備

プログラミングを始めるにあたり、必要となる開発環境を整えていきます。エディタとして「Pulsar」を、ブラウザとして「Firefox」をご紹介します。

プログラミングを行うために整える道具は二つ、「エディタ」と「ブラウザ」です。（もちろん「コンピュータ」も必要です。iPad や iPhone でプログラミングできる環境として cloud9<sup>\*1</sup> や paiza<sup>\*2</sup>、CodePen<sup>\*3</sup>などもありますが、Mac や Linux, Windows が動くコンピュータがお勧めです。）

エディタとは、プログラミングが快適に行えるよう様々な支援機能を備えたコーディングのためのソフトウェアです。さまざまなエディタがありますが、ここでは Pulsar をご紹介いたします。

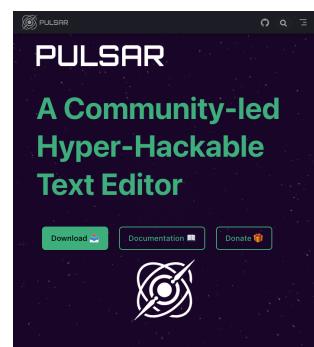
「ブラウザ」は、ウェブサイトを閲覧するためのソフトウェアです。自分の書いたコードの詳細を確認する為の「開発者ツール」や、iPhone や iPad など異なる端末での表示結果を確認する機能も備わっています。さまざまなブラウザがありますが、ここでは Firefox をご紹介いたします。

### 1.1

### コミュニティ主導の高性能エディタ Pulsar

Atom<sup>\*</sup> は「21世紀の高性能エディタ」として GitHub<sup>ギットハブ</sup> により提供されていたエディタです。Microsoft の買収により開発が滞りがちとなり、12月15日で提供終了となりました。Atom の遺産を受け継ぎ、コミュニティ有志により開発、提供されているエディタが、今回ご紹介する Pulsar<sup>パルサー</sup>です。

Pulsar の公式サイトは <https://pulsar-edit.dev/> です。



\*1 <https://aws.amazon.com/jp/cloud9/>

\*2 <https://paiza.io/ja>

\*3 <https://codepen.io>

## ♣ ダウンロード

それでは、お手元のコンピュータへダウンロードして行きましょう。公式サイトの「Download」ボタンよりダウンロードできます。

The screenshot shows the 'Pulsar Downloads' section of the official website. It features a 'Rolling Release' heading with three main download links: 'Linux', 'macOS', and 'Windows'. Each link is accompanied by a small circular icon with a right-pointing arrow.

Linux / macOS / Windows のそれぞれの OS 用のソフトウェアが用意されています。

## ♣ Mac

Macをお使いでしたら Apple Silicon または Intel 用の dmg ファイルをダウンロードします。

The screenshot shows the 'Pulsar Downloads' section for Mac users. It has two main sections: 'Silicon - For Apple Silicon (M1/M2) macs' and 'Intel - For Intel macs'. Each section contains a table with package types and their corresponding file formats.

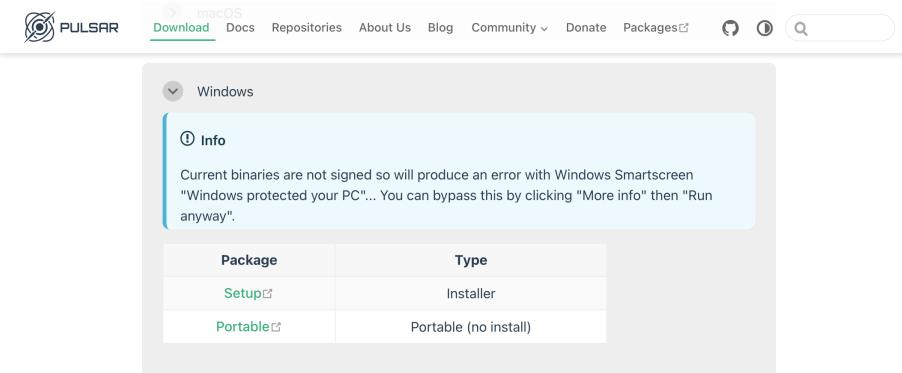
Package	Type
dmg	DMG installer
zip	Zip archive

Package	Type
dmg	DMG installer
zip	Zip archive

ダウンロードが完了したら、落としたファイルをダブルクリックして、インストールして下さい。

## ♣ Windows

Windowsをお使いでしたら Setup をクリックしてダウンロードします。



ダウンロードが完了したら、落としたファイルをダブルクリックして、インストールして下さい。  
(インストールしたPulsarは現在開発中の為、「署名」がされておりません。実行すると「WindowsはあなたのPCを保護した」旨のメッセージが表示されますので、「詳細」→「実行」をクリックすることで、Pulsarを立ち上げることができます。次回以降は、他のアプリと同様に起動できます。)

## ♣ お勧めプラグイン

Pulsarはそのままでも十分高機能に使えるテキストエディタです。そして有志の方により、多くの「プラグイン」が提供されています。プラグインとは、差し込む、差込口などの意味を持つ英単語で、ITの分野では、ソフトウェアに機能を追加する小さなプログラムのことを指します。<sup>\*4</sup>

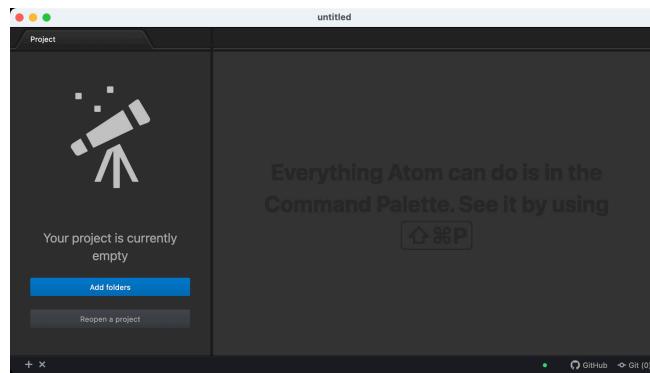
プラグインのことを、Pulsarでは、「パッケージ」と呼んでいます。たくさんのパッケージがありますが、いくつかお勧めをご紹介いたします。

1. メニューバーや設定画面などを日本語化 [japanese-menu](#)
2. カラーピッカー [color-picker](#)
3. CSSの色指定 (#ffffff) を背景色に表示 [pigments](#)
4. ファイルの種類に応じたアイコンを表示 [file-icons](#)
5. カーソルがある行を強調表示 [highlight-line](#)
6. カーソルがある列を強調表示 [highlight-column](#)
7. 選択箇所を強調表示 [selection-highlight](#)
8. =(イコール) の位置を整える [atom-alignment](#)
9. (好みで) AIによるコード補完機能 [tabnine](#)

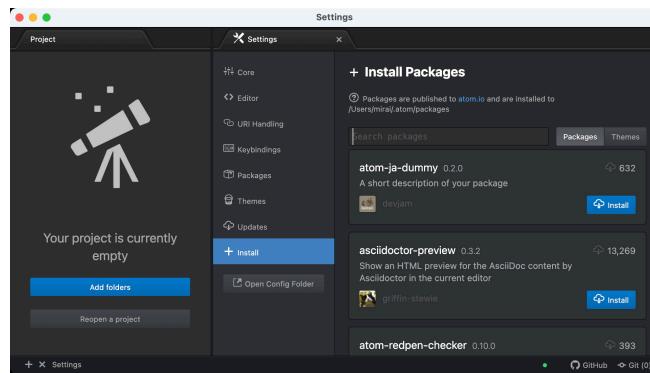
それでは、インストールしていきましょう。

<sup>\*4</sup> 出典: IT用語辞典

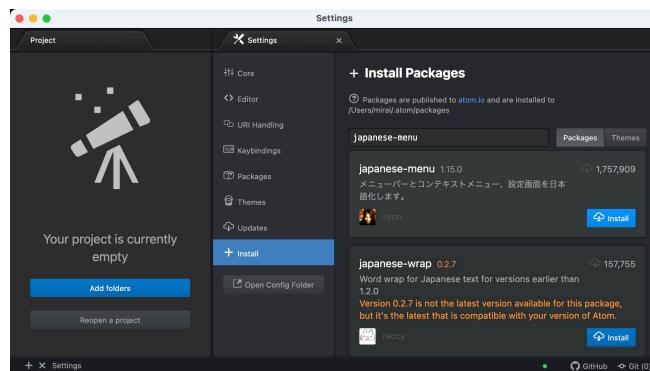
1. Pulsar を起動します。



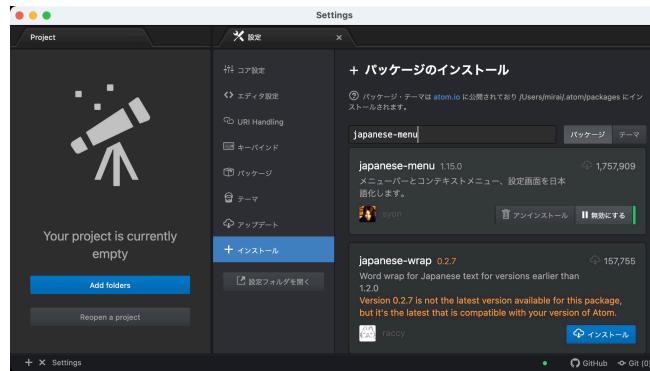
2. File メニューから Preferences をクリックし、環境設定画面を開きます。様々な項目を設定することができます。パッケージをインストールするために、中ほどのメニュー内の「Install」ボタンを押します。



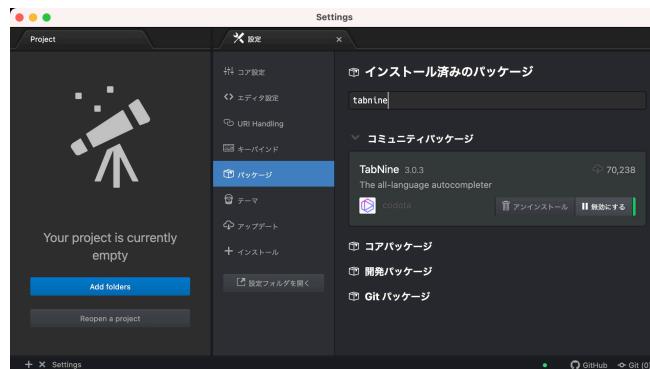
3. 右上の Install Packages の下に入力枠がありますので、インストールしたいパッケージ名として `japanese-menu` を入力し、「Install」ボタンを押します。



4. インストールが完了しました。今まで英語だった説明文が、日本語になっています。



5. 一度インストールしたパッケージを削除したいときには、中ほどのメニュー内の「パッケージ」ボタンを押します。右上の インストール済みのパッケージ の下に入力枠がありますので、アンインストールや無効にしたいパッケージ名を入力します。例えば `tabnine` と入力すると、「アンインストール」ボタンや「無効にする」ボタンが表示されます。



また、有志の方々が創られたさまざまなパッケージが <https://web.pulsar-edit.dev> で公開されています。  
ご自身の使いやすいエディタに育てていって下さい。

The screenshot shows the Pulsar package repository at <https://web.pulsar-edit.dev>. The top navigation bar includes links for 'PULSAR', 'Issues', 'Packages', 'Sign In', and 'Website'. A 'Change Theme' dropdown is also present. The main content area features a search bar with 'Search Packages...' and a 'Featured Packages' section. It lists several packages: 'atom-clock' by 'Antonio Revillaque' (1,091,800 stars), 'dracula-syntax' by 'dracula' (271,849 stars), 'hey-pane' by 'timonsh' (206,981 stars), and 'hydrogen' by 'interact-contributor' (2,564,437 stars). Each package card includes a star icon, a download count, and an 'Install' button.

## 1.2

## セキュリティ重視のブラウザ Firefox

Firefox は、[公式サイト](#)\*5 より、ダウンロードできます。

Firefox は、Mosaic, Netscape の血を受け継ぐブラウザです。CSS グリッドの確認や、丁寧なコメントの開発者ツールが特徴です。

「Firefox をダウンロード」ボタンを押すとダウンロードが始まるので、落としたファイルをダブルクリックして、インストールしてください。



## 1.3

## 基本的な開発方法

Pulsar と Firefox を導入できたので、ウェブサイトの基本的な作り方をご案内します。

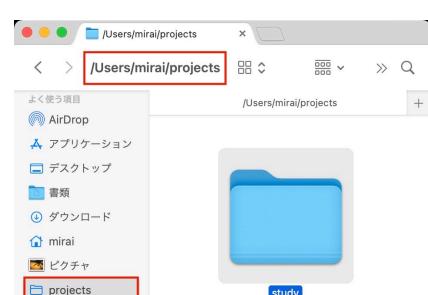
0. 画面の左側に Pulsar を、右側に Firefox を配置します。
1. Pulsar で、プログラムをいろいろ書いていきます。
2. Firefox で、作ったウェブサイトの表示や動作を確認します。
3. 出来上がりが良ければ、ウェブサイトを公開して完了です。

もう少し改良したければ、1. に戻ります。

### ♣ 手順のご案内

作業ディレクトリの作成方法や、Pulsar や Firefox の簡単な使い方の紹介です。 \*6

0. **Users/利用者名ディレクトリ** の直下に、  
**projects** ディレクトリを作成します。  
**projects** ディレクトリには、自分が作成する様々なプロジェクト(案件)を格納する為のディレクトリです。良く使う為、Finder のサイドバーに登録すると便利です。



**projects** ディレクトリを作成し、  
その下に今回の学習用に **study** ディレクトリを作成します。

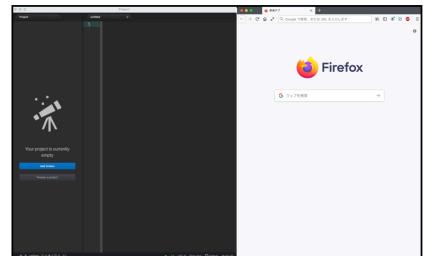
\*5 <https://www.mozilla.org/ja/firefox/new/>

\*6 Mac 利用者向けです。Windows は適宜読み替えてください。

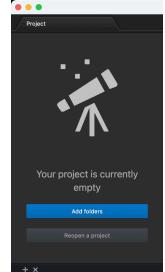
- Pulsar のアイコンと Firefox のアイコンです。  
良く使うのでドックに登録しておくと便利です。



- Pulsar と Firefox を  
左右に配置します。

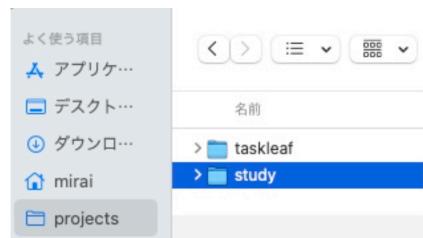


- Pulsar には、「ツリービュー」に「プロジェクトフォルダ」を追加する機能があります。「プロジェクトフォルダ」を追加すると、Pulsar の画面左側の「ツリービュー」にそのプロジェクトフォルダ内のファイルやディレクトリが一覧表示されます。ファイルの追加や編集、削除を簡単に行うことができるようになります。プロジェクトフォルダを追加するためには、左側の青色の「Add folders」ボタンを押します。



- Finder が開くので、サイドバーから projects を選びます。projects ディレクトリから study ディレクトリを開きます。

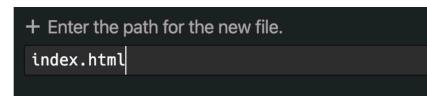
左側の「ツリービュー」に study ディレクトリが表示されますので、この study ディレクトリ内にいろいろなファイルを納めていき、最終的にウェブサイトを創り上げます。新しいファイルを作成しましょう。



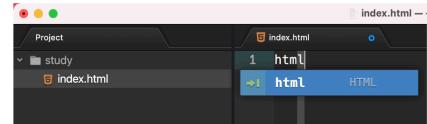
- ツリービューの study ディレクトリを右クリックして、「新規ファイル」をクリックします。



6. 「Enter the path for the new file.」(新しいファイル名を入力して下さい)と、表示されるので、`index.html`とファイル名を入力します。



7. 左側の「ツリービュー」に、新規作成した `index.html` が、表示されましたので、これをクリックします。すると右側の編集領域(「ペイン」と呼びます)に、今作成した `index.html` が表示されます。今作成したばかりなので、中身は何もなく空っぽです。早速 HTML を書いていきましょう。Pulsar には、コードの入力を手助けしてくれる「入力支援機能」が備わっています。`html` と入力して、エンターキーを押します。



8. 次のように雛形が入力されます。

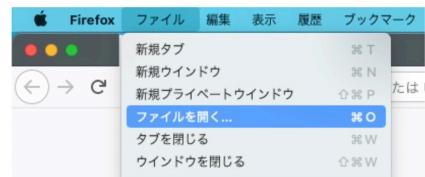
```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>

  </body>
</html>
```

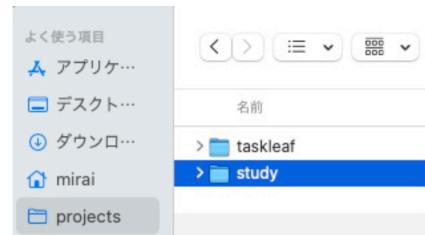
9. 「色」がついていることにも着目してください。「シンタックスハイライト」と呼ばれる、HTML の文法に分かりやすいよう、着色する機能です。それでは雛形で入力した結果を利用して、次のようにしましょう。`h1` とタイプしエンターキーを押すと、`<h1></h1>` と入力されます。Pulsar の入力支援機能を活用して編集しましょう。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>HTML学習</title>
  </head>
  <body>
    <h1>初めてのHTML</h1>
    <p>
      HTMLを学習して
      素敵なサイトを作れるように
      成ります
    </p>
  </body>
</html>
```

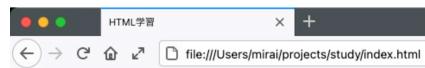
10. HTML を入力して、index.html ファイルが完成しました。Pulsar の「ファイル」メニューから「保存」をクリックするか、Command + S を押して保存します。保存できたら、どのように表示されるか、Firefox を使って確認しましょう。Firefox の「ファイル」メニューから「ファイルを開く」をクリックするか、Command + O を押して、index.html を開きます。



11. Finder が開くので、サイドバーから projects を選びます。projects ディレクトリ内には、taskleaf と study の二つのディレクトリがありますが、ここでは study ディレクトリを開きます。



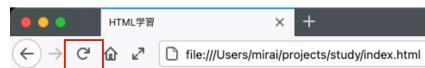
12. Pulsar で編集した index.html を Firefox で閲覧できます。



## 初めてのHTML

HTMLを学習して、素敵なサイトを作れるように成ります。

13. 綺麗に出来上がっていたら完成です。もう少し改善したい点があるなら、Pulsar に戻って index.html を編集します。編集が終わったら Command + S を押して保存します。保存が終わったら、編集結果を Firefox で見てみましょう。Firefox の「再読み込みボタン」を押すか、または Command + R を押して、編集結果を再確認することができます。



## 【コラム】習得したい基本的なショートカット

ショートカットを習得すると、効率的にコーディングできるようになります。

### Mac 基本操作

#### 入力ソース切替

Ctrl + Space	日本語入力と英数入力を切替
--------------	---------------

#### ファイル操作

Command + O	ファイルを開く	Open 開く
Command + S	ファイルを保存する	Save 保存する

#### カーソル移動

Ctrl + F	右へ移動	Forward 先へ進んで
Ctrl + B	左へ移動	Backward 後方へ
Ctrl + P	上へ移動	Previous 以前の行へ
Ctrl + N	下へ移動	Next 次の行へ
Ctrl + A	行頭へ移動	Ahead 前方へ
Ctrl + E	行末へ移動	End 行末へ

#### テキスト編集

Command + Z	元に戻す	
Command + X	切り取り	
Command + C	コピー	Copy
Command + V	貼付	
Command + A	全選択	All
Ctrl + H	カーソルの左の文字を削除	
Ctrl + D	カーソルの右の文字を削除	Delete
Ctrl + K	カーソル以降を切り取り	Kill line
Ctrl + T	カーソル前後の文字を入れ替	Transpose

### Pulsar 基本操作

Shift + Command + D	行の複製	Duplicate
Shift + Ctrl + K	行の削除	Kill line
Command + /	コメントアウト	
Ctrl + F	検索（ファイル内）	Find 見つけ出す
Shift + Ctrl + F	検索（プロジェクト内）	Find 見つけ出す
Ctrl + G	任意行へカーソルを移動	Go
Command + ]	インデントを追加	
Command + [	インデントを削除	
Command + K →	画面分割	
Command + W	画面を閉じる	Window close
Shift + Ctrl + P	コマンドパレット	Pallet
Shift + Command + C	色を選択	pigments

# 第 2 章

## 初めての HTML

じゃんけんゲームの土台となる HTML(HyperText Markup Language) の基礎をご紹介します。HTML は、書かれたものが「見出し」か、「本文」か、あるいは「画像」であるのかという「文書構造」を記述するための言語です。

とても簡潔で理解しやすい文法ですので、習得していきましょう。

### 2.1 ウェブサイトの三つの言語 HTML / CSS / JavaScript

ウェブサイトを作成する為に用いられる言語には、大きく次の三つがあります。

- **ハイパーテキストマークアップランゲージ**

HyperText Markup Language を日本語で表すなら、「ハイパーテキストに目印をつける言語」くらいの意味になります。

目印をつける (Markup) というのは、文書の各部分（見出し・段落・表・リストなど）が果たしている役割が分かるようにすることです。そのために専用の「タグ」を使います。

コンピュータに理解できるよう文書構造を定義することが、HTML の最も重要な役割です。

- **カスケーディング・スタイルシート**

ウェブサイトの見た目を飾り付け、彩る為の言語です。文字の大きさや色の指定から画像の配置先など、様々な装飾を行うことができます。

HTML のタグが親子関係 (包含関係) にある場合、多くの設定値は親要素に指定されたものが子要素、孫要素に引き継がれていきます。このように設定値が上から下へ伝播していく様子を、階段状の瀧を意味する cascade になぞらえて、命名されました。

● ジャバスクリプト  
● JavaScript

ウェブサイトに、「双方向性・相互作用性（インタラクティブ）」を齎すために用いられるプログラミング言語です。

例えば、閲覧者の操作に反応して表示が書き換わったり、ページが表示される際に写真などの要素に動きや効果を加えたり、サーバと通信してデータを取得したりするなど、現在のウェブサイトには欠かせないプログラミング言語となっています。

**まとめ**

- HTML は、ウェブサイトの文書構造を記述します。
- CSS は、ウェブサイトの飾り付けを担当します。
- JavaScript は、ウェブサイトを制御します。

それぞれの詳細を知悉することにより、より高度な作品を作り上げることができるようになっていきます。本書で興味を持たれたならば、ぜひより深く学習なさってください。

ここでは、基礎的なコードを用いて、一緒にじゃんけんゲームを作り上げていきましょう。

## 2.2

## 学習の為の参考サイトや書籍のご紹介

### ♣ HTML / CSS を学ぶ為に

ウェブサイト作成のために、様々な参考書籍があり、またネット上でも有益なサイトが多数あります、始めて学ぶ方向けには、公式サイトである、MDNより提供されている

- ウェブ入門<sup>\*1</sup>
- HTML の基本<sup>\*2</sup>
- CSS の基本<sup>\*3</sup>

が役立つことと思います。

また、どのような仕組みでウェブサイトが閲覧できるのか、少し技術的な背景についても知見があると、（専門家を目指す方はもちろんですが）知的好奇心を満たす点からも楽しいものです。

- ウェブのしくみ<sup>\*4</sup>

\*1 [https://developer.mozilla.org/ja/docs/Learn/Getting\\_started\\_with\\_the\\_web](https://developer.mozilla.org/ja/docs/Learn/Getting_started_with_the_web)

\*2 [https://developer.mozilla.org/ja/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/ja/docs/Learn/Getting_started_with_the_web/HTML_basics)

\*3 [https://developer.mozilla.org/ja/docs/Learn/Getting\\_started\\_with\\_the\\_web/CSS\\_basics](https://developer.mozilla.org/ja/docs/Learn/Getting_started_with_the_web/CSS_basics)

\*4 [https://developer.mozilla.org/ja/docs/Learn/Getting\\_started\\_with\\_the\\_web/How\\_the\\_Web\\_works](https://developer.mozilla.org/ja/docs/Learn/Getting_started_with_the_web/How_the_Web_works)

MDNでの学習を終えた方には、巻末に挙げた参考書籍の中から次の二冊をお勧めします。

- CSS グリッドで作る HTML5&CSS3 レッスンブック
- 作って学ぶ HTML & CSS モダンコーディング

前書は、初心者向けに基礎的なウェブサイトの作成を簡単な技術的な背景も含めて解説されている良書です。本書の執筆に関しても、その多くの部分を負っており、貢献に感謝いたします。後書は、前書の学習を終え、基礎的な HTML/CSS が書けるようになった方が、より進んだサイト作成の技術を学ぶために最適な一冊となっています。

### JavaScript を学ぶ為に

JavaScript の学習に当たっては、以下のMDNの説明が概要を掴むには良いでしょう。

- JavaScript の基本<sup>\*5</sup>

簡潔に説明されてはいますが、始めてプログラミングに触れる方には少し難しいと感じるかもしれません。そういう方へは、巻末の参考書籍

- スラスラ読める JavaScript ふりがなプログラミング

がお勧めです。一語一語、漢文に倣った読み下し文でコードの意味が書かれており、短いコードの一文一文を確かめながら実行することで、理解を深めていくことができるようになっています。

- JavaScript Primer 迷わないための入門書

は、JavaScript をより深く知りたい方にお勧めの一冊。難易度高目ではありますが、前半だけでも読み通すと、言語の全容を知ることができ、他のプログラミング言語の理解にも繋がります。

## 2.3 HTML の基本

テキストエディタ Pulsar の紹介では、次の短い HTML を作成いたしました。この中に基本が詰まっていますので、説明していきます。

### ▼ index.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>HTML学習</title>
6  </head>
```

<sup>\*5</sup> [https://developer.mozilla.org/ja/docs/Learn/Getting\\_started\\_with\\_the\\_web/JavaScript\\_basics](https://developer.mozilla.org/ja/docs/Learn/Getting_started_with_the_web/JavaScript_basics)

```

7 <body>
8   <h1>初めてのHTML</h1>
9   <p>
10    HTMLを学習して、素敵なサイトを作れるように 成ります。
11   </p>
12 </body>
13 </html>

```

```

<!DOCTYPE html>
<html> _____
  <head> _____
    <meta charset="utf-8">
    <title>HTML学習</title>
  </head> _____
  <body> _____
    <h1>初めてのHTML</h1>
    <p>
      HTMLを学習して
      素敵なサイトを作れるように
      成ります
    </p>
  </body> _____
</html>

```

**Webページに関する設定の記述場所**

- ・文字コードとして utf-8を使用すること
  - ・文書の表題が「HTML学習」であること
- を記述しています。

**コンテンツ(内容・出し物)の記述場所**

- ・大見出し(h1)に「初めてのHTML」
  - ・段落(p)に「HTMLを・・・」
- と記述しています。

**♣ ファイル名について**

ファイル名を、index.html としました。index は、「索引」「指針」「指標」などの意味を持つ英単語です。ウェブサイトにアクセスした際、特にページの指定がされなければ、ブラウザは index.html を表示します。例えば、<https://example.com/> にアクセスすると、ブラウザは <https://example.com/index.html> のファイルを表示します。

.html は、**拡張子**と呼ばれます。ファイルの種類が HTML であることを示しています。拡張子が .html であるファイルを開くと、OS により既定のブラウザが開きます。

**♣ DOCTYPE**

HTML 文書の先頭に書かれている <!DOCTYPE html> は、**DOCTYPE 宣言**と呼ばれます。その唯一の役割は **HTML5<sup>\*6</sup>**で書かれている旨をブラウザに伝えることです。今日では無意味な記述

<sup>\*6</sup> ウェブの発明者であるバーナーズリー氏が主催する W3C の規格 HTML5 と、ブラウザ開発企業である Mozilla, Apple, Opera によって設立された団体 WHATWG が推進する規格 HTML Living Standard がありました。HTML Living Standard に合流する形で、HTML5 は廃止されています。

ですが、歴史的な経緯により、先頭に書く必要があります。

### ♣ <html> と <head> と <body> について

<html> 全体は、大きく二つの部分 <head> と <body> に分かれます。<head> は、ウェブページに関する設定を書く場所で、以下のことを記述しています。

- 文字コードとして、utf-8 を使用すること
- 文書の表題が、「HTML 学習」であること

<body> は、コンテンツ(内容・出し物)を書く場所で、以下を記述しています。

- 大見出し(h1)に、「初めての HTML」
- 段落(p)に、「HTML を学習して(略)」

### ..... 文字コードと「文字化け」について

かつてコンピュータが英数字しか扱えなかった時代、アルファベットの「A」は十六進数の 41(十進数の 65)、「B」は 42 と番号を割り振っていました。これを ASCII コードと言います。

その後、日本語を扱えるように拡張され、例えば「日」は十六進数の 93 FA と番号付けされました。これを Shift JIS コードと言います。

Shift JIS コードは長らく使われてきましたが、現在ではほぼ世界中の文字を扱えるようになった UTF-8<sup>ヨーティーエフエイト</sup> が標準となっています。例えば「日」には、E6 97 A5 という文字コードが割り振られています。

同じ「日」という文字ですが、割り当てられている文字コードが違うため、古いウェブサイトを閲覧した際など、正しく文字が読めないことが起こります。これがいわゆる「文字化け」です。

<meta charset="utf-8"> ととても短いコードですが、せっかく作ったウェブサイトが文字化けてしまわぬよう、大事な役割を担っています。

### ♣ 要素とは

HTML では、コンテンツ(内容)を開始タグと終了タグでマークアップ(印付け)することにより、その種類を明確にします。マークアップした部分は左のような構造になり、全体を要素と呼びます。

この例では、開始タグ<h1>と、終了タグ</h1>で囲まれた範囲「初めての HTML」が「大見出し」であることを示しています。



人が見ると「見出し」が「始めての HTML」であり、「本文」が「HTML を学習して・・・」であることは一目瞭然ですが、コンピュータにとってそれが何であるかは自明ではありません。そこで、<h1>始めての HTML</h1>と書くことで、「始めての HTML」が「大見出し」であることを「印

付け（マークアップ）します。

一般に文書には見出しや本文、画像など様々な要素があります。HTMLには100種類以上のタグが用意されていますが、その中から、書かれたそれぞれの要素（部品）が何であるかを表す適切なタグを選択することが、良いウェブサイト作成の第一歩になります。<sup>\*7</sup>

### ♣ タグは入れ子にする

複数のHTMLタグでマークアップする場合、開始タグと終了タグは、他のタグの中に完全に入った状態（入れ子）にすることが求められます。

上は正しいHTMLで、下は誤ったHTMLです。タグの対応状態を確認してください。

```
<body>
  <h1>
    初めてのHTML
  </h1>
</body>
```

```
<body>
  <h1>
    初めてのHTML
  </body>
  </h1>
```

### ♣ 親要素と子要素

入れ子の形で記述すると、階層構造が作られます。このとき、上位階層の要素を「親要素」、下位階層の要素を「子要素」と呼びます。

例えば、<body>は<h1>の親要素、<h1>は<body>の子要素です。



### ♣ コメント(覚書・説明書き)

ソースコードを分かり易くするための覚書、説明書きが出来るよう、コメントと呼ばれる構文があります。

```
<!-- コメント -->
```

作成したウェブサイトを閲覧する際にはないものとして扱われますので、文法上、絶対に書かなければ成らないものはありませんが、適切なコメントを書くことにより、自らの助けともなります。

また、不要となったコードを一時的に非表示にしたり、参考となるコードを残しておく為にも使うことが出来ます。

<!-- --> の間にコメントを書いていきますが、Pulsarなどほとんどのテキストエディタでは、command + / (コマンドキーと /キーを同時に押す) でコメントにしたり、もう一度押す

<sup>\*7</sup> 全部で100以上のタグが用意されていますが、良く用いるタグは十数個です。安心して少しづつ学んでいきましょう。

とコメントを解除できます。エディタの支援機能を活用して、効率的にコーディングしていきましょう。

### ♣ 字下げ(インデント)

HTMLでは、改行や空白は無視されますので、どのように改行しても良いですが、タグの親子関係(入れ子関係)が分かりやすいよう、綺麗に字下げ(インデント)することで、自分が書きたいことを明確にでき、コードの不具合も発見しやすくなります。

#### 好ましい字下げ1

まったく字下げをせず一行に書いていますが、短いコードなのでひと目で分かります。

```
<body><h1>初めてのHTML</h1></body>
```

#### 好ましい字下げ2

タグごとに字下げをした例です。タグの中にタグが配置されているという、抱合関係・親子関係が明確で、これも好ましいコードです。

```
<body>
  <h1>
    初めてのHTML
  </h1>
</body>
```

#### 好ましい字下げ3

全てのタグを字下げすると、行数の増加や、深く成りすぎた字下げにより、かえって分かりにくくなることもあります。そのため、`<h1>`タグはそのまま書いてもひと目で内容が分かるので、字下げせずに書いた例です。これも好ましいコードです。

```
<body>
  <h1>初めてのHTML</h1>
</body>
```

#### 不適切な字下げ

字下げがめちゃくちゃに成っています。少し極端な例を挙げましたが、慣れないうちは書くことに一生懸命でついぐちゃぐちゃに成ってしまいます。上の三つの例のように見易く綺麗に整えることを心がけましょう。

```
<body>
  <h1>
    初めてのHTML</h1>
  </body>
```

## 2.4 ジャンケンゲームの土台を作る

### ♣ 土台となるHTML

HTMLの基礎ができたところで、もう少し発展させていきましょう。ジャンケンゲームの土台とするために、先ほど作成した `index.html` を次のように編集しましょう。

## ▼ index.html

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>じゃんけんゲーム</title>
6     <script src="janken00.js" defer></script>
7   </head>
8
9   <body>
10    <header>
11      <h1>じゃんけんゲーム</h1>
12    </header>
13
14    <main>
15      <p>
16        みんな知っているじゃんけん。  

17        グーは 0, チョキは 1, パー は 2 を入れてね。
18      </p>
19      <input type="number" id="player_hand_type" value="0">
20      <button id="play">開始</button>
21    </main>
22  </body>
23 </html>
```

**♣ コメント付きのHTML**

少し分量が増えました。HTMLには「コメント」機能があります。<!-- 説明書き --&gt;のように記述することで、プログラムの動作には影響を与えることなく、プログラムを読む人のために、説明を記すことができます。</p>

コメント機能を使って、説明書きを追加してみると、次のようにになります。

## ▼ index.html(コメント付き)

```

1 <!DOCTYPE html>
2 <html>
3   <!-- head には 文書に関する設定事項を書きます -->
4   <head>
5     <!-- 文字コードは utf-8 を使います -->
6     <meta charset="utf-8">
7     <!-- このページのタイトルを示すタグで、ブラウザのタブに表示されます。-->
8     <title>じゃんけんゲーム</title>
9     <!-- JavaScriptの読み込み -->
10    <script src="janken00.js" defer></script>
11  </head>
12
13  <!-- body タグには、サイトのコンテンツ（内容・出し物）を記述します。
14    全ての要素は このbodyタグの中に書きます。-->
15  <body>
16    <!-- header タグは、サイトの付帯情報やサイト名などの表示に用います -->
17    <header>
18      <!-- h1 タグは、大見出しを表すタグです。他に 中見出しを表す h2 タグや
```

```

19      小見出しを表す h3 タグなど h6 タグまであります。 -->
20      <h1>じゃんけんゲーム</h1>
21  </header>
22
23  <!-- mainタグは ページの主題となるコンテンツ(内容・出し物)を記述します。-->
24  <main>
25      <!-- p は 段落(paragraph) を表す際に用いるタグです -->
26      <p>
27          <!-- br タグを入れると、途中で改行できます -->
28          みんな知っているじゃんけん。<br>
29          グーは 0, チョキは 1, パー は 2 を入れてね。
30      </p>
31
32      <!-- input タグは、入力枠を作るためのタグです。
33          type="number" と書くと、数字用の入力枠になります。
34          利用者が、0, 1, 2 のいずれかの数字を入力し、、、
35          グー、チョキ、パー のどの手を出すのか、意思表示できるようにします。
36          また、JavaScriptから扱いやすいよう
37          id="player_hand_type" とID属性を付与します。
38          IDを付与すると、HTML要素の中から「一意」に特定の要素を取得できます。
39          value="0" と書いて、枠の中に「0」を表示します。
40          つまりプレイヤーの手は グー(0) と言うことです。 -->
41      <input type="number" id="player_hand_type" value="0">
42
43      <!-- button タグを使って、画面に「開始」ボタンを表示します。
44          このボタンにも、id="play" と ID を付与しておきます。 -->
45      <button id="play">開始</button>
46  </main>
47  </body>
48 </html>
```

紙幅の関係上、細かい説明は割愛しますが、コメントを読むことでそれぞれのコードが果たしている役割は理解できるかと思います。

より詳しく知りたい方には、「[HTML 要素リファレンス\\*8](#)」が、公式サイトとして、とても良くまとまっていますので、是非ご覧になってください。

## ♣ JavaScript の読み込み

注目していただきたいのは次のコードです。

### ▼ index.html

```

9  <!-- JavaScriptの読み込み -->
10 <script src="janken00.js" defer></script>
```

janken00.js という JavaScript で書かれたプログラムを、index.html に読み込んで実行するためのコードです。

JavaScript で書かれたプログラムは、(単体で実行されることはありますが) HTML に読み込み、

\*8 <https://developer.mozilla.org/ja/docs/Web/HTML/Element>

様々な動きを制御する用途で広く用いられます。

本書でも、`janken00.js` というファイルに様々なコードを書き加えていくことで、利用者が入力した手が「グー」であるのか「チョキ」であるのか「パー」であるのかを判断、勝敗を表示するなどのプログラムを作っていきます。

### ♣ ブラウザでの表示結果

いま作成した `index.html` をブラウザで開くと次のようになります。



▲図 2.1: 現状(左) 完成形(右)

左側の現状は、絵も色もなくて少し寂しいですが、最後には右側のように綺麗に仕上げていきます。完成を目指して一緒に学んでいきましょう。

# 第3章

## 初めての JavaScript

JavaScript の基本的な文法を学びつつ、少しずつコードを追加していくことで、じゃんけんゲームを作っていきます。

### 3.1 初めての JavaScript

#### ♣ 初めての JavaScript

初めての JavaScript のプログラムを書き始めていきましょう。

janken00.js というファイルを作り、次のように書きましょう。 \*1

##### ▼ janken00.js

```
1 // じゃんけんの勝ち負けの結果を表示する
2 alert("あなたの勝ちです!")
```

プログラムを書き終わったら保存しましょう。そしてブラウザを再読み込みします。すると、右のように「あなたの勝ちです!」とメッセージが表示されます。

おめでとうございます。初めての JavaScript は、これで完成です!!



#### ♣ 初めての JavaScript 解説

とても少ない行数のプログラムです。それぞれの行について解説していきます。

\*1 「開発環境の準備」の章で、index.html を作りました。同様の手順で、janken00.js というファイルを追加して、これを編集していきます。

## ♣ コメント

// と書かれた部分は、コメントです。コードの説明書きに使います。

### ▼ janken00.js

```
1 // じゃんけんの勝ち負けの結果を表示する
```

処理の大まかな塊ごとに、コメントがついていると、プログラムの全体像が掴みやすく、作成者の意図も分かりやすくなります。一行ずつ全てにコメントを入れる必要はありませんが、要所要所で分かりやすくコメントをつけるように心がけましょう。 \*2

## ♣ alert 関数

JavaScript には、よく使う処理に名前を付けて呼び出す働き・機能があります。この働き・機能のことを「関数」と呼びます。(英語では、機能「function」です。)

メッセージ表示機能は便利ですので、JavaScript には標準で `alert` 関数が備わっています。 \*3

```
2 alert("あなたの勝ちです!")
```

`alert("表示させたいメッセージ")` と書くことで、`alert` 関数を使用できます。

## ♣ ;(セミコロン) の有無

### ▼ セミコロン有りのコード

```
1 // メッセージを表示する
2 alert("こんにちは。JavaScript");
```

### ▼ セミコロン無しのコード

```
1 // メッセージを表示する
2 alert("こんにちは。JavaScript")
```

二種類のコードを掲載いたしました。とてもよく似ているコードで、違いは行末に;(セミコロン) が有るのか無いのかだけです、どちらも正しく動きます。

JavaScript には、自動セミコロン挿入機能があり、多くの場合、セミコロンを付けずとも正しく実行されます。ここでは見た目がすっきりするので、セミコロンを付けないで書くことにします。

JavaScript にセミコロンは入れるのか？入れないのか？ \*4 という記事もございます。よろしかったらお読みください。

## ♣ よりゲームらしくするために

JavaScript を書くことで、勝利メッセージを表示できるようになりました。すこしづつコードを

\*2 /\* コメント \*/ というコメントの書き方もあります。

\*3 標準の関数を利用する他、プログラマが好きな関数を作成できる機能もありますので、後述します。

\*4 <https://zenn.dev/tetsuyaohira/articles/3f84f3971f67d2>

追加していくことで、よりじゃんけんゲームらしくしていきましょう。

それではどのようにすれば、より、じゃんけんゲームらしくなるでしょうか？ いきなり「あなたの勝ちです！」と表示されるのではなく、「プレイヤーが「開始」ボタンを押したら、結果が表示されるようにする」と、よりゲームらしくなります。

それでは、必要となるプログラミングに関する基礎知識を学んで行きましょう。

## 3.2 変数

プログラミング言語には、文字列や数値などのデータに名前をつけることで、繰り返し利用できるようにする変数という機能があります。「変数」とは「値」が格納できる「箱」のようなものであると想像すると理解しやすいでしょう。

より正確な定義が、**IT用語辞典<sup>\*5</sup>**にございますので、確認してみましょう。

変数とは、コンピュータプログラムのソースコードなどで、データを一時的に記憶しておくための領域に固有の名前を付けたもの。変数をつけた名前を変数名と呼び、記憶されているデータをその変数の値という。データの入れ物のような存在で、プログラムの中で複数のデータを扱いたいときや、同じデータを何度も参照したり計算によって変化させたい場合に利用する。

変数をプログラム中で利用するには、これからどんな変数を利用するかを宣言し、値を代入する必要がある。コード中で明示的に宣言しなくても変数を利用できる言語もある。変数に格納された値を利用したいときは、変数名を記述することにより値を参照することができる。

定義ですので、少し難しいかもしれません、何度も出てきますので、すぐに使いこなせるようになる事と思います。

では、これから学ぶ JavaScript ではどのように「変数」を使えば良いのでしょうか。JavaScript では、どのような名前の変数を自分が書くプログラムで使うのか、最初に「宣言」する必要があります。変数宣言をする為のキーワードとして `const`, `let`, `var` の 3 つが用意されていますので、順に見て行きましょう。

- `let` は、「変数」を宣言する時に使います。何度でも値の変更が可能です。

それでは、使って見ましょう。

<sup>\*5</sup> <https://e-words.jp/w/変数.html>

▼ 変数宣言の例

```
// コンピュータの手として、computer という変数を宣言する
let computer
```

変数の宣言をすると、その変数に値を入れる（代入する）ことができます。

▼ 変数に値を入れる例

```
// コンピュータの手として、computer という変数を宣言する
let computer

// computer という変数に値を入れる
computer = 0
```

変数の宣言と同時にその変数に値を入れる（代入する）こともできます。

▼ 変数宣言と、変数への値の代入を同時に行う例

```
// コンピュータの手として、
// computer という変数を宣言するとともに、
// computer という変数に値を入れる
let computer = 0
```

変数には値を何度も代入することができます。次の例は、0, 1, 2 と変化させた例です。

▼ 変数への値の代入を何度も行った例

```
computer = 0
computer = 1
computer = 2
```

また、変数名の宣言は一度だけで十分です。同じ変数名で何度も宣言する必要はありませんし、また出来ませんので、ご注意下さい。

次の例はエラー（誤り）となります。

▼ 変数宣言を何度も行った例

```
let computer
let computer = 0
computer = 1
let computer = 2
```

.....  
=(イコール)記号について

= (イコール) は、数学ですと、computer と 1 が「等しい」ということを意味する記号ですが、プログラミングの世界では 左の変数に右の値を入れる（代入する）という記法になります。

初期には「等しい」ことを示す = と区別するために別の記号が用いられた時代もありましたが、C言語シーゲンゴやRuby, JavaScriptなど現在主流の多くの言語では「代入する」記号として=を用いています。（「等しい」意味では == や === を用います。）

始めのうちは **computer** ⇄ 1 のことだと、頭の中で置き換えると良いでしょう。

.....

## 3.3 定数

それでは、次に「定数」を見て行きましょう。IT用語辞典<sup>\*6</sup>には、次のように書かれています。

定数とは、コンピュータプログラムのソースコードなどで、ある特定の数値やデータに名前を与えたもの。変数とは異なり、宣言時に決めた値をコードの中で後から変更することができない。

プログラムのコード中で何度も参照される値で、実行時に変更する必要のないものに用いられる。値（数値や文字列）が開発の途中で変更される可能性がある場合や、値が長くて何度も入力すると打ち間違える危険がある場合、値に分かりやすい名前をつけたい場合などに用いられる。

「定数」<sup>\*7</sup>を宣言する為には、**const**というキーワードを使います。英語の **constant**（一定の、不变の）に由来しています。定数は一度決めた値を変更することは出来ません。（しようとするとエラーになります。）

- **const** は、「定数」を宣言する時に使います。初回のみ値を設定できます。

それでは、早速、使って見ましょう。先の変数の例では、**computer = 0**などと書いていました。**computer**という変数に 0 を代入するという意味ですが、この 0 という数値は一体どういう意味なのか、覚えておくのも大変です。そこで分かりやすい名前を付けたいと思います。

### ▼定数宣言の例

```
// 定数宣言
const GUU    = 0 // グー
const CHOKI = 1 // チョキ
const PAA    = 2 // パー
```

**const GUU** と定数を宣言すると同時に = 0 で値を設定しています。

こうして定数を宣言しましたので、プログラム中では次のように使うことが出来ます。

\*6 <https://e-words.jp/w/定数.html>

\*7 厳密には「再代入できない変数」の宣言です。

### ▼定数を利用した例

```
// 定数宣言  
const GUU = 0 // グー1  
const CHOKI = 1 // チョキ  
const PAA = 2 // パー2  
  
// コンピュータの手  
let computer = GUU
```

`let computer = GUU` と、プログラムを書くことが出来るようになりました。`let computer = 0` と書くよりも文字数は増えていますが、0, 1, 2 の単なる数字とは異なり、その意味するところが明確で、とても分かりやすくなりました。活用してきましょう。

## 3.4 以前使用されていた変数宣言のキーワード

JavaScriptには、`let`, `const` の他に、`var` という変数宣言の為のキーワードもあります。`var` は、以前は使われていましたが、数々の不具合があるので、それに代わる変数宣言の為のキーワードとして `let`, `const` が導入されました。今でも、古いプログラムコードには使われていますが、これから書くときには、`let` または `const` を用いるようにしましょう。

- `var` は、以前は使われていましたが、数々の不具合があるので、今日では使用しません。

## 3.5 命名規則

`computer` という名前の変数や、`GUU` という名前の定数を宣言しましたが、変数や定数にはどのような名前を付けられるのでしょうか。

変数名には アルファベット (ヒポテト) `a-z`, `A-Z`, `$`, `_` (アンダースコア) が使えます。これらのアルファベット (ヒポテト) の後ろに `0-9` までの数字を続けることも出来ます。また、JavaScript 自身が使う為の単語 (予約語と言います) そのものは、変数名や定数名として宣言することは出来ませんが、予約語の一部を含むものは使うことができます。

### 使える例

- `let a` // 一文字の変数名も使えます

- `let computer` // 英単語も使えます
- `let computer` // つづりが間違っていても使えます
- `let player1` // 英単語の後に数字を繋げても良いです
- `let player9876` // 沢山の数字を繋げることも出来ます
- `let playerName` // 英単語を繋げるときには（慣習的に）先頭を大文字にします
- `let player_name` // (JavaScript では一般的ではありませんが) \_ も使えます
- `const GUU` // 定数を宣言する時には（慣習的に）全て大文字にします
- `const GUU_CHOKI_PAA` // いくつかの単語を繋げるときは \_ で繋ぎます
  
- `let knife` // if は予約語なので使えませんが、if を含む単語は使えます

### 使えない例

- `let 123` // 数字そのものを変数名にすることは出来ません
- `let 20th_century_pear` // 数字から始まる変数名を使うことも出来ません
- `let twentieth-century` // -(ハイフン) は使えない記号です
- `let var` // var は、予約語そのものなので使えません
- `let variable` // variable は、予約語を含んでいますがなので使えません
  
- `let if;` // if は予約語なので使えません。

## 3.6

## 演算子

JavaScript はプログラミング言語ですので、様々な計算を行うことができます。「**演算子**」とは聞きなれない用語かもしれません、平たく言えば、 $7 + 5$  の + や、 $18 - 3$  の - のように、どのような計算（演算）を行うかを示すための記号が「**演算子**」です。

### ♣ 四則演算

足し算には + を、引き算には - を使うことは、日常慣れ親しんだ算数や数学と同じです。掛け算や割り算は、キーボードに × や ÷ の記号がないので、代わりに アスタリスク \* や スラッシュ / を用います。

	演算子
加算	+
減算	-
乗算	*
除算	/
剰余	%

## ♣ 剰余演算

「剰余」演算は耳慣れないかもしれません、平たく言えば、「余りを求める」演算の事です。

$$22 \div 7 = 3 \text{ 余り } 1$$

ですが、この余りの 1 を求める演算の事です。

$$22 \% 7 = 1$$

剰余演算は周期性の有るもの種類を分類する際などに有用です。例えば、ある年の元日 (=1月1日) が月曜日だったとします。元日から七日経過した日 (=1月8日) の曜日を知る為には、

$$1 \% 7 = 1 \text{ (1 日は月曜日)}$$

$$8 \% 7 = 1 \text{ (8 日も月曜日)}$$

と、7で割った余りがどちらも 1 になるので、月曜日と知ることが出来ます。  
元日から十日経過した日 (=1月11日) の曜日を知る為には、

$$11 \% 7 = 4 \text{ (1 は月, 2 は火, 3 は水, 4 は木曜日)}$$

として、木曜日であることを求めることができます。<sup>\*8</sup>

じゃんけんの勝ち負けを効果的に判定する方法として、後ほど、「剰余演算」が登場します。お楽しみに。

## 3.7 関数

関数とは、ある一連の手続き（文の集まり）を一つの処理としてまとめる機能です。関数を利用することで、同じ処理を毎回書くのではなく、一度定義した関数を呼び出すことで同じ処理を実行できます。<sup>\*9</sup>

「関数」といえば、数学で良くある関数を思い浮かべる方もいると思います。例えば、次のような二次式（放物線）の関数です。

$$y = f(x) = x^2 - \frac{2}{5}x + 1$$

ある  $x$  を与えると、それに対応した値  $y$  を返します。

<sup>\*8</sup> 年月日から曜日を求める公式として ツェラーの公式 (<https://ja.wikipedia.org/wiki/ツェラーの公式>) がよく知られています。

<sup>\*9</sup> 例えば、勝ち負けの判定を纏めておくと、使い勝手が良さそうです。

JavaScript でも、もちろんこのような値を返す関数を定義することもできますし、また、値を返すことが目的ではなく、何か処理を行うことが目的の関数を作ることもできます。<sup>\*10</sup>

JavaScript では、関数を定義するために `function` キーワードを使います。 `function` から始まる文は関数宣言と呼び、次のように関数を定義できます。

### ▼ 関数宣言

```
// 関数宣言
function 関数名(仮引数1, 仮引数2) {
    // 関数が呼び出されたときの処理
    // ...
    return 関数の返り値
}

// 関数呼び出し
const 関数の結果 = 関数名(引数1, 引数2)
alert(関数の結果) // => 関数の返り値を表示する
```

関数は次の 4 つの要素で構成されています。

### 関数を構成する 4 つの要素

#### 関数名

利用できる名前は変数名と同じ

#### 仮引数

関数呼出時に渡された値が入る変数で、複数の場合は,(カンマ) で区切る

#### 関数の中身

「{」と「}」で囲んだ関数の処理を書く場所

#### 関数の返り値

関数を呼び出したときに、呼び出し元へ返される値

宣言した関数は、`関数名()` と関数名に括弧を付けて呼び出します。関数を引数と共に呼ぶ際は `関数名(引数)` とし、引数が複数ある場合は `関数名(引数1, 引数2)` のように、,(カンマ) で区ります。関数内では `return` 文により、関数の実行結果として任意の値を返せます。<sup>\*11</sup>

それでは、理解を深めるために、例を挙げてみましょう。とても単純ですが、`a` と `b` という二つの数を足し算する関数です。足し算(`add`)用に、`add.html` という HTML ファイルと `add.js` という JavaScript ファイルを作ります。

`add.html` は、`add.js` を呼び出すだけの HTML ファイルです。

\*10 例えば、前に登場した `alert` 関数も、メッセージ表示処理を行うことが目的の関数です。

\*11 出典：JavaScript Primer 迷わないための入門書

▼ add.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>足し算練習</title>
6     <script src="add.js" defer></script>
7   </head>
8   <body>
9     </body>
10  </html>
```

add.js は、二つの数 の足し算を行い、結果を表示するためのプログラムです。

▼ add.js

```
1 // 関数宣言
2 // 関数名は add
3 // 仮引数は、a と b の二つ
4 function add(a, b) {
5   // 関数が呼び出されたときの処理
6   // 二つの数を足して、answer という 変数に代入する。
7   let ans = a + b
8   // return文を使い、関数の返り値を、呼びだし元に返す。
9   return ans
10 }
11
12 // 関数呼び出し
13 // answer という変数を宣言して、
14 // 足し算を行う関数 add に 引数 5 と 27 を渡して、呼び出す。
15 // そして、呼び出した結果を、受け取る。
16 const answer = add(5, 27)
17 // 関数の結果を表示する
18 alert(answer)
```

32 という結果が画面に表示されました。とても簡単な例でしたが、関数の作り方、使い方は理解できましたでしょうか。

それでは、次の話題へと進んでいきましょう。



## 【コラム】新しい関数の書き方 アロー関数のご紹介

### エクマスクリプト ECMAScript

JavaScript は、Ecma International (旧欧洲電子計算機工業会 European Computer Manufacturers Association) によって仕様が定められています。第一版から始まり、第五版の ECMAScript 5th edition(ES5) まで順調に機能向上が図られてきましたが、全ての仕様を策定し第〇版として取りまとめる都合上、公開に時間を要するという課題がありました。

そのため、より迅速に公開できるよう、それぞれの仕様について合意が得られた部分から順次公開していくように方針の転換が図られました。この方針転換を受けて最初に公開されたのが「ECMAScript 2015(ES2015)」です(グレゴリオ暦に因んだ命名で、第二千十五版ではありません)。それ以降、毎年様々な仕様の追加が行われ、ES2016, ES2017, ES2018... と呼ばれています。

### ES2015

ES2015 は、従来の JavaScript(ES5) を、より近代的なプログラミング言語に生まれ変わらせるべく、大きな仕様の変更が行われました。

- アロー関数
- クラス構文
- テンプレート文字列
- 分割代入
- デフォルト引数
- 新しい変数宣言 (`let`, `const`)
- 新しい繰り返し構文 (`for of`)
- モジュール (`import`, `export`)
- 各種ライブラリの追加
- などなど

### アロー関数

ここでは、新しい関数の書き方「アロー関数」について、ご紹介いたします。(アロー関数は、関数を表す為の記号  $\Rightarrow$  が、「矢印、arrow」に見えるので、そう呼ばれます。)

二つの数 `a` と `b` を与えると、その和を返す関数 `add` を定義してみましょう。

新しく導入されたアロー関数を使う場合は、次のように書きます。

### ▼新しく導入されたアロー関数

```
// アロー関数
const add = (a, b) => {
  return a + b
}
```

従来の関数宣言では次のようにになります。

#### ▼従来の関数宣言

```
// 従来の関数宣言
function add(a, b) {
  return a + b
}
```

使い方はどちらも同じで、次のようにして結果を表示できます。

#### ▼関数の呼び出し

```
// 関数の呼び出しと結果の表示
const answer = add(5, 27)
alert(answer)
```

### 従来型の関数と「アロー関数」への変形

「変数」とは「値」が格納できる「箱」のようなものです。ですので「5」や「27」といった数値、「September」などの文字列を代入（格納）できるのは、理解しやすいでしょう。

そして、JavaScript の関数は「第一級オブジェクト」と呼ばれ、関数を変数に代入できる！性質を持ちます。5 と 27 を足し算した結果である 32 を変数に代入するのではなく、「足し算する」という「機能・働き・関数」そのものを変数に代入できるという、特筆すべき性質を持っています。

ですので、次のように変数 add に関数 tashizan を代入することができます。

```
// 変数 add に 関数tashizanを代入する
const add = function tashizan(a, b) {
  return a + b
}
```

そして使うときには、あたかも変数 add が関数であるかのように（関数を代入しているのですから全く関数と同じように）利用できます。

```
// 関数の呼び出しと結果の表示
const answer = add(5, 27)
alert(answer)
```

変数 add に代入して、「足し算をする」という機能を呼び出せるのであれば、わざわざ関数に tashizan と命名する必要も無いので、名前を省けます（「無名関数」と言います）。

```
// 変数 add に 無名関数を代入する
const add = function (a, b) {
    return a + b
}
```

`function` と毎回書くのも文字が長いので省略して、代わりに アロー関数の記法 => を`()` の次に書きます。

#### ▼ アロー関数の完成

```
const add = (a, b) => {
    return a + b
}
```

このテキストでは、`function` というキーワードの存在が、始めての方に分かりやすいかと考え、従来型の関数宣言でコーディングしています。

「アロー関数」は、従来型の関数宣言と全く同一のものではありませんが、簡潔に書けるよう、これから主流として導入され、広く用いられていますので、ぜひ使ってみてください。

またより詳しい説明が、公式サイト MDN [アロー関数式<sup>\\*12</sup>](#) にございますので、ご興味を持った方はお読みになってください。

## 3.8 JavaScript から HTML を操作する為の仕組み - DOM

わたくしたちが実現したい事は「開始ボタンを押すとメッセージを表示する」ことです。ここまでで、変数や定数、関数といったプログラミングの基本的な部品について紹介しました。そしてあと二つ学ぶべき事柄が残っています。一つは「DOM」でもう一つは「イベント」です。まず「DOM」についてご紹介いたします。

DOM とは何でしょうか。IT 用語辞典<sup>\*13</sup> の説明を見て見ましょう。

DOM とは、HTML 文書を構成する要素をコンピュータプログラムで参照したり操作したりするための取り決め。

HTML で記述されたウェブページなどの構成要素（見出し、段落、画像、リンクなど）と、それらの配置や見栄えなどを定めた属性情報などを参照、制御する手法を定めている。ウェブブラウザなどに実装されており、ページ上に JavaScript などで記述されたスクリプトからページ内の各要素を読み取ったり、内容や設定の変更、要素の追加や削除などを行う標準的な手段

\*13 <https://e-words.jp/w/DOM.html>

として用いられる。

辞典ですので、きっちりと書かれておりますが、少し難しく感じるかもしれません、使い方は簡単です。

練習用の簡単な HTML を準備して、実験して見ましょう。ファイル名は `dom.html` としましょう。

#### ▼ `dom.html`

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>DOM練習</title>
6   </head>
7   <body>
8     <button id="play">開始</button>
9   </body>
10 </html>
```

作成した `dom.html` を、ブラウザで開いて見ましょう。小さな「開始」ボタンが表示されているはずです。

次に、もう一つファイルを作成し、ファイル名は `dom.js` とします。そして、次のような JavaScript のプログラムを書きましょう。

#### ▼ `dom.js`

```
1 // HTML文書から、IDがplayである要素(=開始ボタン)を取得し、
2 // play_button という変数に格納(代入)する。
3 const play_button = document.getElementById("play")
4
5 // 取得した開始ボタンの文字を「もう一度」に更新する。
6 play_button.innerText = "もう一度"
```

この JavaScript を HTML から読み込む為に、最初に書いた `dom.html` の `<head>`タグの中に `<script src="dom.js" defer></script>` の一行を追加します。

#### ▼ `dom.html`

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>DOM練習</title>
6     <script src="dom.js" defer></script>
7   </head>
8   <body>
9     <button id="play">開始</button>
10    </body>
11 </html>
```

これで準備は完了です。

もう一度、作成した `dom.html` を、ブラウザで開いて見ましょう。先ほど表示されていた小さな「開始」ボタンに代わり、「もう一度」と表示されているはずです。

JavaScript から HTML を操作することが出来ました。

## 3.9 イベントリスナと関数

### ♣ イベントリスナ

それでは次に利用者（ブラウザを閲覧、操作する人）が、「何か操作を行った時」に、「メッセージを表示する」ようにしましょう。

JavaScript には、**イベントリスナ** と呼ばれる仕組みが有り、「何か操作を行った時」に指定した動作をする機能が備わっています。

それでは、**イベント**、**リスナー** とはなんでしょうか。それぞれの用語の意味を調べてみましょう。

プログラミングにおけるイベント (英: event) は、プログラム内で発生した動作・出来事、またそれらを表現する信号である。メッセージあるいはアクション（動作）とも呼ばれる。イベントの例としてウェブブラウザにおける「ページが読み込まれた時」、「クリック動作」、「スクロール操作」などさまざまなイベントがある。 \*14

リスナーとは、聞く人、聞き手、聴取者、聴講生、などの意味を持つ英単語。一般の外来語としてはラジオ（局/番組）の聴取者を意味する用法が有名である。

プログラミングの分野では、何らかのきっかけに応じて起動されるよう設定されたサブルーチンや関数、メソッドなどをイベントリスナー (event listener) あるいは単にリスナーという。例えば、「マウスがクリックされると起動するよう指定された関数」のことを「マウスクリックを待ち受けるリスナー」といったように呼ぶ。 \*15

つまり、**イベントリスナー** とは次のようにになります。

「ページ読み込みやクリック動作など、ウェブページで行われる様々な動作を常時起動し待ち受け（聴取し続けて）、イベントが起きた時に指定された処理を行う関数のこと」\*16

この、**イベントリスナー** の仕組みを利用して、「利用者」が「グー」を出したときに、勝ち負けを判定するなど、じゃんけんゲームを作り上げていきます。

## 3.10 開始ボタンを押すと応答するプログラム

ここまでで、お膳立てができましたので、以下のように書きましょう。

### ▼ janken10.js

```

1 // イベントリスナーの設定
2 const playButton = document.getElementById("play")
3 playButton.addEventListener("click", jankenHandler)
4
5 // ジャンケンの勝ち負けの結果を表示する関数
6 function jankenHandler(event) {
7   alert("あなたの勝ちです!")
8 }
```

一行ずつ解説していきます。

2行目の `const playButton = document.getElementById("play")` についてです。まず `document.getElementById("play")` で、HTMLに書いた開始ボタン要素を取得します(DOMという仕組みにより、JavaScriptからHTML要素を操作できました)。そして取得した開始ボタン要素をプログラムの中で扱いやすくするために `playButton` という変数を宣言し、代入しています。

3行目の `playButton.addEventListener("click", jankenHandler)` です。まず `playButton` にイベントリスナーを追加します(`addEventListener`)。画面が読み込まれた時、スクロールされた時など、さまざまなイベントがありますが、ここでは、クリック(`click`)された時に、`jankenHandler` という関数が呼ばれるようにします。<sup>\*17</sup>

6行目から8行目は、`jankenHandler` という関数を定義しています。`jankenHandler` は、Handle(車のハンドル、操舵輪、取扱者)の意味です。ジャンケンに関する事柄を取り扱うので、分かりやすいように `jankenHandler` と関数名を付けています。関数名は任意ですが、分かりやすい名前にすることで、良いプログラムを書くことができます。

`playButton` にイベントリスナーを追加したので、`playButton` がクリックされると、`jankenHandler` 関数が呼び出されて実行されるようになります。

それでは、`janken10.js` を保存して、ブラウザを再読み込みしてみましょう。「開始」ボタンを押すと、「あなたの勝ちです!」とメッセージが出るようになりました。

### キャッシュについて

ブラウザには、一度読み込んだファイルを再び読み込むことなく素早く応答できるよう「キャッシュ」機能が備わっています。

大変便利な機能ですが、新しくプログラムを変更しても、以前のコードが実行されるために、変化が見られない時があります。その対策として、

- 1) ブラウザのキャッシュを削除する
- 2) ファイル名を変更する

があります。ここではプログラムの作成が進むにつれ、段階ごとにファイル名を変更することにしましょう。

---

<sup>\*17</sup> `jankenHandler` という関数が、クリックされたときに呼ぶ声に応答できるよう、耳を澄ましてラジオを聴取しているイメージを持つと理解しやすいでしょう。

janken00.js から janken10.js へとファイル名を変更しています。

呼び出し元となる index.html のコードも

▼ index.html

```
<!-- JavaScriptの読み込み -->
<script src="janken00.js" defer></script>
```

から、次のように変更しましょう。

```
<!-- JavaScriptの読み込み -->
<script src="janken10.js" defer></script>
```

.....

## 3.11

## 繰り返し処理

### ♣ 繰り返しの基本構文

三回、勝利メッセージを表示したいときは、どのようにすれば良いでしょうか。

```
// ジャンケンの勝ち負けの結果を表示する関数
function jankenHandler(event) {
    alert("あなたの勝ちです!")
    alert("あなたの勝ちです!")
    alert("あなたの勝ちです!")
}
```

このように三回書いたらできます。それでは百回表示させたいときはどうでしょうか。エディタのコピー機能を使うこともできますが、それでも百回書くのは大変です。

プログラミング言語には、同じ処理を繰り返したいときの為の専用の構文(書き方)が用意されています。for文を使い、次のように書いてみましょう。

```
// ジャンケンの勝ち負けの結果を表示する関数
function jankenHandler(event) {
    for(let i = 0; i < 3; i++) {
        alert("あなたの勝ちです!")
    }
}
```

三回、勝利メッセージが表示されました。上のプログラム中、3と書かれているところを、100と書き換えると、百回表示させることもできます。<sup>\*18</sup>

<sup>\*18</sup> アラートループ事件 (<https://ja.wikipedia.org/wiki/アラートループ事件>) 兵庫県警によりウイルス罪として女子中学生ら五人が捕えられた誤認・冤罪事件。JavaScriptの生みの親であるブレンダン・アイクは「Chromeの

`for` 文は繰り返す範囲を指定した反復処理を書くことができます。名著「JavaScript Primer 迷わないための入門書」から、`for` 文の説明を見てみましょう。

#### ▼ for 文

```
for (初期化式; 条件式; 増分式) {
    実行する文;
}
```

`for` 文の実行の流れは次のようにになります。

1. 初期化式 で変数の宣言
2. 条件式 の評価結果が `true`(真・成り立つ) なら次のステップへ、`false`(偽・成り立たない) なら終了
3. 実行する文 を実行
4. 増分式 で変数を更新
5. ステップ 2 へ戻る

### ♣ 繰り返し処理で合計を求める

`for` 文の例として、1 から 10 までの値を合計して、その結果を出力するプログラムを作ってみましょう。今まで作って来た `janken10.js` の最後に追加してみましょう。

#### ▼ 1 から 10 までの合計を求めるプログラム

```
// totalの初期値は0
let total = 0

// for文の実行の流れ
// まず、iを1で初期化
// iが10以下（条件式を満たす）ならfor文の処理を実行
// iを一つ増加させ(i++)、再び条件式の判定へ
for (let i = 1; i <= 10; i++) {
    // 1から10の値をtotalに加算している
    total = total + i
}

// コンソールに55が出力される
console.log(total)
```

「コンソール」という新しい言葉が登場しました。「操作盤」という意味をもつ英単語です。`console.log()` という命令(メソッド、関数)を使って、ブラウザのコンソール画面に結果を表示できます。

---

初版よりも 10 年も前にリリースされた Netscape 4 でもユーザーがループする JavaScript をキルすることができます。」「この事件の公判で専門家証人になるつもりでいる。これはウイルスなどではなく、犯罪扱いされるべきではない。」とツイートした。

Firefox では、次のようにすると、コンソール画面と呼ばれる画面を表示できます。

1. ブラウザ画面を右クリック
2. 「調査」をクリック
3. 「コンソール」パネルをクリック

コンソール画面に 55 と出力されています。

## じゃんけんゲーム

みんな知っているじゃんけん。グーは 0, チョキは 1, パーは 2 を入れてね。

0  開始



## 3.12 条件分岐

じゃんけんは、勝つこともあります、負けることもあります。条件分岐を使うと、特定の条件を満たすかどうかで行う処理を変更できます。

この章では **if 文** を使った条件分岐について学んでいきます。  
if 文 は次のような構文が基本形となります。条件式の評価結果が **true** (真・成り立つ) \*19 であるならば、実行する文が実行されます。

### ▼ if 文

```
if (条件式) {
    実行する文
}
```

**もし、条件式が真(成り立つ)ならば、波括弧{ }内に書かれた文を実行する** という意味です。

例として、映画館の料金を表示するプログラムを考えてみましょう。

次のコードでは条件式が **true** (真・成り立つ) であるため、 **if 文** の中身が実行され、コンソール画面に「大人料金です」と表示されます。

```
1 // 年齢は20歳
2 let age = 20
3
4 if (age >= 18) {
5     console.log("大人料金です")
6 }
```

複数の条件分岐を書く場合は、 **if 文** に続けて **else if 文** を使うことで書けます。たとえば、次のように 3 つに条件分岐するプログラムを書けます。

\*19 反対語は **false** (偽・成り立たない) です。

```

1 // 年齢は10歳
2 let age = 10
3
4 if (age >= 18) {
5   console.log("大人料金です")
6 } else if (age >= 12) {
7   console.log("中高生料金です")
8 } else if (age >= 6) {
9   console.log("小学生料金です")
10 }

```

`if` 文と `else if` 文では、条件に一致した場合の処理をブロック内に書いていました。いずれの条件にも一致しなかった場合の処理は、`else` 文を使うことで書けます。

```

1 // 年齢は5歳
2 const age = 5
3
4 if (age >= 18) {
5   console.log("大人料金です")
6 } else if (age >= 12) {
7   console.log("中高生料金です")
8 } else if (age >= 6) {
9   console.log("小学生料金です")
10 } else {
11   console.log("無料です")
12 }

```

小学生未満無料とは、気前のよい映画館ですね。

それでは本題のじゃんけん機能を充実させていきましょう。勝負のドキドキが味わえるようにしていきます。

そのために、`if` 文を使って、以下のように書いてみましょう。(キャッシュの影響を避けるために、ファイル名を `janken30.js` にしています。`index.html` でも `<script src="janken30.js" defer></script>` ）と更新しましょう。

#### ▼ janken30.js

```

1 // イベントリスナの設定
2 // 開始ボタンを押されるとゲーム開始
3 const playButton = document.getElementById("play")
4 playButton.addEventListener("click", jankenHandler)
5
6 // player の手を取得
7 const inputBox = document.getElementById("player_hand_type")
8 let player = Number(inputBox.value)
9
10 // computer の手を設定
11 let computer = 0 // グー
12

```

```

13 // ジャンケンの勝ち負けの結果を表示する関数
14 function jankenHandler(event) {
15   // === は「厳密等価演算子」で、「等しい」ことを調べます。
16   if (player === 0) {
17     // プレイヤーがグーの時に行う処理を記します。
18     // ここでは、alert文を使い、画面表示します。
19     alert("あいこです!")
20   } else if (player === 1) {
21     // プレイヤーがチョキの時の処理を記します。
22     alert("あなたの負けです!")
23   } else {
24     // プレイヤーがパーの時の処理を記します。
25     alert("あなたの勝ちです!")
26   }
27 }
```

4行目までは、前回と一緒です。

7行目と8行目で、プレイヤーが入力した手を取得します。index.html には `<input type="number" id="player_hand_type" value="0">` とプレイヤーが自分の手（グーなら0、チョキなら1、パーなら2）を入力するための「入力枠」要素を設けてあります。

この「入力枠」にどのような手（数値）が入力されたのかを JavaScript で取得し、プレイヤーの手（数値）に依って、勝敗を判定、結果を表示するようにします。

まず `document.getElementById("player_hand_type")` と書き、このプレイヤーの入力枠要素自身を JavaScript で取得します。取得した入力枠に名前があると扱いやすいので、`const inputBox` と書き `inputBox` という名前の変数を用意します。そして `const inputBox = document.getElementById("player_hand_type")` と代入演算子 `=`<sup>イコール</sup> を使って、取得したプレイヤーの入力枠要素を `inputBox` 変数に代入します。これで `inputBox` 変数がプレイヤーの入力枠を指すようになりましたので、`document.getElementById("player_hand_type")` と書く代わりに、短く `inputBox` という名称（変数名）で使うことができます。

`inputBox` から プレイヤーが枠に入力した値を取得するには、`value`メソッドを使います。`inputBox.value` と書くとプレイヤーが枠に入力した値を得ることができます。

`Number` は、文字列としての "0" を数値としての 0 にする関数です。一般にプログラミング言語では、文字列としての "0" と、整数値としての 0 とは異なります（ちょうど、漢数字の〇と、数値の 0 が異なるようにです）。整数値に変換した結果を、`let player =` と書いて、代入しています。

以上で、プレイヤーが入力した手を取得することができるようになりました。

11行目の、`let computer = 0` は、コンピュータの手を設定しています。

0はグー、1はチョキ、2はパー という約束でした。今のところ、コンピュータは必ず「グー」

を出すことにして作っていきましょう。

プレイヤーの手とコンピュータの手が出そろったので、先ほど学んだ `if` 文を使って、勝ち負けの判定を行います。

16行目以降で、勝ち負けを判定して、それに応じたメッセージを表示させています。

0を入力した場合、1を入力した場合、2を入力した場合、それぞれで、正しくメッセージが表示されるか確認してみてください。(メッセージが表示されたらブラウザを再読み込みして違う手を入力してください。)

ぐっと、じゃんけんゲームらしくなってきましたね。

## 3.13 亂数の利用

### ♣ 亂数を使う

さて、コンピュータの手は「0」つまりいつも「グー」でしたので、プレイヤーは「2」つまりパーを出せば、必ず勝てます。コンピュータも無作為に「グー」「チョキ」「パー」の手を出すようにしましょう。

そのためには、「乱数」と呼ばれる機能を使います。JavaScriptには`Math.random()`と呼ばれる関数が用意されています。`Math.random()`を呼ぶと、0から1未満の浮動小数点数を返します。<sup>\*20</sup>

```
Math.random()
0.9200533064823014
0.5017996980638613
0.15088182883224843
```

わたくしたちが欲しいのは、0, 1, 2 の3つの数ですから、三倍すれば良さそうです。

```
Math.random() * 3
2.760159919446904
1.5053990941915838
0.4526454864967453
```

小数点以下は不要ですから、切り捨てましょう。JavaScriptには`Math.floor()`と呼ばれる切り捨ての為の関数が用意されています。(床関数と呼ばれます。`Math.ceil()`は天井関数で、切り上げ、`Math.round()`は四捨五入です。)

```
Math.floor(Math.random() * 3)
2
```

<sup>\*20</sup> 表示は一例で、「乱数」なので実行するたびに値は変わります。

```
1
0
```

```
// computer の手を 亂数で設定
let computer = Math.floor(Math.random()*3)
```

### 乱数関数を自作する

関数化して分かりやすい名前を付けると、一目でプログラムの処理を理解できる、作成したコードの再利用が図れるなどの利点があります。有益ですので、ここで作ってみましょう。

```
// 亂数関数
// rand(0, 2) と呼ぶと 0, 1, 2 と グーチョキパー の乱数を返す
function rand(min, max){
    return Math.floor(Math.random() * (max - min + 1)) + min
}

function rand(min, max){
    return Math.floor(Math.random() * (max - min + 1)) + min
}
```

この乱数関数を使うと、無作為な手を取得するコードは、次のように書くことができます。

```
// computer の手を 亂数で設定
let computer = rand(0, 2)
```

すっきり、分かりやすくなりました。いろいろ関数を自作してみて下さい。

## 3.14 入れ子になった if 文

### ♣ 入れ子になった if 文

コンピュータが無作為な手を出すようになったので、いつもパーを出して勝ちというわけにはいかなくなりました。勝敗判定も書き直す必要があります。

プレイヤーのグーチョキパー、それぞれについて、コンピュータのグーチョキパー、全部で九通りの勝敗判定が必要です。if 文 の中に if 文 を書くことができます。入れ子になった if 文、ネストした if 文 といいます。次のように、書き直してみましょう。

#### ▼ janken40.js

```
1 // イベントリスナの設定
2 // 開始ボタンを押されるとゲーム開始
3 const playButton = document.getElementById("play")
```

```
4 playButton.addEventListener("click", jankenHandler)
5
6 // player の手を取得
7 const inputBox = document.getElementById("player_hand_type")
8 let player = Number(inputBox.value)
9
10 // computer の手を 亂数で設定
11 let computer = rand(0, 2)
12
13 // ジャンケンの勝ち負けの結果を表示する関数
14 function jankenHandler(event) {
15     if (player === 0) {
16         // === は「厳密等価演算子」で、「等しい」ことを調べます。
17         // プレイヤーがグーの時なら
18         if (computer === 0) {
19             // コンピュータがグーを出した場合、
20             alert("あいこです!")
21         } else if (computer === 1) {
22             // コンピュータがチョキを出した場合
23             alert("あなたの勝ちです!")
24         } else {
25             // コンピュータがパーを出した場合
26             alert("あなたの負けです!")
27         }
28     } else if (player === 1) {
29         // プレイヤーがチョキの時に、
30         if (computer === 0) {
31             // コンピュータがグーを出した場合
32             alert("あなたの負けです!")
33         } else if (computer === 1) {
34             // コンピュータがチョキを出した場合
35             alert("あいこです!")
36         } else {
37             // コンピュータがパーを出した場合
38             alert("あなたの勝ちです!")
39         }
40     } else {
41         // プレイヤーがパーの時に、
42         if (computer === 0) {
43             // コンピュータがグーを出した場合
44             alert("あなたの勝ちです!")
45         } else if (computer === 1) {
46             // コンピュータがチョキを出した場合
47             alert("あなたの負けです!")
48         } else {
49             // コンピュータがパーを出した場合
50             alert("あいこです!")
51         }
52     }
53 }
54
55 // 亂数関数 rand(0, 2)と呼ぶと 0, 1, 2 と グーチョキパー の乱数を返す
56 function rand(min, max){
```

```

57   return Math.floor(Math.random() * (max - min + 1)) + min
58 }
```

それでは、ブラウザを再読み込みしてみましょう。うまく動いてくれるでしょうか。

## 3.15 リファクタリング（再構成）

だいぶプログラムが長くなってしまった。より見通しの良い、分かりやすいコードとするために、ここで「リファクタリング」を行いましょう。

リファクタリングとは、ソフトウェア開発において、プログラムの動作や振る舞いを変えることなく、内部の設計や構造を見直し、コードを書き換えたり書き直したりすること。<sup>\*21</sup>

主なリファクタリング手法として、**定数**の利用、**関数化**、**アルゴリズム改善**等が、よく用いられます。順に行っていきましょう。

### ♣ 定数の利用

**定数**とは、コンピュータプログラムのソースコードなどで、ある特定の数値やデータに名前を与えたもので<sup>\*22</sup>、慣習的に定数名は全て大文字で書かれます。

コンピュータにとって、0, 1, 2 は分かりやすくて扱いやすいですが、人にとっては、グー、チョキ、パー のほうが分かりやすいものです。単なる数値に名前を付けることで、理解しやすいコードを書くことができるようになります。

```

// 定数宣言
// プログラム内で共通して使う定数を宣言する。
// 慣習的に定数名は全て大文字で書かれる。
const GUU = 0 // グー
const CHOKI = 1 // チョキ
const PAA = 2 // パー
```

と、定数を宣言します。

すると、先程の if 文 は次のように書き直せます。

```

// じゃんけんの勝ち負けの結果を表示する関数
function jankenHandler(event) {
  if (player === GUU) {
    if (computer === GUU) {
      alert("相手です!")
    } else if (computer === CHOKI) {
      alert("あなたの勝ちです!")
    } else {
      alert("あなたの負けです!")
```

```

    }
} else if (player === CHOKI) {
    if (computer === GUU) {
        alert("あなたの負けです!")
    } else if (computer === CHOKI) {
        alert("相手です!")
    } else {
        alert("あなたの勝ちです!")
    }
} else {
    if (computer === PAA) {
        alert("あなたの勝ちです!")
    } else if (computer === CHOKI) {
        alert("あなたの負けです!")
    } else {
        alert("相手です!")
    }
}
}

```

`0, 1, 2`といった数値から、`GUU, CHOKI, PAA`という定数に変わりました。プログラムの機能は変わりませんが、単なる数値ではなく、人にとって理解しやすい意味を持つ`GUU, CHOKI, PAA`という文字になったので、コメントも不要となるほどとても読み易いコードとなりました。

## ♣ 関数化とアルゴリズムの改善

じゃんけんの勝敗判定部分も長く成りましたので、この部分を取り出して関数にすることにより、全体を見やすくしましょう。

関数とは、ある一連の手続き（文の集まり）を1つの処理としてまとめる機能です。関数を利用することで、同じ処理を毎回書くのではなく、一度定義した関数を呼び出すことで同じ処理を実行できます。また、一度しか行わない処理でも、適切な命名を行って、処理の詳細に関する部分をプログラムの呼び出し元から分離することで、コード全体の見通しが良くなる利点が得られます。

また、より良いアルゴリズムを考えることで、9通りの`if`文を綺麗に書き直していきましょう。

## 3.16 じゃんけんのアルゴリズム

じゃんけん勝敗判定アルゴリズムの思い出<sup>23</sup> というブログがあります。こちらを参考に、より簡潔に書けるよう、じゃんけんのアルゴリズムを考察していきましょう。

素直に`if`文を書くと、プレイヤーがグーの時、チョキの時、パーの時のそれにつき、コン

<sup>23</sup> <https://staku.designbits.jp/check-janken/>

ピュータがグーの時、チョキの時、パーの時と、九通りの勝敗判定が必要でした。

`if` 文を書き連ねず、もう少し簡潔に書けるか調べるために、勝敗表にまとめます。

じゃんけんの勝敗表

	グー 0	チョキ 1	パー 2
グー 0	相手	勝ち	負け
チョキ 1	負け	相手	勝ち
パー 2	勝ち	負け	相手

自分の手と相手の手が等しい時に「相手」になることが分かります。等しいかどうかを調べるために、**引き算してその結果が 0 になるか**で判定できますので、**自分の手から相手の手を引き算**してみます。すると次の表が得られます。

じゃんけんの勝敗表 【引き算】

	グー 0	チョキ 1	パー 2
グー 0	相手 0	勝ち -1	負け -2
チョキ 1	負け 0	相手 0	勝ち -1
パー 2	勝ち 2	負け 1	相手 0

相手になるのは 0 の時、負けになるのは -2 か 1 の時、勝ちになるのは -1 か 2 の時であることが判明しました。これで九通りではなく、五通りの `if` 文で良いと分かりました。

もう少し考察を加えます。勝ち負け相手の三通りの判定をする為に「本当に」五通りの `if` 文が必要でしょうか。

よく見ると -2 と 1 は 3 つ離れており、-1 と 2 も 3 つ離れていますので、3 を足してみます。すると、

- 相手になるのは、0 か 3 の時、
- 贠けになるのは、1 か 4 の時、
- 勝ちになるのは、2 か 5 の時となります。

なにか法則性がありそうです。もう少し 3 を足してみます。

- 相手になるのは、0 か 3 か 6 か 9 の時、
- 贠けになるのは、1 か 4 か 7 か 10 の時、
- 勝ちになるのは、2 か 5 か 8 か 11 の時となります。

法則性が見えてきたでしょうか？

- 相手になるのは、3 の倍数の時、
- 贠けになるのは、3 の倍数に 1 を足した数の時、
- 勝ちになるのは、3 の倍数に 2 を足した数の時 のようです。

3 の倍数であるか調べるにはどうしたら良いでしょうか？

- 3で割ってみて、余りが0であれば、3の倍数です。

3の倍数に1を足した数であるか調べるにはどうしたら良いでしょうか？

- 3で割ってみて、余りが1であれば、3の倍数に1を足した数です。

3の倍数に2を足した数であるか調べるにはどうしたら良いでしょうか？

- 3で割ってみて、余りが2であれば、3の倍数に2を足した数です。

以上の考察から、

- 相子になるのは、3で割って余りが0の時、
- 負けになるのは、3で割って余りが1の時、
- 勝ちになるのは、3で割って余りが2の時であることが分かりました。

余りを求める演算のことを「**剰余演算**」と言います。

JavaScriptでは、%を使うと、剰余演算を行うことができます。

	演算子
加算	+
減算	-
乗算	*
除算	/
剰余	%

まとめです。

- 勝負の判定には、最初は九通りのif文が必要でした。
- 自分の手 - 相手の手とすると、五通りになりました。
- (自分の手 - 相手の手 + 3) % 3として、0, 1, 2の三通りになりました。

それでは、相子が0、負けが1、勝ちが2と、結果を返す関数を作りましょう。

```
// プレイヤーの手とコンピュータの手が与えられると、
// 0: 引き分け 1: 负け 2: 勝ち を返す関数
function judge(player, computer) {
    return (player - computer + 3) % 3
}
```

延々とif文を繰り返していた長い行が、とっても短く纏まりました。

それでは、ここで定義したjudge関数を使って、プログラムを書き直してみましょう。せっかくですので「相子」「負け」「勝ち」を表す定数も使いましょう。次のようにとても分かりやすくなりました。

#### ▼ janken60.js

```
1 // 定数宣言
2 // プログラム内で共通して使う定数を宣言する。
3 // 慣習的に定数名は全て大文字で書かれる。
4 const DRAW = 0 // あいこ
```

```

5 const LOSE  = 1 // 負け
6 const WIN   = 2 // 勝ち
7
8 const GUU   = 0 // グー
9 const CHOKI = 1 // チョキ
10 const PAA   = 2 // パー
11
12 // イベントリスナの設定
13 // 開始ボタンを押されるとゲーム開始
14 const playButton = document.getElementById("play")
15 playButton.addEventListener("click", jankenHandler)
16
17 // メイン処理
18 // player の手を取得
19 const inputBox = document.getElementById("player_hand_type")
20 let player = Number(inputBox.value)
21
22 // computer の手を 亂数で設定
23 let computer = rand(0, 2)
24
25 // じゃんけんの勝ち負けの結果を表示する関数
26 function jankenHandler(event) {
27     // judge関数に、プレイヤーとコンピュータの手を渡して、
28     // 勝敗(相手なら0 , 負けなら1, 勝ちなら2)を得ます。
29     const result = judge(player, computer)
30
31     if (result === DRAW) {
32         alert('引き分けです!')
33     } else if (result === LOSE) {
34         alert('あなたの負けです!')
35     } else {
36         alert('あなたの勝ちです!')
37     }
38 }
39
40 // 亂数関数 rand(0, 2)と呼ぶと 0, 1, 2 と グーチョキパー の乱数を返す
41 function rand(min, max){
42     return Math.floor(Math.random() * (max - min + 1)) + min
43 }
44
45 // プレイヤーの手とコンピュータの手が与えられると、
46 // 0: 引き分け 1: 負け 2: 勝ち を返す関数
47 function judge(player, computer) {
48     return (player - computer + 3) % 3
49 }
```

以上で、簡素なじゃんけんゲームは完成です。短いコードですが、じゃんけんゲームの基本が詰まっています。

- 定数を使って、読みやすいコードにする。
- イベントリスナを設定、開始ボタンを押すと始まる。
- プレイヤーの入力した手を取得する。
- コンピュータも乱数を用いて無作為な手を出す。
- 勝敗判定関数の作成。
- 9通りの if 文からアルゴリズムの見直しを行い、わずか一行の判定式に改良。

# 第 4 章

## 初めての CSS

この章では、じゃんけんゲームの意匠を整てるために用いる CSS の基礎について学びます。また、利用者が快適に使えるよう、UI/UX の観点をご紹介します。

これまで、文書構造を HTML で、利用者の入力に応じた処理を JavaScript で書いてきました。これにより、じゃんけんゲームの大まかな機能の実装は完了いたしました。

せっかくのじゃんけんゲームですから、CSS を使って綺麗に意匠も整えて、より楽しめるゲームにしていきたいところですので、CSS の基礎について学びましょう。また、UI/UX の観点を知り、どのように意匠を整えると良いか、考えていきましょう。

### 4.1 ユーザインターフェース (UI) と 利用者体験 (UX)

ユーザインターフェース (User Interface) は「ユーザーインターフェイス」とも表記されます。どのような意味か、いつものように「IT 用語辞典」を参照し、確認してみましょう。

ユーザーインターフェースとは、機器やソフトウェア、システムなどとその利用者の間で情報をやり取りする仕組み。システムから利用者への情報の提示・表示の仕方と、利用者がシステムを操作したり情報を入力したりする手段や方式、機器、使い勝手などの総体を表す。

ディスプレイ装置などの画面表示、マイクやスピーカー、イヤフォンによる音声入出力、キーボードなどによる文字入力、マウスやペンタブレット、タッチパネルなどによる（画面上の）位置や方向の入力、カメラなどによる画像・映像入力、およびこれらの組み合わせによって構成されることが多い。

とあります。人とコンピュータの間にある窓口であり、利用者がコンピュータの状態を把握し操作しやすいよう、画面上には、アイコンやメニュー、ボタンといった操作要素が表示されています。利用者はこれらのアイコンをクリックすることで、コンピュータを操作できます (GUI, Graphical User Interface)。

じゃんけんの絵を表示し、入力用のボタンを用意することで、快適なユーザインターフェースを提供し、便利に使ってもらえるようにしましょう。

また昨今では「利用者体験 (User eXperience)」も重視されるようになってきました。こちらも「IT用語辞典」を参照し、確認してみましょう。

UXとは、ある製品やサービスとの関わりを通じて利用者が得る体験およびその印象の総体。使いやすさのような個別の性質や要素だけでなく、利用者と対象物の出会いから別れまでの間に生まれる経験の全体が含まれる。

対象物の機能や性能、内容、使い勝手といった性質そのものよりも、それを通じて利用者が得られる経験がどのようなものであるかに着目する概念である。対象物の持つ特性だけでは決まりず、利用者側の属性や個性、利用者を取り巻く環境や利用時の状況などにも強く影響を受けるため、作り手側ですべてを制御することは難しい。

「作り手側ですべてを制御することは難しい」とあるように、良い「利用者体験」の実現はなかなか大変なものがございますが、できる限りのこととして、利用者に気持ちよく楽しかったと感じてもらえるよう、完成形は次のようにしていきましょう。

1. コンピュータが出した手を表示する機能がありませんでしたので、グーチョキパーのいずれなのかを示すアニメーション機能を実装します。
2. プレイヤーが出す手を、0, 1, 2と、数字で入力するのではなく、グーチョキパーの3つのボタンを押すことで、選べるようにします。
3. ○勝○敗と、今までの勝敗を表示できるようにします。
4. 開始ボタンなどを分かりやすくするために、色を付け、大きくします。

## じゃんけんゲーム



みんな知っているじゃんけん  
グー・チョキ・パーどれかを押してね



0 勝 0 敗

開始

## 4.2 完成形の HTML

UI/UX に考慮し、次のように index.html を更新します。これが完成形の HTML です。

### ▼ index.html

```

1 <!DOCTYPE html>
2 <html>
3   <!-- head には、文書に関する設定事項を書きます。 -->
4   <head>
5     <!-- 文字コードは utf-8 を使います -->
6     <meta charset="utf-8">
7     <!-- このページのタイトルを示すタグで、ブラウザのタブに表示されます。-->
8     <title>じゃんけんゲーム</title>
9     <!-- ビューポート(視点)を設定し、iPhoneでも綺麗に見られるようにします。 -->
10    <meta name="viewport" content="width=device-width">
11    <!-- スタイルシートを読み込み、サイトを飾り付けします。-->
12    <link rel="stylesheet" href="janken.css">
13    <!-- JavaScriptの読み込み、じゃんけんゲームを動かします。 -->
14    <script src="janken.js" defer></script>
15  </head>
16
17  <!-- body タグには、サイトのコンテンツ（内容・出し物）を記述します。
18      全ての要素は このbodyタグの中に書きます。-->
19  <body>
20    <!-- header タグは、サイト名などの表示に用います。 -->
21    <header>
22      <!-- h1 タグは、大見出しを表すタグです。 -->
23      <h1>じゃんけんゲーム</h1>
24    </header>
25
26    <!-- mainタグは
27        ページの主題(main)となるコンテンツ(内容・出し物)を記述します。-->
28    <main>
29      <!-- figure タグは 図版(figure)を示すタグです。 -->
30      <figure>
31        <!-- img タグは、絵(image)を表示するためのタグです。 -->
32        
33      </figure>
34
35      <!-- p は 段落(paragraph)を表す際に用いるタグです。 -->
36      <p>
37        <!-- br タグを入れると、途中で改行できます。 -->
38        みんな知っているじゃんけん<br>
39        ゲー・チョキ・パー どれかを押してね
40      </p>
41
42      <!-- div は汎用的に使えるタグです。
43          クラス名をcontrol_areaとし、配置を整え易くします。 -->
44      <div class="control_area">
45        <!-- プレイヤーの手を選ぶ為のボタンです。 -->
46        <!-- button タグを使って、画面に「ゲーチョキパー」のボタンを表示します。
47            JavaScriptでの操作用に id属性をguu 値を0 にします。 -->
```

```

48 <button id="guu"    value="0"></button>
49 <button id="choki"  value="1"></button>
50 <button id="paa"   value="2"></button>
51 <!-- 得点領域です -->
52 <p class="score">
53     <!-- JavaScriptから操作しやすいよう、id属性をwinとloseにします。 -->
54     <span id="win">0</span> 勝
55     <span id="lose">0</span> 敗
56 </p>
57 <!-- button タグを使って、画面に「開始」ボタンを表示します。
58     このボタンにも、id="play" と ID を付与しておきます。 -->
59 <button id="play">開始</button>
60 </div>
61 </main>
62 </body>
63 </html>

```

いろいろな変更が追加されていますので、大まかに解説していきます。

### ♣ ビューポート(視点)の設定

ビューポート(視点)を設定しています。ウェブサイトを見るための様々な端末(デバイス)がありますが、様々な画面幅の端末でも綺麗に見られるにするための記述です。

```

9 <!-- ビューポート(視点)を設定し、iPhoneでも綺麗に見られるようにします。 -->
10 <meta name="viewport" content="width=device-width">

```

### ♣ スタイルシートの読み込み

```

11 <!-- スタイルシートを読み込み、サイトを飾り付けします。-->
12 <link rel="stylesheet" href="janken.css">

```

スタイルシートを読み込むためのコードです。`janken.css` ファイル内にいろいろな意匠(文字や画像の大きさや色、配置など)を記述し、ウェブサイトの見た目を整えていきます。

### ♣ 画像ファイルの指定

```

31 <!-- img タグは、絵(image)を表示するためのタグです -->
32 

```

`img` タグを使うことで、ウェブサイト上に画像を表示することができます。`src="guu.webp"`として、グーの絵を表示させています。

## ♣ id 属性

```
32 
```

`id="computer_hand_type"` と `id` 属性 を付与しています。 `id` 属性 はページ内で一つのみ存在する要素に対して用います。コンピュータが、どの手を出すかによって、グー、チョキ、パー、三つの画像を変更する必要があります。JavaScript から画像を区別しやすくするために `id` 属性 を付与しています。

## ♣ class 属性

```
44 <!-- div は汎用的に使えるタグです。
45   クラス名をcontrol_areaとし、配置を整え易くします -->
46 <div class="control_area">
```

`class="control_area"` と `class` 属性 を付与しています。 `id` 属性 と異なり、`class` 属性 はページ内で複数要素が存在しても構いません。CSS で要素を分類 (Classify、分類分け) し、一括してスタイルを適用する目的で幅広く用いられます。

## ♣ button 要素

```
48 <!-- プレイヤーの手を選ぶ為のボタンです 。 -->
49 <!-- button タグを使って 、画面に「グーチョキパー」のボタンを表示します 。
50   JavaScriptでの操作用に id属性をguu 値を0 にします 。 -->
51 <button id="guu"  value="0"></button>
52 <button id="choki" value="1"></button>
53 <button id="paa"   value="2"></button>
```

UI 改善のため、ボタン要素を導入します。後ほど CSS によりグーの絵を表示させます。今まで入力枠に 0 などと数値で入力していましたが、絵を見てボタンを押せば良いので、とても使いやすくなります。

グー・チョキ・パーのボタンを押した時に JavaScript でどのボタンが押されたのか分かるよう、`value="0"` と記述しています。後ほど、この 0 を受け取って処理をするよう、JavaScript を改修していきます。

## 4.3 CSS の基本

### ♣ 現状把握

それでは、HTML 文書に、飾り付けを行う前に、現状をみておきましょう。左は CSS を適用していない現在の状況、右は CSS を提要した後の状況です。



▲図 4.1: CSS 適用前 (左) 適用後 (右)

## ♣ 初めての CSS

見出しとなる「じゃんけんゲーム」を見てみましょう。次のような違いがあります。

### CSS 適用前後の違い

	CSS 適用前	CSS 適用後
文字の大きさ	大きい	とても大きい
文字の色	黒	黒羽色
文字の装飾	何もなし	影が付いている
文字の配置	左端から配置	中央揃えされている

この装飾を与えてるのが次の CSS コードです。

### ▼見出しのための CSS

```

1 header h1 {           /* header 内の h1 (大見出しの指定) */
2   font-size: 40px;      /* 書体の大きさは40px */
3   color: var(--kurohairo); /* 文字色を黒羽色にする */
4   text-shadow: 3px 4px 5px var(--nibiro); /* 鈍色の影を付ける */
5   text-align: center;    /* 文字は中央揃えにする */
6 }
```

CSS は、HTML 文書のどの要素に装飾を与えるのかを、「セレクタ (選択子)」で指定します。`header h1` と書いているので、HTML 文書の `<header>` の中にある `<h1>` に対して装飾が適用されます。

具体的な装飾内容は、`{}` の中に書きます。

HTML の各要素に対してどのような装飾が設定可能かが規定されていますが、`<h1>` 要素へは「書体の大きさ」を設定することができます。どのようなスタイル付け・装飾方法は「プロパティ」と呼ばれています。「書体の大きさ」を指定するには、`font-size` プロパティを用いられます。

そして、「書体の大きさ」をどのくらいにするのか、値を設定する必要があります。この設定する値のことを「プロパティ値」と呼びます。

プロパティとプロパティ値の間は、`:(コロン)` で区切り、最後に`;(セミコロン)` で終了します。

まとめると、`font-size: 40px;` と書くことで、書体の大きさを `40px` に設定できます。

最後に「コメント」があります。CSS のコメントは `/* */` の間に書くことができます。ここでは、`/* 書体の大きさは 40px */` と、説明書きを入れています。一行ごとにコメントする必要はありませんが、要所要所のコメントで読みやすくなりますので、活用してください。

## ♣ CSS とは何か

公式サイト MDN に、[CSS の基本<sup>\\*1</sup>](#) という、優れた記事がございます。

CSS (Cascading Style Sheets) は、ウェブページのスタイルを設定するコードです。CSS は、HTML の要素を選択的にスタイル付けするために使用するものです。例えば、この CSS は段落のテキストを選択し、色を赤に設定しています。

```

p {
  color: red;
}
```

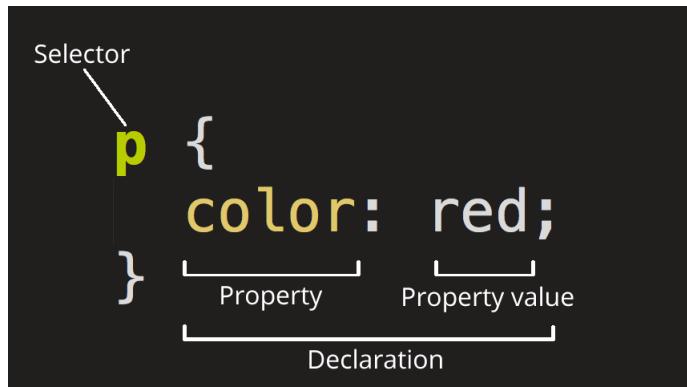
<sup>\*1</sup> [https://developer.mozilla.org/ja/docs/Learn/Getting\\_started\\_with\\_the\\_web/CSS\\_basics](https://developer.mozilla.org/ja/docs/Learn/Getting_started_with_the_web/CSS_basics)

## ♣ CSS 規則セットの構造

赤い段落テキストの CSS コードを分解して、その仕組みを理解してみましょう。

全体の構造は規則セットと呼びます（規則セットという語はよく、単に規則とも呼ばれます）。

それぞれの部分の名前にも注意してください。



### セレクタ (Selector)

これは規則セットの先頭にある HTML 要素名です。これはスタイルを設定する要素（この例の場合は `<p>` 要素）を定義します。別の要素をスタイル付けするには、セレクタを変更してください。

### 宣言 (Declaration)

`color: red;` のような単一の規則です。これは要素のプロパティのうち、スタイル付けしたいものを指定します。

### プロパティ (Property)

これらは、HTML 要素をスタイル付けするための方法です。（この例では、`color` は `<p>` 要素のプロパティです。）CSS では、規則の中で影響を与えるプロパティを選択します。

### プロパティ値 (Property value)

プロパティの右側には `:`（コロン）の後に プロパティ値 があります。与えられたプロパティの多くの外観から一つを選択します。（例えば、`color` の値は `red` 以外にもたくさんあります。）

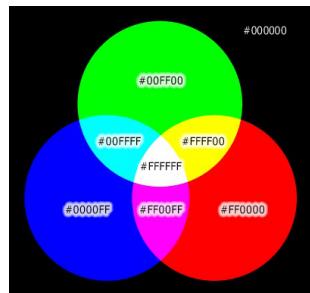
構文の他の重要な部分に注意してください。

- セレクタを除き、それぞれの規則セットは中括弧 `({})` で囲む必要があります。
- それぞれの宣言内では、コロン `(:)` を使用してプロパティと値を分離する必要があります。
- それぞれの規則セット内でセミコロン `(;)` を使用して、それぞれの宣言と次の規則を区切る必要があります。

## ♣ CSS での色指定

色を指定する際に `color: red;` と、色の名前を書きましたが、CSS では、光の三原色 赤 (Red)・緑 (Green)・青 (Blue) の組み合わせで色を表現する「RGB カラー」が良く用いられます。<sup>a</sup>

光を重ねるほど明るくなり、三色が重なる部分は白になります（加法混色）。色の三原色と呼ばれる 黄・赤紫・青緑は、重ねるほどに暗くなります（減法混色）。光の三原色「赤・緑・青」と、色の三原色「黄・赤紫・青緑」、相補性を成しています。



<sup>a</sup> 基本的な色指定である RGB 値 ([https://www.webcolordesign.net/color\\_basic/hsb\\_basic/rgb\\_color.html](https://www.webcolordesign.net/color_basic/hsb_basic/rgb_color.html)) より、引用・改変。

赤 (Red)・緑 (Green)・青 (Blue) のそれぞれを、「点灯している」「消灯している」の二段階で組み合わせると  $2^3 = 8$  色を表現可能できます。中間色を表現するために 各色に 256 段階の輝度を持たせることとすると  $256^3 = 16,777,216$  色を表現可能です。

色の指定は、#(シャープ) に続けて、この 256 段階を 16 進数に変換した表記が良く用いられます。赤は `#ff0000` 緑は `#00ff00` 青は `#0000ff` です。赤と緑を合わせると 黄 `#ffff00` になります。ですので `color: red;` に代えて `color: #ff0000;` と書くことができます。

良く使う色 140 色には「CSS カラーネーム」として、名前が付けられています。 \*2

black #000000	aliceblue #f0f8ff	darkcyan #008b8b	lightyellow #ffffe0	coral #ff7f50
dimgray #696969	lavender #e6e6fa	teal #008080	lightgoldenrodyellow #fafad2	tomato #ff6347
gray #808080	lightsteelblue #b0c4de	darkslategray #2f4f4f	lemonchiffon #fffacd	orangered #ff4500
darkgray #a9a9a9	lightslategray #778899	darkgreen #006400	wheat #f5deb3	red #ff0000

▲ 図 4.2: CSS カラーネーム (一部)

## ♣ 和名での色指定

英語では 140 の色名が付けられていますが、和色大辞典<sup>\*3</sup> には 465 色、和の色日本の伝統色<sup>\*4</sup> には 759 色もの日本の伝統色が紹介されています。

\*2 出典: 原色大辞典 (<https://www.colordic.org/>)

\*3 <https://www.colordic.org/w>

\*4 <https://irononamae.web.fc2.com/wa/>



▲図 4.3: 和の色 日本の伝統色 (一部)

二千六百余年にも渡る歴史の中で受け継がれてきた情緒豊かな伝統色をぜひ使いたいものです。そこで、次のように **CSS カスタムプロパティ** を定義します。これにより `color: var(--amairo);` などと、和名での色指定ができるようになります。

```
:root {
    --kyohiiro: #ff251e; /* 京緋色(きょうひいろ) */
    --shinonomeiro: #f19072; /* 東雲色(しののめいろ) */
    --nanohanairo: #ffec47; /* 菜の花色(なのはないろ) */
    --sanaeiro: #67a70c; /* 早苗色(さなえいろ) */
    --amairo: #2ca9e1; /* 天色(あまいろ) */
    --utsushiiro: #3d6eda; /* 移色(うつしいろ) */
    --botaniro: #e7609e; /* 牡丹色(ぼたんいろ) */
    --ayameiro: #674196; /* 菖蒲色(あやめいろ) */
    --sakurairo: #fef4f4; /* 桜色(さくらいろ) */
    --momijiyo: #a61017; /* 紅葉色(もみじいろ) */
    --nibiyo: #9ea1a3; /* 鈍色(にびいろ) */
    --kurohairo: #0d0d0d; /* 黒羽色(くろはいろ) */

    --harukazeiro: transparent; /* 春風色(はるかぜいろ) */
}
```

▲図 4.4: CSS カスタムプロパティによる和色指定

ウェブサイト作成で使いやすいよう、**和の色 日本の伝統色<sup>5</sup>**として、838 色を纏めていますので、ご利用下さい。

<sup>5</sup> [https://github.com/Atelier-Mirai/wa\\_no\\_iro](https://github.com/Atelier-Mirai/wa_no_iro)

## 4.4 CSS グリッドレイアウト

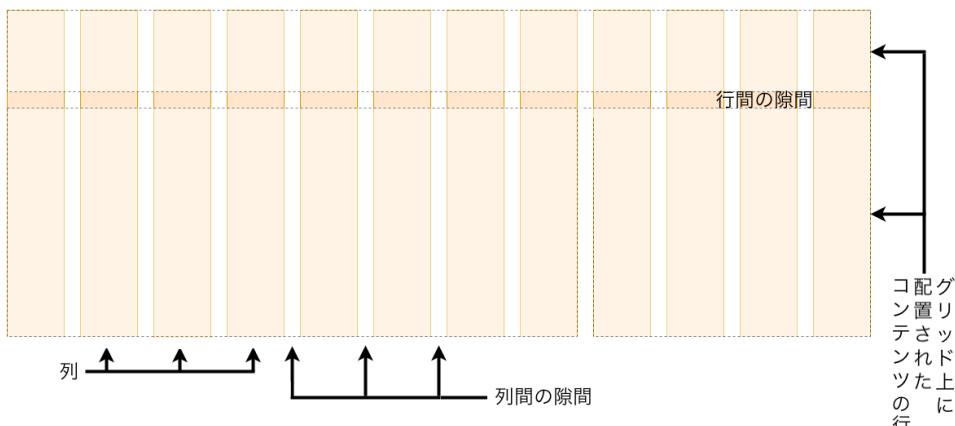
通常、ウェブページは、上から下、左から右に、配置されます。従来、ページ上の要素を自由自在に配置するには大変な苦労が伴いました。CSS グリッドレイアウトを用いると、縦と横の補助線（グリッド・格子）を用いて、自由自在に要素を配置することができます。

公式サイト MDN に、[グリッド<sup>\\*6</sup>](#) について書かれた記事がございますので、引用しつつ進めていきましょう。<sup>\*7</sup>

### ♣ グリッドとは

CSS グリッドレイアウト (Grid Layout) は、ウェブ用の二次元レイアウトシステムです。これにより、コンテンツ (内容物・出し物) を行と列にレイアウト (配置) することができ、複雑なレイアウトを簡単に構築できるようにする多くの機能があります。グリッドは、列と行を定義する水平線と垂直線の集合が交差したものです。要素をグリッドの行と列に並べて配置することができます。

グリッドには通常、列 (column)、行 (row)、そしてそれぞれの行と列の間の間隔 (gap) があります。



### ♣ グリッドを定義する

練習として、以下のように grid.html と grid.css を用意しましょう。

#### ▼ grid.html

```
1 <!DOCTYPE html>
```

<sup>\*6</sup> [https://developer.mozilla.org/ja/docs/Learn/CSS/CSS\\_layout/Grids](https://developer.mozilla.org/ja/docs/Learn/CSS/CSS_layout/Grids)

<sup>\*7</sup> グリッドレイアウトの基本概念 ([https://developer.mozilla.org/ja/docs/Web/CSS/CSS\\_Grid\\_Layout/Basic\\_Concepts\\_of\\_Grid\\_Layout](https://developer.mozilla.org/ja/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout)) にも分かりやすく説明されており、お勧めです。

```

2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>CSSグリッドレイアウト練習</title>
6     <link rel="stylesheet" href="grid.css">
7   </head>
8
9   <body>
10    <h1>CSSグリッドレイアウト練習</h1>
11
12   <div class="container">
13     <div>One</div>
14     <div>Two</div>
15     <div>Three</div>
16     <div>Four</div>
17     <div>Five</div>
18     <div>Six</div>
19     <div>Seven</div>
20   </div>
21
22   </body>
23 </html>

```

## ▼grid.css

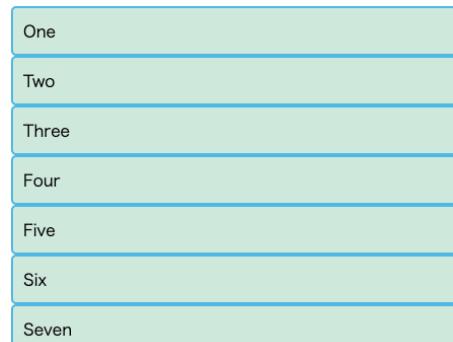
```

1 .container > div {
2   border-radius: 5px;
3   padding: 10px;
4   background: #cfe8dc;
5   border: 2px solid #4fb9e3;
6 }

```

.container > div は、container クラスの直下の div タグを選択するためのセレクタです。  
<div>One</div>、<div>Two</div>などが薄い緑地に水色の枠で囲まれるようになります。

## CSSグリッドレイアウト練習



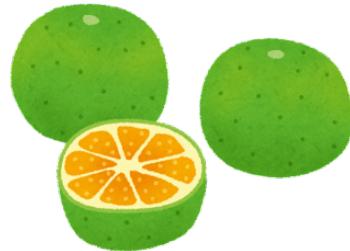
グリッドを定義するためには、display プロパティに grid の値を使います。これによりグリッドレイアウトが有効になり、コンテナの直接の子の全てがグリッド項目（アイテム）になります。ここでは、コンテナが <div class="container"> で、グリッド項目が<div>One</div>、<div>Two</div>などです。

「みかん箱」に「みかん」を入れる、そのような想像をすると分かりやすいかもしれません。グ

リッドコンテナ=「みかん箱」、グリッド項目=「みかん」です。\*8



コンテナ = 箱



アイテム = みかん

それでは、CSS に次を追加してみましょう。

#### ▼grid.css

```
8 .container {  
9   display: grid;  
10 }
```

ブラウザを再読み込みして確認してみても、特に変化が見られません。`display: grid` を宣言すると 1 列のグリッドになるので、項目は上下に表示され続けます。

よりグリッドらしく見せるには、グリッドにいくつかの列を追加する必要があります。ここでは `grid-template-columns` プロパティにより、200 ピクセルの列を 3 つ追加しましょう (`grid-template-rows` プロパティを使うと、行数を指定できます)。

以下のように CSS を更新しましょう。

#### ▼grid.css

```
8 .container {  
9   display: grid;  
10  grid-template-columns: 100px 100px 100px;  
11 }
```

CSS 規則に 2 番目の宣言を追加してからページを再読み込みすると、作成したグリッドの各セルに項目が 1 つずつ再配置されていることがわかります。

## CSSグリッドレイアウト練習

One	Two	Three
Four	Five	Six
Seven		

### ♣ fr 単位での柔軟なグリッド

`fr` は、`fraction(分数)` に由来した単位で、CSS グリッドレイアウトのために誕生した新しい

\*8 出典: 季節のイベント・動物・子供などのかわいいイラストが沢山見つかるフリー素材サイト いらすとや (<https://www.irasutoya.com/>)

単位です。

`px`などの長さとパーセントを使用してグリッドを作成するだけでなく、この `fr` 単位を使用して柔軟にグリッドの行と列のサイズを変更できます。この単位は、グリッドコンテナ内の使用可能スペースの割合を表します。

列のリストを次の定義に変更し、`1fr` の列を 3 つ作成します。

#### ▼grid.css

```
8 .container {
9   display: grid;
10  grid-template-columns: 1fr 1fr 1fr;
11 }
```

グリッドコンテナ内を三等分して、それぞれの `div` 要素が配置されていることが確認できます。

### CSSグリッドレイアウト練習

One	Two	Three
Four	Five	Six
Seven		

より柔軟に列幅を変更することもできます。`fr` 単位はスペースを比例して配分するので、各列には異なる正の値を指定できます。

例えば `grid-template-columns: 2fr 1fr 1fr;` と定義することで、右のようにできます。

### CSSグリッドレイアウト練習

One	Two	Three
Four	Five	Six
Seven		

## ♣ 行間や列間の間隔

間隔を作成するには、列間の間隔には `column-gap` プロパティ、行間の間隔には `row-gap` プロパティ、両方を同時に設定するには `gap` プロパティを使用します。

#### ▼grid.css

```
8 .container {
9   display: grid;
10  grid-template-columns: 2fr 1fr 1fr;
11  gap: 20px;
12 }
```

右のように、行間、列間、それぞれ `20px` の間隔を空けることができました。

### CSSグリッドレイアウト練習

One	Two	Three
Four	Five	Six
Seven		

## ♣ 行指定や列指定の繰り返し

反復記法を使用して、行指定や列指定を繰り返すことができます。

### ▼ grid.css

```
8 .container {
9   display: grid;
10  grid-template-columns: repeat(4, 1fr);
11  gap: 20px;
12 }
```

グリッドコンテナ内を四等分して、それぞれの div 要素が配置されていることが確認できます。

`grid-template-columns: 1fr 1fr 1fr 1fr;` と書いたことと同じですが、繰り返す回数が多いときに便利です。

### ♣ グリッドの線の番号を使った要素の配置

グリッドの作成から、グリッド上に要素を配置することに移ります。グリッドは 1 から始まる線番号を持っていてます。列の 1 線目がグリッドの左側にあり、行の 1 線目が一番上にあります。

開始線と終了線を指定することで、これらの線に従って要素を配置できます。

CSS からそれぞれの要素を扱いやすくするために、少しだけ `grid.html` を次のように更新し、`<div class="one">One</div>` と、クラス名「one」を付けましょう。

### ▼ grid.html

```
12 <div class="container">
13   <div class="one" >One</div>
14   <div class="two" >Two</div>
15   <div class="three">Three</div>
16   <div class="four" >Four</div>
17   <div class="five" >Five</div>
18   <div class="six" >Six</div>
19   <div class="seven">Seven</div>
20 </div>
```

CSS は次のように更新します。

### ▼ grid.css

```
8 .container {
9   display: grid;
10  grid-template-columns: 1fr 1fr 1fr 1fr;
11  grid-template-rows:    1fr 1fr 1fr 1fr;
12  gap: 20px;
13 }
```

表示結果は先ほどと変わりませんが、四行四列のグリッド（格子）を作成することができました。

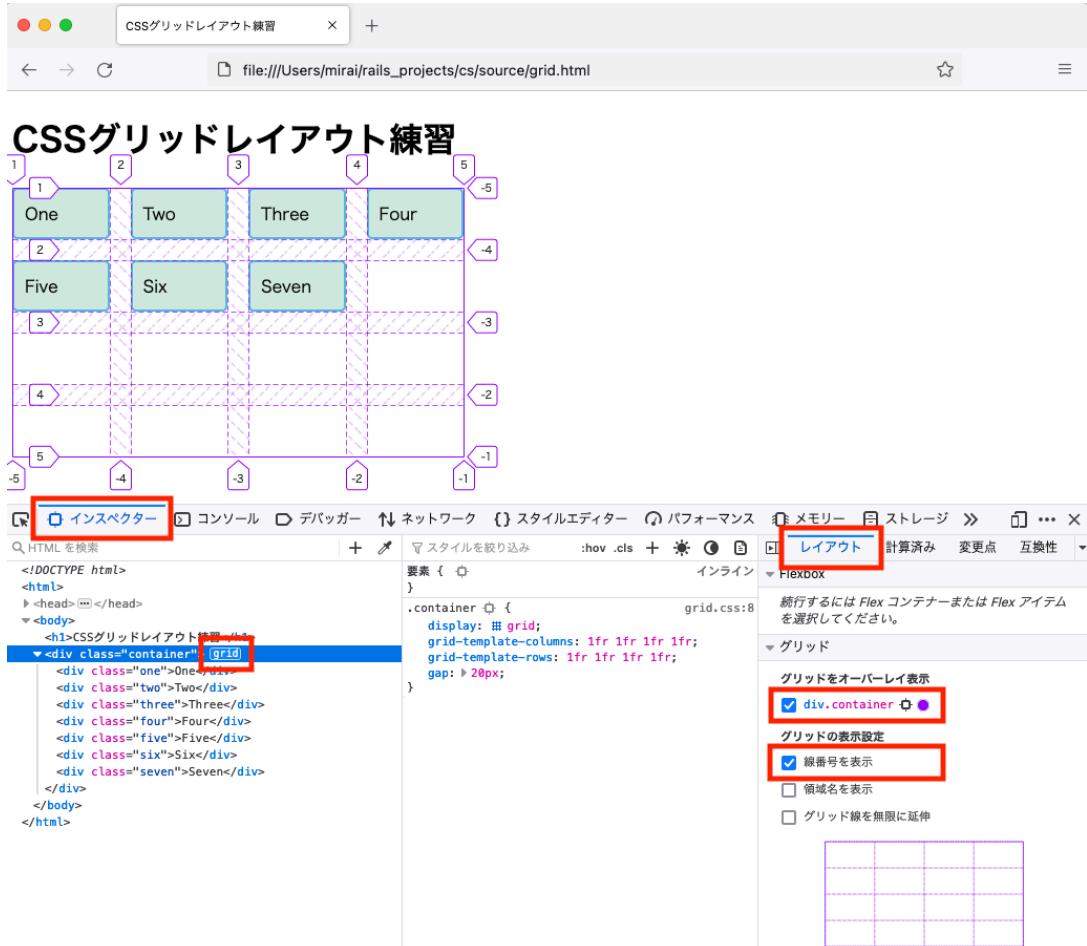
## CSSグリッドレイアウト練習



「開発者ツール」を使うと、どのようにグリッドが構成されたか、確認できます。画面上で、右



クリック \*9 して表示される「ショートカットメニュー」から「調査」をクリックします。



開発者ツールが表示されましたので、グリッドの線番号を表示させましょう。

「インスペクタ」をクリックします。- 1. 下に HTML コードが表示されています。`<div class="container">` の右側に、「grid」と書かれた小さなボタンがありますので、これをクリックします。または、右側の「レイアウト」欄内の「グリッド」の項目にある「グリッドをオーバーレイ表示」にチェックを入れます。- 2. 次に、右側の「レイアウト」欄内の「グリッド」の項目にある「グリッドの表示設定」で「線番号を表示」にチェックを入れます。

グリッドの線番号が表示されました。上部を見ると、左から、1, 2, 3, 4, 5 と番号が振られています（四列作りましたので、線は 1～5 までの五本できます）。

それでは、次のように CSS を追加して、「One」から「Seven」までの要素を配置してみましょう。

\*9 システム設定 - トランクパッド - ポイントとクリック - 副ボタンのクリック から 「右下隅をクリック」 に設定すると、右クリックできるようになります。

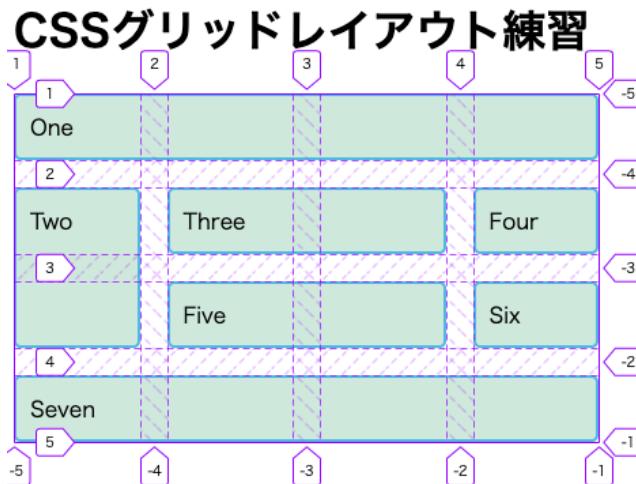
## ▼grid.css

```

15 .one { grid-column: 1 / 5; grid-row: 1; }
16 .two { grid-column: 1; grid-row: 2 / 4; }
17 .three { grid-column: 2 / 4; grid-row: 2; }
18 .four { grid-column: 4; grid-row: 2; }
19 .five { grid-column: 2 / 4; grid-row: 3; }
20 .six { grid-column: 4; grid-row: 3; }
21 .seven { grid-column: 1 / 5; grid-row: 4; }

```

結果は次のようにになります。



.one { grid-column: 1 / 5; grid-row: 1; } は、列は一本目の線から五本目の線まで、行は一行目に配置されています。

.two { grid-column: 1 ; grid-row: 2 / 4; } は、列は一本目の線に、行は二本目から四本目に配置されています。

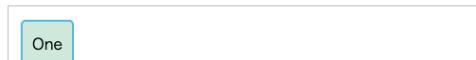
このように、グリッドの線を使って、好きなように要素を配置することができます。

### ♣ 要素の左揃え、中央揃え、右揃え

`justify-self` プロパティを使うと、「**水平**」方向での要素の配置を制御できます。

CSS を追加して実験してみましょう。

**左揃え** `.one { justify-self: start; }`



**中央揃え** `.one { justify-self: center; }`



**右揃え** `.one { justify-self: end; }`

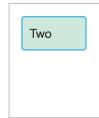


## ♣ 要素の上揃え、中央揃え、下揃え

`align-self` プロパティを使うと、「垂直」方向での要素の配置を制御できます。

CSS を追加して実験してみましょう。

**上揃え** `.two { align-self: start; }`



**中央揃え** `.two { align-self: center; }`



**下揃え** `.two { align-self: end; }`



## ♣ 線番号に名前を付ける

線番号に名前を付け、より使いやすくすることができます。

### ▼ grid.css

```
.container {
  grid-template-rows: 1fr 1fr 1fr 1fr;
}
```

と記述して、四行作成しましたが、以下のように書くことで、行のそれぞれの線に名前を付けることができます。

### ▼ grid.css

```
.container {
  grid-template-rows:
    [head] 1fr /* 一本目の線に head と命名 */
    [main] 1fr /* 二本目の線に main と命名 */
    [article] 1fr /* 三本目の線に article と命名 */
    [foot] 1fr; /* 四本目の線に foot と命名 */
}
```

そして、この線番号に付けた名前を使って、

### ▼ grid.css

```
.one  { grid-column: 1 / 5; grid-row: head; }
.two  { grid-column: 1;      grid-row: main / span 2; } /* 二行使い、配置する */
.three { grid-column: 2 / 4; grid-row: main; }
.four  { grid-column: 4;      grid-row: main; }
.five  { grid-column: 2 / 4; grid-row: article; }
.six   { grid-column: 4;      grid-row: article; }
.seven { grid-column: 1 / 5; grid-row: foot ; }
```

と書くことができます。

結果は同じですが、それぞれの要素をどの行に配置するのか、より分かりやすくなりますので、ウェブサイト全体など、良く使う線には名前を付けておくと便利です。

## 4.5 iPhone での表示結果を確認する

開発者ツール<sup>\*10</sup>を使うと、iPhone での表示結果を確認することができます。

みんな知っているじゃんけん  
グー・チョキ・パー どれかを押してね

iPhoneとiPadが  
重なっている印

```

<head>
  <title>じゃんけんゲーム</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
  <link href="janken.css" rel="stylesheet">
</head>
<body>
  <h1>じゃんけんゲーム</h1>
  <img alt="Illustration of a hand making a fist" data-bbox="258 354 738 478" />
  <p>みんな知っているじゃんけん  
グー・チョキ・パー どれかを押してね</p>
  <div class="control_area">
    <div><img alt="Rock icon" data-bbox="238 534 403 648" /></div>
    <div><img alt="Paper icon" data-bbox="418 534 583 648" /></div>
    <div><img alt="Scissors icon" data-bbox="598 534 763 648" /></div>
  </div>
</body>

```

janken.css:26

```

body {
  margin: 0;
  box-sizing: border-box;
}

html {
  font-size: 10px;
}

```

janken.css:7

```

:root {
  --kyohiro: #ff251e;
  --shinonomeiro: #f19072;
  --nanohanairo: #fec47;
}

```

絵の右下に iPhone と iPad が重なったアイコンがあります。「レスポンシブデザインモード」と呼ばれるアイコンで、クリックすると iPhone や iPad での見え方を確認できます。

<sup>\*10</sup> キーボードショートカットも用意されています。開発者ツールの表示は command + option + M、レスポンシブデザインモードへの切り替えは command + option + M です。



レスポンシブデザインモードに変更すると、画面上部には、様々な端末に切り替えられるよう、良く用いられている端末名が用意されています。クリックしてみましょう。

左側のツールバーで「レスポンシブ」を選択すると、右側の「デバイスリスト」が表示されます。各端末名の横には、選択状態を示す赤い枠やチェックマークがあります。

デバイス名	選択状態
iPad iPadOS 14.7.1	<input checked="" type="checkbox"/>
iPad Pro (12.9-inch) iPadOS 14.7.1	<input type="checkbox"/>
iPhone 12/13 Pro Max iOS 14.6	<input type="checkbox"/>
✓ iPhone SE 2nd gen iOS 14.6	<input checked="" type="checkbox"/>

レスポンシブデザインモードに変更すると、画面上部には、様々な端末に切り替えられるよう、良く用いられている端末名が用意されています。クリックしてみましょう。

- iPad iPadOS 14.7.1
- iPad Pro (12.9-inch) iPadOS 14.7.1
- iPhone 12/13 Pro Max iOS 14.6
- iPhone SE 2nd gen iOS 14.6

リストを編集...

iPad や iPhone など様々な端末を選ぶことができます。iPhone SE 2nd gen iOS 14.6 \*<sup>11</sup> を選びましたので、端末の画面サイズ、横 375px 縦 667 px が表示されています。

その右側にある iPhone のマークをクリックすると、横向きにした際の表示も確認できます。

また「リストを編集」をクリックすると良く使う端末だけを表示させたり、新しい端末が発売されたときに、画面サイズを登録することができます。

## 4.6 完成形の CSS

CSS による装飾の基本と、グリッドを活用した配置方法を学びました。

完成形の CSS は次のようにになります。

### ▼ janken.css

```

1  /*
2   * =====
3   * じゃんけんゲームの為のスタイルシート
4   * =====
5  */
6  /* 和の色 (CSSカスタムプロパティ)
7  -----*/
8  :root {
9    --kyohiiro:          #ff251e; /* 京緋色(きょうひいろ) */
10   --shinonomeiro:     #f19072; /* 東雲色(しののめいろ) */
11   --nanohanairo:      #ffec47; /* 菜の花色(なのはないいろ) */
12   --sanaeiro:          #67a70c; /* 早苗色(さなえいろ) */
13   --amairo:            #2ca9e1; /* 天色(あまいいろ) */
14   --utsushiiro:        #3d6eda; /* 移色(うつしいいろ) */
15   --botaniiro:         #e7609e; /* 牡丹色(ぼたんいろ) */
16   --ayameiro:          #674196; /* 菖蒲色(あやめいろ) */
17   --sakurairo:         #fef4f4; /* 桜色(さくらいろ) */
18   --momijiiro:         #a61017; /* 紅葉色(もみじいろ) */
19   --nibiiro:            #9ea1a3; /* 鈍色(にびいろ) */
20   --kurohairo:          #0d0d0d; /* 黒羽色(くろはいろ) */
21   --harukazeiro:       transparent; /* 春風色(はるかぜいろ) */
22 }
23
24 /* 基本設定(リセットCSS)
25 -----*/
26 * {                      /* 全ての要素(*)を対象に */
27   margin: 0;              /* 余白を0にする */
28   box-sizing: border-box; /* 要素の幅を制御しやすくする */
29 }
30
31 img {                  /* 全ての画像要素(img)を対象に */
32   width: 100%;            /* 幅を100% にする */

```

\*<sup>11</sup> iPhone SE 第二世代のことです。現在は第三世代が提供されていますが、画面サイズは同じとなっています。

```
33 height: auto;           /* 高さは 幅に応じて 自動調整する */
34 }
35
36 /* CSSグリッドレイアウトで ページ全体の配置設定する
37 -----*/
38 body {                  /* body は全ての要素の親 */
39   display: grid;        /* グリッド(格子)を使うモードにする */
40   grid-template-columns: /* column(列) の設定を行う */
41     20px 1fr 20px;      /* 左右に20px 残りは中央 */
42   grid-template-rows:   /* row(行)の設定を行う */
43     [head]    80px      /* 一行目の高さは8px headと命名 */
44     [main]    auto;     /* 二行目の高さは自動 titleと命名 */
45 }
46
47 /* 部品の配置
48 -----*/
49 body > * {             /* body直下(>)の全要素(*)を対象に */
50   grid-column: 2 / 3;   /* 列配置は 左から2番目の線から3番目の線まで */
51 }
52
53 /* ヘッダー
54 -----*/
55 header {                /* header 要素を対象に */
56   grid-row: head;       /* 行の配置は先に命名したhead線の下に */
57   justify-self: center; /* 左右中央揃えで配置する */
58   align-self: center;   /* 上下中央揃えで配置する */
59 }
60
61 header h1 {             /* header 内の h1 (大見出しの指定) */
62   font-size: 40px;       /* 書体の大きさは40px */
63   color: var(--kurohairo); /* 文字色を黒羽色にする */
64   text-shadow: 3px 4px 5px var(--nibiiro); /* 鈍色の影を付ける */
65   text-align: center;    /* 文字は中央揃えにする */
66 }
67
68 /* メイン(サイト主要機能部)
69 -----*/
70 main {                  /* main 要素を対象に */
71   grid-row: main;       /* 行の配置は先に命名したmain線の下に */
72 }
73
74 main figure img {       /* メインの画像(=コンピュータの手)を対象 */
75   max-height: 40vh;
76   aspect-ratio: 1 / 1;
77 }
78
79 main p {                /* じゃんけんの説明文 */
80   text-align: center;
81   margin-bottom: 20px;
82 }
83
84 main .control_area {    /* control_areaクラスを対象に */
85   display: grid;
86   /* 内部要素をグリッドで配置 */
```

```

86 grid-template-columns: repeat(6, 1fr); /* column(列) を六列 用意する */
87 grid-template-rows: 2fr 1fr; /* row(行) は 二行 用意する */
88 gap: 20px; /* 間隔を 20px 開ける */
89 justify-items: center; /* グリッド内の要素を水平方向に中央揃えする */
90 align-items: center; /* グリッド内の要素を垂直方向に中央揃えする */
91 }
92
93 /* button要素 を対象に */
94 main .control_area button {
95   aspect-ratio: 1 / 1; /* 横縦比は 1 対 1 に */
96   width: 100%; /* 幅100%で表示 */
97   background-size: contain; /* 背景画像が全て含むようにする */
98   background-repeat: no-repeat; /* 背景画像を繰り返さないようにする */
99   margin: 10px 0; /* 上下に10px 左右に0px の余白 */
100  background-color: var(--amairo); /* 背景色は 天色 */
101  color: white; /* 文字の色は白*/
102  font-size: 24px; /* 書体の大きさは24px */
103  border-radius: 15px; /* 角は 半径(radius) 15pxで丸くする */
104  cursor: pointer; /* カーソルの形状を手のマークに */
105 }
106
107 /* ボタンの配置指定 */
108 /* 列は、左から一本目の線から三本目の線の間で、行は一行目に配置する */
109 main .control_area #guu { grid-column: 1 / 3; grid-row: 1; }
110 main .control_area #choki { grid-column: 3 / 5; grid-row: 1; }
111 main .control_area #paa { grid-column: 5 / 7; grid-row: 1; }
112
113 main .control_area .score { grid-column: 1 / 4; grid-row: 2;
114   font-size: 24px; } /* 書体の大きさを指定する */
115 main .control_area #play { grid-column: 4 / 7; grid-row: 2;
116   aspect-ratio: 3 / 1; } /* 横縦比を横 3 対 縦 1 にする */
117
118 /* グー、チョキ、パー 各ボタンの背景色と背景画像を指定 */
119 main .control_area #guu { background-color: var(--sanaeiro);
120   background-image: url('player_guu.webp'); }
121 main .control_area #choki { background-color: var(--nanohanairo);
122   background-image: url('player_choki.webp'); }
123 main .control_area #paa { background-color: var(--kyohiro);
124   background-image: url('player_paa.webp'); }

```

紙幅の制約から、説明は割愛致しますが、それぞれの行にコメントも付与してございますので、読み取っていただければと思います。



# 第 5 章

## じゃんけんゲームの完成

長かったじゃんけんゲームの旅もあと一息です。前章では、利用者に心地よく使ってもらえるよう、CSS を導入し綺麗な意匠を整えました。この章では、JavaScript のコードを追加し、いよいよじゃんけんゲームを完成させます。ウェブサイトへの公開方法も記載いたしましたので、是非遊んでみてください。

利用者に心地よく使ってもらえるよう、HTML を追記し、CSS を導入したことでの綺麗な意匠を実現しました。美しい見栄えになったことで、創作意欲も湧きます。

この章では次の三点を実装し、遂にじゃんけんゲームが完成を迎えます。

1. プレイヤーがどの手を出すのか、今までには、0, 1, 2 と、数字で入力していました。より人に分かりやすい、グーチョキパーの 3 つのボタンを用意し、それを押すことで、プレイヤーが手を選べるようにします。
2. 今までには、コンピュータがどの手を出したのか、表示する機能がありませんでした。グーチョキパーどれを出しているのか明示し、アニメーション機能も実装します。
3. ○勝○敗と、今までの勝敗を表示できるようにします。

### 5.1 プレイヤーの手の取得と コンピュータの手の表示

#### ♣ プレイヤーの手の取得

これまでには、数値入力枠の中にプレイヤーが 0, 1, 2 の数字を入力することによって、グーチョキパーを得ていました。そのためのコードが次のコードでした。

##### ▼ janken.js

```
// player の手を取得
const inputBox = document.getElementById("player_hand_type")
let player = Number(inputBox.value)
```

今回より利用者に分かりやすいよう「UI/UX」の改善を図ったので、グーチョキパー、どのボタンが押されたのか取得する必要があります。そのため、次のように書き換えましょう。

#### ▼ janken.js

```
const guu_button = document.getElementById("guu");
const choki_button = document.getElementById("choki");
const paa_button = document.getElementById("paa");
guu_button.addEventListener("click", jankenHandler);
choki_button.addEventListener("click", jankenHandler);
paa_button.addEventListener("click", jankenHandler);
```

`document.getElementById("guu")` で、グーボタン要素を取得します。そして、プログラム内で扱いやすいよう、`guu_button` という変数に代入します。この後、`guu_button` という名前で呼ぶことで、利用者が押したボタンの値を取得します。

`guu_button.addEventListener("click", jankenHandler);` では、`click` イベントを聴取する関数として `jankenHandler` を設定しています。これで、三つのボタンそれぞれが押されたら、ともに `jankenHandler` 関数が呼ばれるようになりました。

それでは、`player` の手 (0, 1, 2) は、`jankenHandler` 関数内ではどのようにして取得すれば良いのでしょうか？

```
function jankenHandler(event) {
  // (略)
}
```

`jankenHandler` に、引数 `event` が渡されていることに着目してください。`event.target` で、イベントの呼び出し元の要素を取得することができます。つまり、グー、チョキ、パー、どのボタンが押されたのかを知ることができます。

`event.target.value` と書くことで、`<button id="guu" value="0"></button>` と書かれていた `value` 属性の値 "0" が取得できます。

`Number` は、文字列としての "0" を、整数値 0 に変換する関数です。

これで、グーボタンを押した時は 0、チョキボタンを押した時は 1、パーボタンを押した時は 2 が、`player` 変数に格納されます。

```
const player = Number(event.target.value)
```

## ♣ コンピュータの手の表示

今まででは、コンピュータの手は表示されていませんでしたので、HTML で次のように書くことで、グーの絵が表示されるようにしました。

```

```

出来ればこれも、無作為に変わるようにしたいものです。JavaScript では、書かれた HTML をプログラム上から動的に書き換えることができます。

```
// コンピュータの手を無作為に決定する
computer = rand(0, 2)
// コンピュータの手の画像を動的に変更する
document.getElementById("computer_hand_type").src =
  ["guu.png", "choki.png", "paa.png"][computer]
```

一行目は以前に触れた乱数でコンピュータの手を決定しています。二行目を解説します。

`document.getElementById("computer_hand_type")` で、HTML ファイルに書いた イメージ要素 を取得します。`img` 変数には、`` が入っています。`src="guu.png"` と書いたので、グーの画像が表示されました。

`const img = document.getElementById("computer_hand_type")` として、取得した要素を `img` という変数に格納します。そして取得した `img` 要素の `src` 属性を `choki.png` にすればチョキの画像を、 `paa.png` にすればパーの画像を表示させることができます。

JavaScript では、取得した要素の属性をいろいろ操作することができます。画像を変更するには、次のように書きます。

```
img.src = "choki.png"
```

## 配列

配列はとてもよく使われるデータ構造です。いくつかの変数を一緒にものとして扱いたい時に、重宝します。

グーの画像、チョキの画像、パーの画像 それぞれをまとめて扱いたいので、配列を使うとともに便利です。

じゃんけんの手の画像の集まり（配列）として、 `images` という変数を宣言し、初期値として `"guu.png", "choki.png", "paa.png"` 三つの画像名があるようにします。

### ▼ じゃんけん画像配列の宣言

```
const images = ["guu.png", "choki.png", "paa.png"]
```

`const images = [];` と書くと、中身が空っぽの配列を作成することができます。`const images = ["guu.png", "choki.png", "paa.png"]` と書くと、 配列 `images` の中に、 `"guu.png", "choki.png", "paa.png"` の三つの要素があるようになります。

## 配列内の要素の指定法

配列内の各要素を指定するには、配列名の後に【何番目かを指示する数字】と書きます。この「何番目かを指示する数字」のことを、**添字(そえじ)** と呼びます。

配列の要素は、 `0, 1, 2` と `0` から数え始めますので、 `images[0]` と書くと `"guu.png"` を指定でき、 `images[1]` と書くと `"choki.png"` を指定できます。逆に、 `"paa.png"` が欲しい時には、 `images[2]` と書くと良いです。

配列はとってもよく使う基礎的な**データ構造**で、少し大きなプログラムでは不可欠です。是非、習得なさってください。

.....  
配列と並ぶ重要な**データ構造**に、**連想配列**（ハッシュや辞書とも呼ばれます）があり、これもとても重要なのですが、紙幅の関係上、割愛いたします。さまざまな学習資源がありますので、ぜひ学んでみてください。  
.....

無作為にグーチョキパーが表示されるようにしたいので、「乱数」を使うと便利です。JavaScriptに標準で備わっている乱数からは、0から1の浮動小数点数が得られます。既にじゃんけん用の乱数を自作したので、それを使いましょう。

```
// 亂数を利用して、コンピュータの手を無作為に決定する  
computer = rand(0, 2)
```

ですので、乱数で選ばれた画像ファイル名は、次のようにになります。

```
const image_filename = images[computer]
```

よって、以下のように書くことで、画像ファイルを都度都度変更することができます。

```
img.src = image_filename
```

つまり、一行ずつ分けて書くと次のようなコードになります。

#### ▼一行ずつ分けて書いたコード

```
computer      = rand(0, 2)  
images        = ["guu.png", "choki.png", "paa.png"]  
const image_filename = images[computer]  
img           = document.getElementById('computer_hand_type')  
img.src       = image_filename
```

これを、それぞれの変数に代入するのではなく、まとめて書くと次のようにになります。

#### ▼まとめて書いたコード

```
computer = rand(0, 2)  
document.getElementById('computer_hand_type').src =  
    ["guu.png", "choki.png", "paa.png"][computer]
```

ご自身の分かりやすいと感じる書き方で、実践してみてください。

## 5.2 アニメーション機能と勝敗更新機能

### ♣ アニメーション機能

JavaScript から、コンピュータの手を切り替える方法は分かりました。せっかくですので、アニメーション機能を実装したいところです。「グー、チョキ、パー」と 1 秒間に 24 回絵が切り替わるとアニメーションの完成です。

**fps** 【frames per second】 フレーム毎秒 fps とは、動画のなめらかさを表す単位の一つで、画像や画面を 1 秒間に何回書き換えているかを表したもの。24fps の動画は 1 秒あたり 24 枚の静止画で構成され、約 0.041 秒（41 ミリ秒）ごとに画像を切り替えて再生される。<sup>\*1</sup>

一定周期ごとに、処理を繰り返したいときに使う関数として、JavaScript では、`setTimeout` という関数が用意されています。使い方は次の通りです。

```
setTimeout(タイマーが満了した後に実行したい関数,  
          指定した関数を実行する前に待つ時間をミリ秒単位で指定)
```

関数名は、`animation` や `changeComputerHand` も良いでしょう。そしてここでは、アニメーション機能がこのじゃんけんプログラムの主となる機能であることから、`main`<sup>\*2</sup> という関数名にします。すると、次のように書けます。

```
setTimeout(main, 41)
```

41 ミリ秒ごとに一回ですから、1000 ミリ秒ごとに、24 回、じゃんけんの絵が入れ替わることになります。そして、41 と書くと、意味が分かりにくいので、FPS という定数を定義しましょう。FPS は、Frame Per Second の略で、「一秒間あたり、何コマ（フレーム）を表示するか？」の意味です。

```
const FPS = 24 // 一秒間あたり、24コマ表示する
```

この FPS を使って、次のように書き直してみましょう。

```
setTimeout(main, 1000 / FPS)
```

意味も明確になりますし、毎秒 60 コマのフレームレートに変更したければ、一箇所、更新するだけですみますので、保守性も上がります。

いつもアニメーション表示中ではなく、「開始」ボタンを押した時に、アニメーションが始まり、プレイヤーが手を選んだら、アニメーションが停止するようにしましょう。

<sup>\*2</sup> JavaScript はプログラムは上から順に実行されますが、C 言語や Java では main 関数から始まります。

アニメーション実行中か、否かを表す変数として、`isPause` 変数<sup>\*3</sup> を用いることになると、次のように書けます。

```
// ゲー・チョキ・パーの切替アニメを制御するための変数
// trueなら、アニメーション停止
let isPause = true;

// コンピュータの手を無作為に変更し、
// ゲー・チョキ・パーの切替アニメを表示させる関数
function main(){
    if(!isPause){ // 停止中でなければ
        computer = rand(0, 2)
        document.getElementById("computer_hand_type").src =
            ["guu.png", "choki.png", "paa.png"][computer]
    }
    setTimeout(main, 1000 / FPS)
}
```

コードの解説を行っていきます。

```
if(!isPause){ // 停止中でなければ
```

`if` 文の中の条件式には、真偽値を直接書くこともできます。「！」は否定演算子です。`isPause` が `true`(真) だった時には、`!isPause` は `false`(偽) となりますので、`if(!isPause){ // 停止中でなければ }` として、アニメーション切り替えのための `setTimeout` 関数を呼び出しています。

注目して欲しいのは、`main` 関数の中に書かれている `setTimeout` 関数から、もう一度、自分自身の関数 `main` を呼び出していることです。自分自身を呼び出す関数のことを「**再帰関数**」といいます。プログラミングを行う際に、時々出現するテクニックです。

### ♣ アニメーションの開始と終了処理

```
// ゲー・チョキ・パーの切替アニメを制御するための変数
// trueなら、アニメーション停止
let isPause = true
```

```
// 切替アニメを停止し、もう一度、じゃんけんを行います
function pause(){
    isPause = true
}

// 切替アニメを再開し、もう一度、じゃんけんを行います
function resume(){
    isPause = false
}
```

---

<sup>\*3</sup> Animation is pause? が変数名の由来です。

`isPause` という変数に、`true`, または `false` をセットしているだけの関数ですが、`pause` 停止、`resume` 再開 と名前を付けることで、コードを読むだけで意図を汲み取ることができます、とても分かりやすくなります。名前はとっても重要です。

## ♣ 勝敗更新機能の実装

最後に、勝敗更新機能を実装しましょう。

勝負の結果に応じて、○勝○敗を更新していきたいので、`jankenHandler` 内に実装するのが良さそうです。

既に勝敗結果を得る処理は書いていますから、次のように書くと良いでしょう。

```
// 勝敗に応じ、メッセージ表示 & 勝敗更新
if (result === DRAW) {
  alert("引き分けです!")
} else if (result === LOSE) {
  alert("あなたの負けです!")
  // 敗数を一つ増やす
  updateScore(LOSE)
} else {
  alert("あなたの勝ちです!")
  // 勝数を一つ増やす
  updateScore(WIN)
}
```

`updateScore` という関数を作って、その引数として、`LOSE` 敗北か、`WIN` 勝利を渡しています。実際の処理は、`updateScore` 内で行っていますが、こうやって字面を読むだけでも処理の内容が分かり、コードの見通しがよくなります。

それでは、`updateScore` 関数ですが、どのように書けば良いでしょうか？ 勝ち数、負け数は、HTML 内で `<span id="win">0</span>` のように書いていました。

JavaScript で扱いやすいよう、ID 属性を付与したので、`document.getElementById("win")` と書けばこの `win` 要素を取得できます。早速 `win` 変数に格納しましょう。

`win.textContent` と書くことで、`<span id="win">0</span>` と書いていた「0」を取得することができます。この「0」は、**文字列としての「0」** です。一般にプログラミングでは、文字列としての "0" と、数値としての 0 は区別されます。

### ▼ 文字列 "0" と 数値 0 は区別される

```
"0" + "1" // => "01" と文字列の追加が行われます。
0 + 1 // => 1 と、数値演算が行われます。
```

ですので、`Number` 関数を用いて、文字列としての "0" を与えて、整数値としての 0 得ます。整数値としての 0 得ることができましたから、「+ 1」と足し算することで、勝ち数を一つ増やせます。

`win.innerText = 1` と JavaScript を書くと、`<span id="win">0</span>` と書かれていた元々の HTML を `<span id="win">1</span>` へと更新できます。

以上をまとめると updateScore 関数は 次のようになります。

```
// 勝敗更新処理
function updateScore(result) {
    // 要素を取得
    const win = document.getElementById("win")
    const lose = document.getElementById("lose")

    if (result === WIN) { // 勝ちの場合
        win.innerText = parseInt(win.textContent) + 1
    } else if (result === LOSE) { // 負けの場合
        lose.innerText = parseInt(lose.textContent) + 1
    }
}
```

## 5.3 じゃんけんプログラム完成

長い道のりを経て、遂に完成したじゃんけんプログラム。ソースコードは次の通りです。

### ▼ janken.js

```
1 // 定数宣言
2 const DRAW = 0 // 引き分け
3 const LOSE = 1 // 負け
4 const WIN = 2 // 勝ち
5
6 const GUU = 0 // グー
7 const CHOKI = 1 // チョキ
8 const PAA = 2 // パー
9
10 const FPS = 24 // 一秒間あたり、24コマ表示する
11
12 // グローバル変数宣言
13 let isPause = true // グー・チョキ・パーの切替アニメを制御するための変数
14 let computer // コンピュータの手（グー:0、チョキ:1、パー:2 のいずれか）
15
16 // メイン処理
17 function main(){
18     if(!isPause){ // 停止中でなければ
19         computer = rand(0, 2)
20         document.getElementById("computer_hand_type").src =
21             ["guu.webp", "choki.webp", "paa.webp"][computer]
22     }
23     setTimeout(main, 1000 / FPS)
24 }
25
26 // ボタン初期化関数
27 function initButton() {
28     const guu_button = document.getElementById("guu")
29     const choki_button = document.getElementById("choki")
```

```

30 const paa_button = document.getElementById("paa")
31 guu_button.addEventListener("click", jankenHandler)
32 choki_button.addEventListener("click", jankenHandler)
33 paa_button.addEventListener("click", jankenHandler)
34
35 // playボタンがクリックされた時には、resume関数を実行して、
36 // ジャンケンの切替アニメが再開(resume)されるようにする。
37 const play_button = document.getElementById("play")
38 play_button.addEventListener("click", resume)
39 }
40
41 // ジャンケンの勝敗を取り扱う関数
42 function jankenHandler(event) {
43     // 開始ボタンが押された際に、ボタン表示を、「もう一度」に更新する
44     const play_button = document.getElementById("play")
45     play_button.innerText = "もう一度"
46
47     // アニメーション停止処理実行
48     pause()
49
50     // プレイヤーの手の取得
51     const player = Number(event.target.value)
52     // 勝敗結果の取得
53     const result = judge(player, computer)
54     // 勝敗に応じ、メッセージ表示 & 勝敗更新
55     if (result === DRAW) {
56         alert("引き分けです!")
57     } else if (result === LOSE) {
58         alert("あなたの負けです!")
59         // 敗数を一つ増やす
60         updateScore(LOSE)
61     } else {
62         alert("あなたの勝ちです!")
63         // 勝数を一つ増やす
64         updateScore(WIN)
65     }
66 }
67
68 // ジャンケンの効果的な勝敗判定アルゴリズム
69 function judge(player, computer) {
70     return (player - computer + 3) % 3
71 }
72
73 // 勝敗更新処理
74 function updateScore(result) {
75     const win = document.getElementById("win")
76     const lose = document.getElementById("lose")
77
78     if (result === WIN) { // 勝ちの場合
79         win.innerText = Number(win.textContent) + 1
80     } else if (result === LOSE) {
81         lose.innerText = Number(lose.textContent) + 1
82     }

```

```
83 }
84
85 // 切替アニメ停止処理
86 function pause(){
87     isPause = true
88 }
89
90 // 切替アニメ再開処理
91 function resume(){
92     isPause = false
93 }
94
95 // 亂数を返す関数
96 // rand(0, 2)と呼び出せば、0, 1, 2 と グーチョキパー の乱数を返す
97 function rand(min, max){
98     return Math.floor(Math.random() * (max - min + 1)) + min
99 }
100
101 // 実際の処理の開始
102 initButton() // ボタンの初期化を行う。
103 pause()      // 切替アニメを停止状態にする。
104 main()       // 切替アニメを実行待ちにし、
               // 開始ボタンが押されると切替アニメが実行される。
```

# 第6章

## ウェブサイトの公開

従来、ウェブサイトを公開するためには、**サーバー**と呼ばれるコンピュータに適宜様々な設定を施し、運用してきました。安全に運用するためにはそれなりの伎倆が求められるものでしたが、こうした中、台頭してきたのが、各種の**ホスティングサービス**です。<sup>ネットリファイ</sup>簡単にそして安価に運用することができます。ここでは、Netlify をご紹介します。

### 6.1

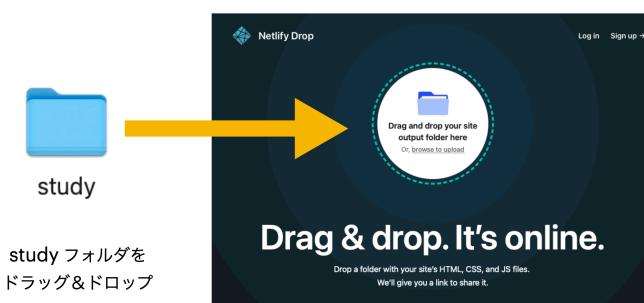
### ネットリファイ Netlify でウェブサイトを公開する

Netlify は、「ウェブサイトを構築する最速の方法です」と紹介されているように、HTML / CSS / JavaScript を簡単に配信できるサイトです。

#### ♣ 簡単公開

[https://app.netlify.com/drop<sup>\\*1</sup>](https://app.netlify.com/drop) をブラウザで開き、ウェブサイト一式を直接アップロードするだけで、全世界に公開できます。

じゃんけんゲームを、例えば「study フォルダ」に作成したのであれば、この「study フォルダ」をドラッグ&ドロップするだけで、世界中に 1 時間だけ公開できます。



\*1 <https://app.netlify.com/drop>

**6.2****ネットリファイ  
Netlify に会員登録する****♣ 会員登録の利点**

Netlify は、小規模利用<sup>\*2</sup>なら無料でサイトを公開できるサービスです。会員登録せざとも利用可能ですが、登録することにより、

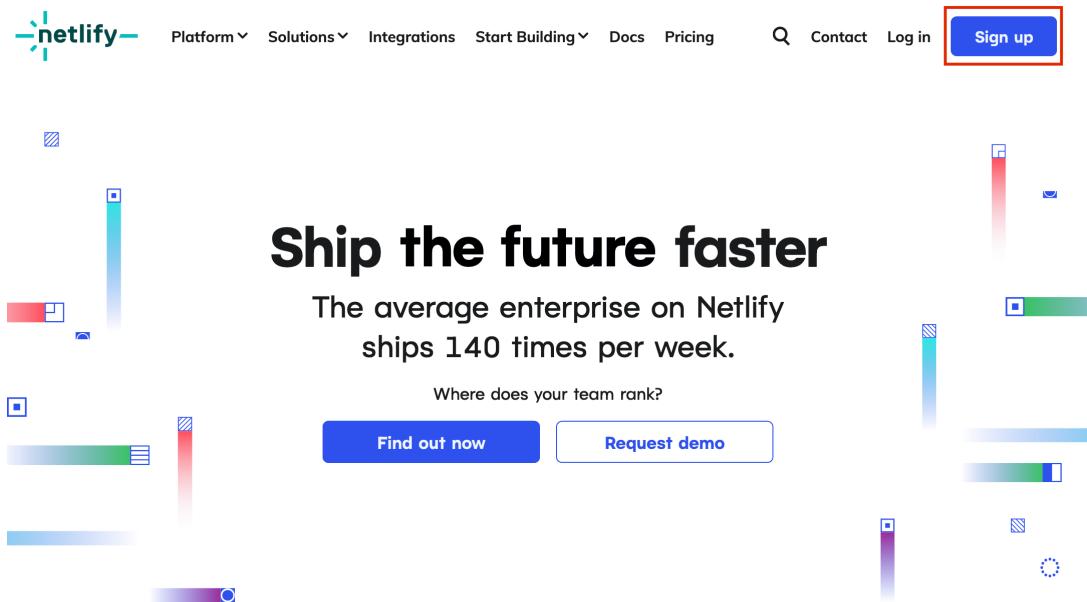
- サイトを永久的に公開できる。
- サイト名を好きな名前に変更できる。
- 問い合わせフォームの投稿を、メールで通知できる。

など、様々な機能が使えるようになりますので、登録がお薦めです。

また、バージョン管理システムとして Git<sup>ギット</sup> や リポジトリサービスである GitHub, GitLab,  
Bitbucket を使っている方なら、より便利に連携して使うことができます。

**♣ 会員登録の方法**

1. ブラウザを開いて Netlify<sup>\*3</sup> にアクセスします。



<sup>\*2</sup> Starter プラン(無料プラン)は、転送量: 100GB/月まで、容量: 100GB までです。作成した桜吹雪は約 700kB の容量があります。単純計算ですが、毎日 5000 人が見るようにになったら、有料プランへの移行を考えると良いでしょう。

<sup>\*3</sup> <https://www.netlify.com/>

2. 右上の青い「Sign up ボタン」を押すと、次の画面になります。

GitHub, GitLab, Bitbucket を使っていると、それぞれのアカウントを利用して、Netlifyへの登録ができますが、ここでは一番下の「Email」を押します。

### Sign up to deploy your project

 [Sign up with GitHub](#)

 [Sign up with GitLab](#)

 [Sign up with Bitbucket](#)

[Sign up with email](#)

Already have an account? [Log in](#)

3. メールアドレスとパスワードを入力し、「Sign up ボタン」を押します。



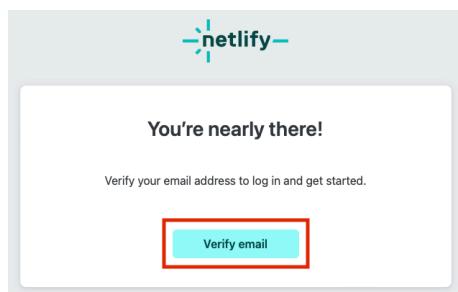


Sign up

4. 次の画面が表示されます。メールを送ったので確認してくれとのことです。



5. メールソフトを確認すると、Netlifyからのメールが届いていました。緑色の「Verify email」を押します。（不正利用やメールアドレスの入力ミス対策などの為に、良く用いられる仕組みです。）



6. 「Verify email」ボタンを押すと、利用者登録の続きをを行うことができます。

Nice to meet you! Let's get acquainted.(よろしくお願いします！ 知り合いになりましょう。)とご挨拶が有り、How ar your planning to use Netlify?(Netlify の利用予定は？)と尋ねられます。「Personal」「Work」「School」「Client」から、今回は「Personal(個人)」を選びます。

The screenshot shows the Netlify account verification process. At the top, there's a green header bar with the message "Your email is now verified!". Below it, a large button says "Nice to meet you! Let's get acquainted.". A question "How are you planning to use Netlify?" follows, with four options: "Personal" (selected), "Work", "School", and "Client". A note "(Required field)" is next to "Personal". The next section asks "What kind of site do you want to build first?", with options: "Documentation", "Games", "Personal site" (selected), "Company site", "Blog", "eCommerce", "Web app", and "Something else". The "Personal site" option is highlighted with a teal border. The next section asks "What best describes your role?", with options: "Freelancer", "Hobby Developer" (selected), and "Other". The "Hobby Developer" option is highlighted with a teal border. The final section asks "What is the name of your team?", with a text input field containing "AtelierMirai". A note below says "People who use Netlify for personal work often use a project name or their own name." At the bottom, a teal button labeled "Continue to deploy" is highlighted with a red box.

What kind of site do you want to build first?(まず、どんなサイトを作りたいですか？)との質問には、いくつか選択肢がありますが、「Personal site」にしておきます。

What best describes your role?(あなたの役割は何ですか？)では、「Freelancer」「Hobby Developer」「Other」と選択肢がありますので、「Hobby Developer(趣味の開発者)」にします。

最後に What is the name of your team?(チーム名は何ですか？)と問われます。People who use Netlify for personal work often use a project name or their own name.(個人でNetlifyを使っている人は、プロジェクト名や自分の名前を使うことが多いようです。)と注釈があります。プロジェクト名やご自身の名前などお好みで入力なさってください。

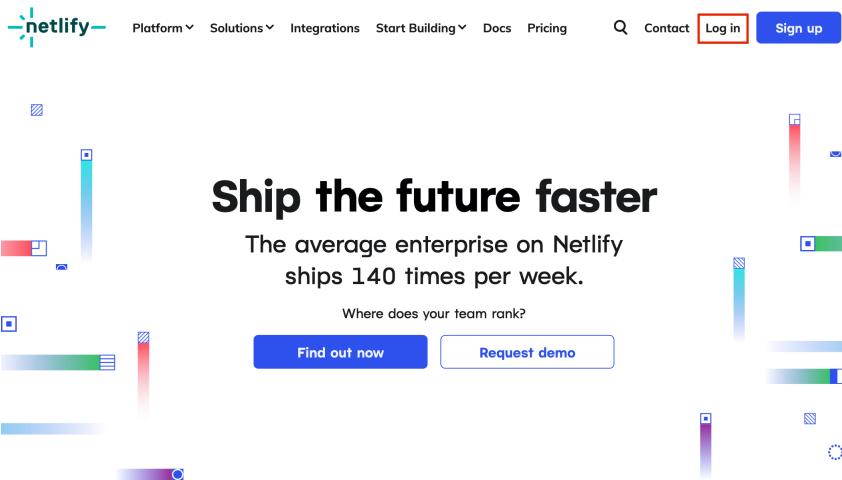
最後に、「Continue to deploy」(デプロイを続ける)ボタンを押します。これで会員登録は完了です。

## 6.3 ネットリファイ Netlify の利用方法

### ♣ ログインする

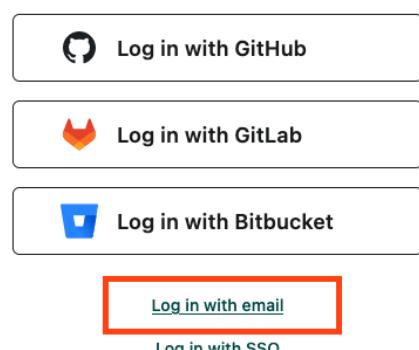
Netlify にログインするには次のようにします。

1. ブラウザを開いて [Netlify<sup>\\*4</sup>](https://www.netlify.com/) にアクセスし、右上の青い「Sign up ボタン」の左側にある「Log in」を押します。



2. (GitHub 等と連携することもできますが) ここでは「Log in with email」を押します。

**Login**

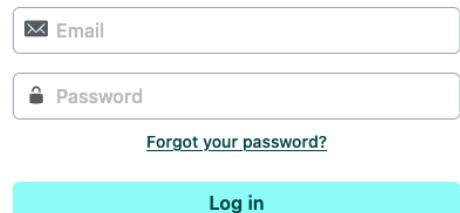


No account yet? [Sign up](#)

<sup>\*4</sup> <https://www.netlify.com/>

3. メールアドレスとパスワードを入力して、「Log in」ボタンを押します。

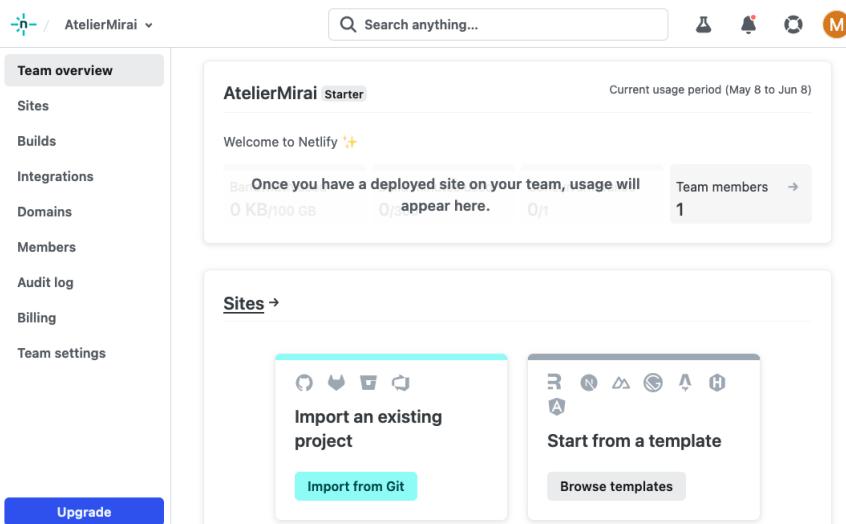
### Log in to Netlify



The image shows the Netlify login interface. It features two input fields: 'Email' with an envelope icon and 'Password' with a lock icon. Below these is a link 'Forgot your password?'. A large blue 'Log in' button is centered at the bottom. At the very bottom, there's a note about GitHub, GitLab, or BitBucket, followed by a link 'Log in with a Git provider'.

Don't have an account yet? [Sign up](#)

4. ログインすると、「Team overview」の画面が開きます。



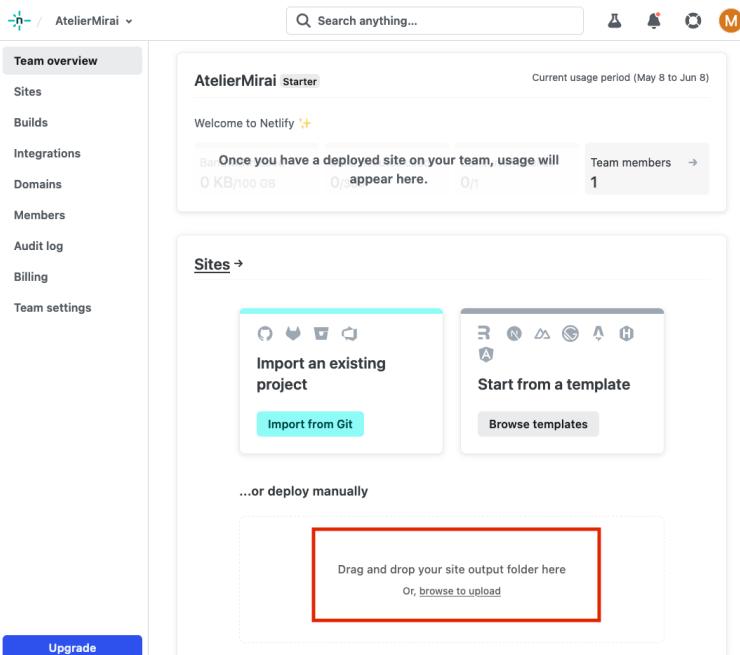
The image shows the Netlify Team overview page for the 'AtelierMirai' team. The left sidebar lists 'Team overview', 'Sites', 'Builds', 'Integrations', 'Domains', 'Members', 'Audit log', 'Billing', and 'Team settings'. A prominent blue 'Upgrade' button is at the bottom of this sidebar. The main content area displays the team's usage statistics: '0 KB/100 GB', '0/1', and '1 Team members'. It also features sections for 'Import an existing project' (with a 'Import from Git' button) and 'Start from a template' (with a 'Browse templates' button). A search bar at the top right says 'Search anything...'.

## ♣ デプロイ（配備・配信）方法

作成したウェブサイトを公開することを、デプロイ（配備・配信）と呼びます。Weblio 英和辞典によると **deploy** とは「〈部隊・兵力などを〉展開する、配置する。」意味の英単語です。IT 分野では開発したソフトウェアなどを実用に供することを意味する用語として、用いられています。

作成したウェブサイトを公開・デプロイ・配備・配信するためには次のようにします。

1. ログインすると、「Team overview」の画面が開いています。



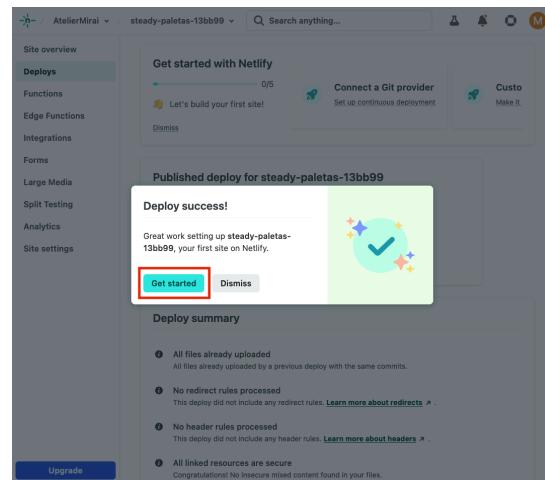
「Sites 欄」には今まで公開したサイトが表示されますが、今はまだ何も公開したサイトがないので、代わりに公開方法の案内が表示されています。

ギットハブ GitHub 等を使っている場合には「Import an existing project(既存のプロジェクトを取り込む)」「Import from Git(Git から取り込む)」ボタンを押すことで デプロイ（配備・配信）できます。

ここでは、手動で公開することにしましょう。画面中央下の「Drag and drop your site output folder here(サイトの出力フォルダをここにドラッグ&ドロップしてください)」に、公開したサイト一式を納めたフォルダを、ドラッグ&ドロップすると公開できます。

2. しばらくすると、デプロイ（配備・配信）が完了します。`steady-paletas-13bb99` というサイト名で作成されたようです。

公開したウェブサイトへの設定を続けるために、「Get Started(行動に移る)」ボタンを押します。



## ♣ デプロイしたサイト名を変更する

1. 「Team overview」の画面になります。チームの活動状況が一覧できる画面です。「Sites」を押すと、今までにデプロイしたサイトの一覧画面へと遷移します。

The screenshot shows the Netlify Team overview for 'AtelierMirai'. The left sidebar has options like 'Team overview', 'Sites' (which is highlighted with a red box), 'Builds', 'Integrations', 'Domains', 'Members', 'Audit log', 'Billing', and 'Team settings'. The main content area starts with a 'Get started with Netlify' section. Below it is a summary for 'AtelierMirai Starter' with metrics: Bandwidth used (0 KB/100 GB), Build minutes used (0/300), Concurrent builds (0/1), and Team members (1). There's a note about importing environment variables. At the bottom, there's a 'Sites' section with a table showing one site: 'steady-paletas-13bb99' (Manual deploys). A blue 'Upgrade' button is at the bottom left.

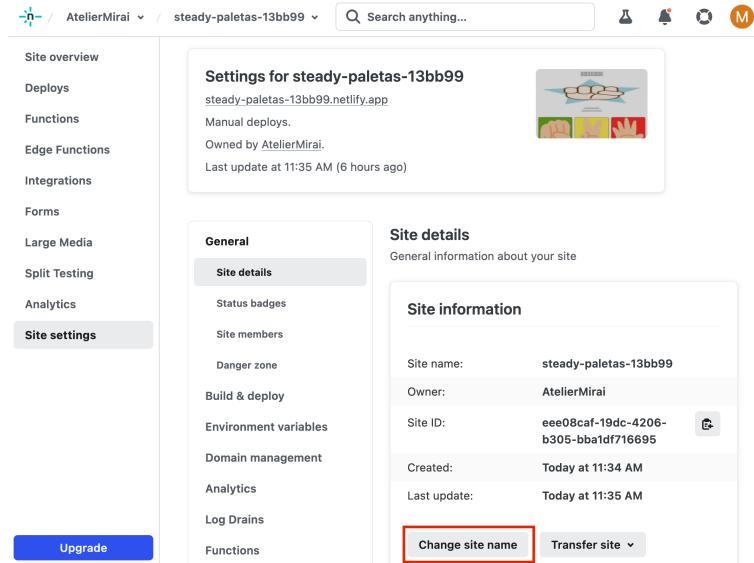
2. 「steady-paletas-13bb99」と付けられたサイト名は、Netlify が適宜命名したものなので、改名しましょう。「steady-paletas-13bb99」をクリックすると、設定画面へ遷移します。

The screenshot shows the Netlify team overview interface. On the left, a sidebar lists 'Team overview', 'Builds', 'Integrations', 'Domains', 'Members', 'Audit log', 'Billing', and 'Team settings'. The 'Sites' option is selected. In the main area, there's a search bar and a button to 'Add new site'. A site card for 'steady-paletas-13bb99' is displayed, showing it was owned by 'AtelierMirai', last published at 11:34 AM (6 hours ago), and has manual deploys. Below the card is a dashed box with instructions for deploying without Git: 'Want to deploy a new site without connecting to Git? Drag and drop your site output folder here' or 'Or, browse to upload'.

3. 「Site settings」を押すと、「steady-paletas-13bb99」に関する様々な設定を行えます。もちろん名称を変更できますので、「Site settings」ボタンを押しましょう。

The screenshot shows the 'Site settings' page for the site 'steady-paletas-13bb99'. The sidebar includes options like 'Deploy', 'Functions', 'Edge Functions', 'Integrations', 'Forms', 'Large Media', 'Split Testing', 'Analytics', and 'Site settings', with 'Site settings' highlighted. The main content area features a 'Set up your site' section with three numbered steps: 1. Your site is deployed (with a green checkmark), 2. Set up a custom domain (with a right-pointing arrow), and 3. Secure your site with HTTPS. A note at the bottom states: 'Netlify DNS can turn your deployed branches into their own subdomains'.

4. 様々な設定を行うことが出来ますが、名称変更の為には 下に表示されている「Change site name」ボタンをクリックします。



AtelierMirai / steady-paletas-13bb99 / Search anything... M

Site overview Deploy Functions Edge Functions Integrations Forms Large Media Split Testing Analytics Site settings Upgrade

**Settings for steady-paletas-13bb99**  
steady-paletas-13bb99.netlify.app  
Manual deploys.  
Owned by AtelierMirai.  
Last update at 11:35 AM (6 hours ago)

**General** Site details Status badges Site members Danger zone Build & deploy Environment variables Domain management Analytics Log Drains Functions

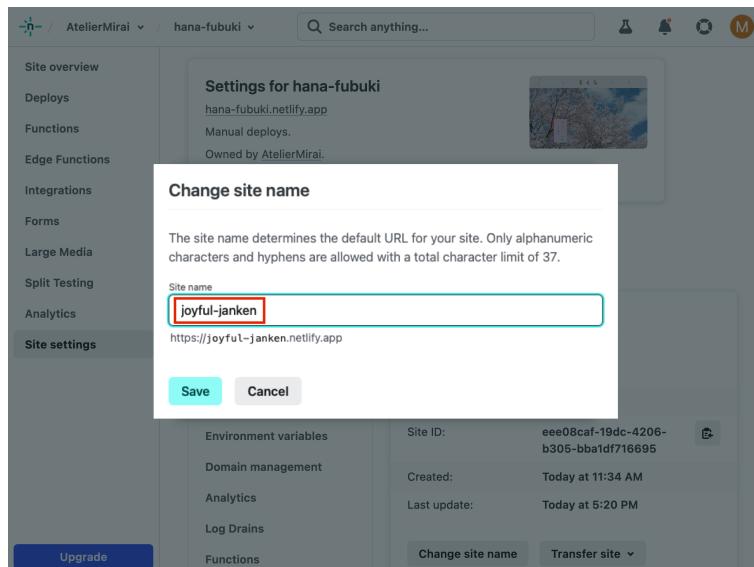
**Site details**  
General information about your site

**Site information**

Site name:	steady-paletas-13bb99
Owner:	AtelierMirai
Site ID:	eee08caf-19dc-4206-b305-bba1df716695
Created:	Today at 11:34 AM
Last update:	Today at 11:35 AM

**Change site name** Transfer site

5. Site name 欄に新しいサイト名を入力します。アルファベット数字の他 -(ハイフン) が使えます。入力し終えたら、Save ボタンを押します。



AtelierMirai / hana-fubuki / Search anything... M

Site overview Deploy Functions Edge Functions Integrations Forms Large Media Split Testing Analytics Site settings Upgrade

**Settings for hana-fubuki**  
hana-fubuki.netlify.app  
Manual deploys.  
Owned by AtelierMirai.

**Change site name**

The site name determines the default URL for your site. Only alphanumeric characters and hyphens are allowed with a total character limit of 37.

Site name  <https://joyful-janken.netlify.app>

**Save** **Cancel**

**Site information**

Environment variables	Site ID: eee08caf-19dc-4206-b305-bba1df716695
Domain management	Created: Today at 11:34 AM
Analytics	Last update: Today at 5:20 PM

Change site name Transfer site

6. サイト名を joyful-janken.netlify.app に変更することができました。

The screenshot shows the Netlify site overview for the 'AtelierMirai' organization. The sidebar includes links for Site overview, Deploys, Functions, Edge Functions, and Integrations. The main area displays the 'Settings for joyful-janken' page, which shows the site URL 'joyful-janken.netlify.app' highlighted with a red box. Other details include 'Manual deploys.', 'Owned by AtelierMirai.', and 'Last update at 5:25 PM (a few seconds ago)'. A small icon of hands playing rock-paper-scissors is also visible.

joyful-janken.netlify.app をクリックすると、今まで手元のコンピュータ上で動作確認していた「じゃんけんゲーム」が、今や世界中の人々に見られるように公開されています。



## ♣ デプロイしたサイトを更新する

公開したウェブサイトは、次の手順で更新することが出来ます。

1. 画面左上の Netlify のロゴマークをクリックします。Team overview の画面に遷移します。

The screenshot shows the Netlify Team overview page for the 'AtelierMirai' team. On the left sidebar, the 'Sites' option is highlighted with a red box. The main content area displays the 'Get started with Netlify' section, followed by usage statistics for the 'AtelierMirai Starter' plan, and a list of sites under the 'Sites' heading. The 'joyful-janken' site is listed, showing its status as 'Manual deploys'. A blue 'Upgrade' button is visible at the bottom left.

2. **Sites** をクリックします。今まで公開しているサイトの一覧が表示されますので、変更したいサイト（ここでは `sakura-fubuki` のみですが）を選び、クリックします。

The screenshot shows the Netlify Site details page for the 'joyful-janken' site. The 'Sites' option in the sidebar is highlighted with a red box. The main content area shows the site's thumbnail, name, owner information, and deployment status. Below this, there is a dashed box containing instructions for deploying a new site without connecting to Git. At the bottom, there are links to various Netlify documentation and support pages.

3. Site overview の画面が表示されます。このサイトに関する様々な設定を行うことが出来ます。Deploys をクリックします。

The screenshot shows the Netlify Site overview page for the site 'joyful-janken'. The left sidebar has a 'Deploys' button highlighted with a red box. The main content area displays the site's URL (<https://joyful-janken.netlify.app>), manual deployment status, and a last published time (11:34 AM). Below this are three numbered steps for setting up the site: 1. Your site is deployed ✓, 2. Set up a custom domain →, and 3. Secure your site with HTTPS. A note at the bottom says you can change the default Netlify subdomain by going to the Production domains panel. A blue 'Upgrade' button is visible at the bottom left.

4. 画面中央 Deploys の欄に「Need to update your site? Drag and drop your site output folder here. サイトの更新が必要ですか？ サイトの出力フォルダをここにドラッグ&ドロップしてください。」と表示されていますので、 ドラッグ&ドロップ すると更新完了です。

The screenshot shows the Netlify Deploys page. The left sidebar has a 'Deploys' button highlighted with a red box. The main content area shows a 'Production Published' status with a 'No deploy message' and a timestamp ('Today at 11:34 AM'). Below this is a large red-bordered box containing instructions: 'Need to update your site? Drag and drop your site output folder here' and 'Or, browse to upload'.

## ♣ 新しいサイトを追加する

今まで公開しているウェブサイトに加えて、また別のウェブサイトを公開したい場合もあると思います。次の手順で追加することが出来ます。

1. 画面左上の Netlify のロゴマークをクリックします。Team overview の画面に遷移します。

2. Sites をクリックします。今まで公開しているサイトの一覧が表示されます。

画面中央下に「Want to deploy a new site without connecting to Git? Drag and drop your site output folder here. Git に接続することなく、新しいサイトをデプロイしたいですか？ サイトの出力フォルダをここにドラッグ＆ドロップしてください。」と表示されていますので、ドラッグ＆ドロップすると更新完了です。

## ♣ 公開したサイトを削除する

何らかの理由で今まで公開していたウェブサイトを削除したい場合もあると思います。次の手順で削除することが出来ます。

1. 画面左上の Netlify のロゴマークをクリックします。Team overview の画面に遷移します。

The screenshot shows the Netlify Team overview interface. On the left, a sidebar lists 'Team overview' sections: Sites, Builds, Integrations, Domains, Members, Audit log, Billing, and Team settings. A blue 'Upgrade' button is at the bottom. The main area has a 'Get started with Netlify' section with a progress bar at 1/5 and a 'Let's build your first site!' button. Below it is a card for 'AtelierMirai Starter' with resource usage statistics: Bandwidth used (1 MB/100 GB), Build minutes used (0/300), Concurrent builds (0/1), and Team members (1). A search bar at the top right says 'Search anything...'.

2. **Sites** をクリックします。今まで公開しているサイトの一覧が表示されますので、削除したいサイト（ここでは joyful-janken のみですが）を選び、クリックします。

This screenshot shows the same Netlify Team overview page, but with the 'Sites' section highlighted in red in the sidebar. The main area displays a list of sites, with 'joyful-janken' selected and highlighted in red. This site card includes a thumbnail, the name 'joyful-janken', 'Manual deploys', 'Owned by AtelierMirai', and 'Last published at 11:34 AM (8 hours ago)'. Below the site list is a dashed box containing deployment instructions: 'Want to deploy a new site without connecting to Git? Drag and drop your site output folder here' and 'Or, [browse to upload](#)'. The bottom of the page includes links for Docs, Pricing, Support, Blog, Changelog, Terms, and a copyright notice: '© 2023 Netlify'.

3. Site overview の画面が表示されます。このサイトに関する様々な設定を行うことが出来ます。Site settings をクリックします。

Site overview

- Deploys
- Functions
- Edge Functions
- Integrations
- Forms
- Large Media
- Split Testing
- Analytics
- Site settings**

**joyful-janken**

- https://joyful-janken.netlify.app
- Manual deploys.
- Last published at 11:34 AM.

**Site settings** **Domain settings** **Favorite site**

**Set up your site**

- Your site is deployed ✓  
Try a test build and deploy, directly from your Git repository or a folder.
- Set up a custom domain →**  
Buy a new domain or set up a domain you already own.
- Secure your site with HTTPS  
Your site is secured automatically with a Let's Encrypt certificate.

With Netlify CLI, you can [share your development server](#) over HTTPS.

**Upgrade**

4. Danger one をクリックします。

Site overview

- Deploys
- Functions
- Edge Functions
- Integrations
- Forms
- Large Media
- Split Testing
- Analytics
- Site settings**

**Settings for joyful-janken**

- joyful-janken.netlify.app
- Manual deploys.
- Owned by AtelierMirai.
- Last update at 5:25 PM (3 hours ago)

**General**

**Danger zone**

**Site details**

General information about your site

Site name:	sakura-fubuki
Owner:	AtelierMirai
Site ID:	eee08caf-19dc-4206-b305-bba1df716695
Created:	Today at 11:34 AM
Last update:	Today at 5:25 PM

**Site information**

Change site name Transfer site ▾

**Upgrade**

5. Delete this site をクリックします。

The screenshot shows the Netlify Site overview interface. On the left sidebar, 'Site settings' is selected. In the main content area, under the 'Danger zone' section, there is a large red warning box containing the text: 'Once you delete a site, there is no going back.' Below this is a prominent red button labeled 'Delete this site'. At the bottom of the page, there is a footer with links to Docs, Pricing, Support, Blog, Changelog, Terms, and an Upgrade button.

6. 注意書きが表示されます。

Are you absolutely sure you want to delete joyful-janken?  
If you have submitted a support request about this site, it will be difficult  
for Netlify's Support team to help you debug the situation if you delete it.

Remember to remove any DNS records that point to this site's URL.

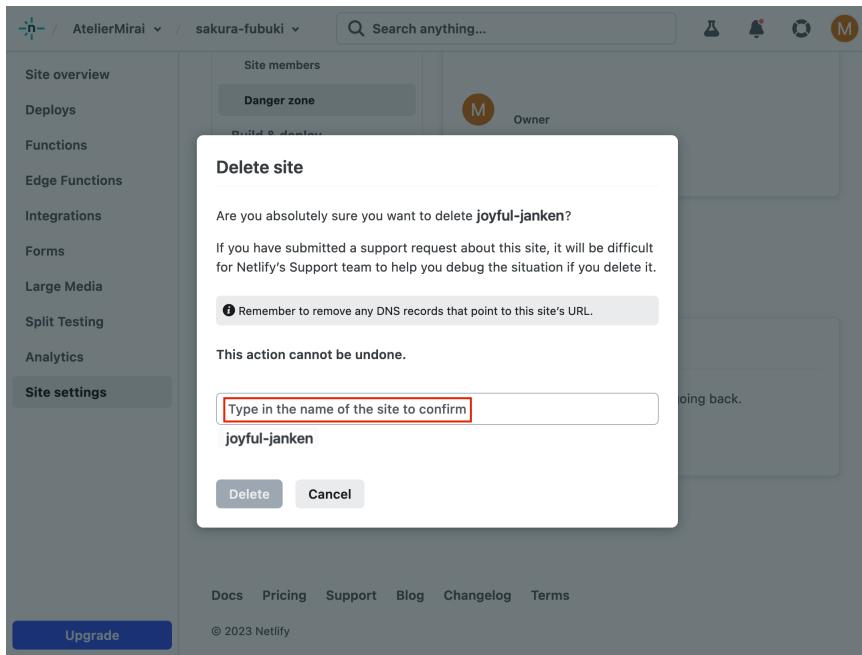
**This action cannot be undone.**

本当にjoyful-jankenを削除していいのでしょうか？  
このサイトについてサポートリクエストを提出している場合、削除してしまうと、  
サポートチームが状況をデバッグするのを助けるのが難しくなってしまいます。

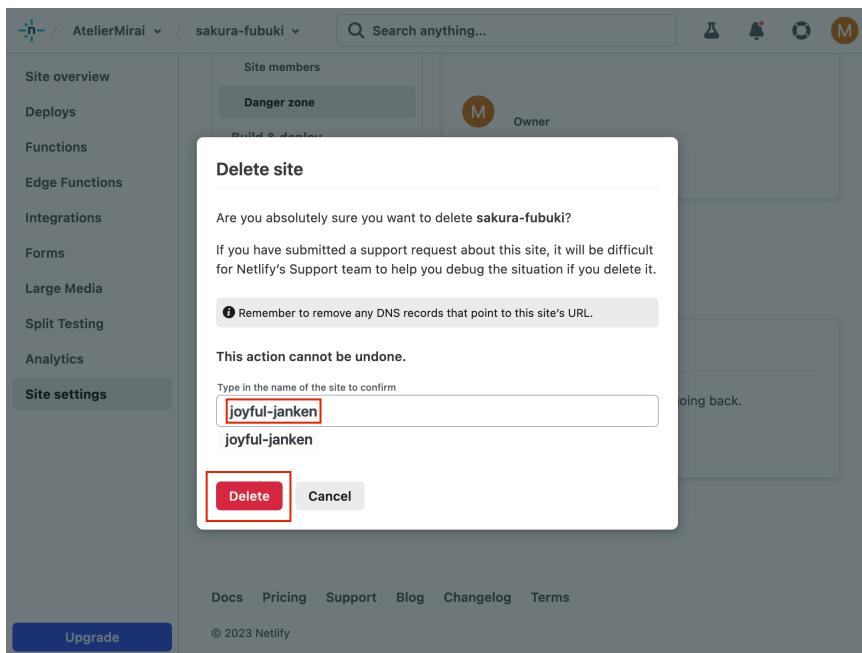
このサイトの URL を指す DNS レコードを削除することを忘れないでください。

**この操作は元に戻せません。**

注意書きを了承した上で、「Type in the name of the site to confirm. 確認のためサイト名を入力します。」と書かれていますので、サイト名を入力します。

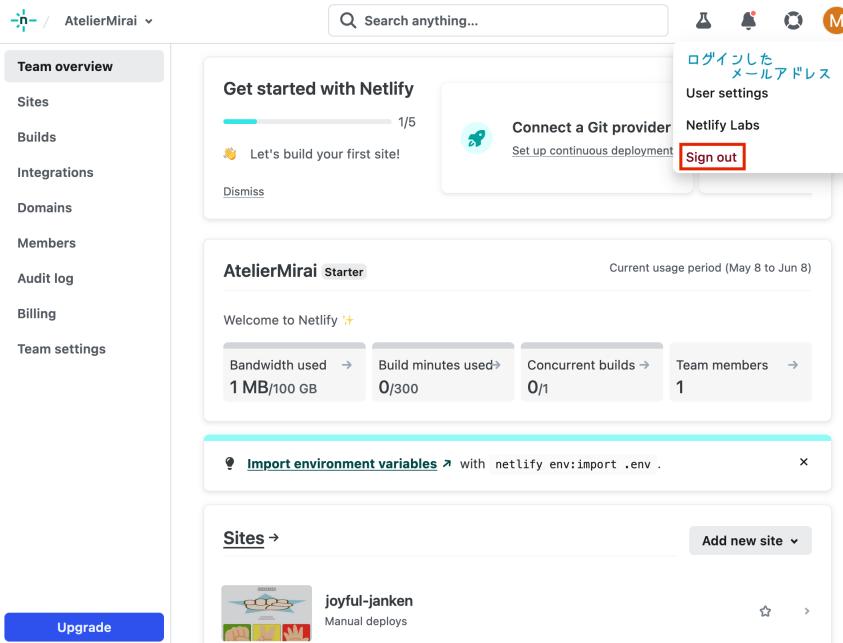


7. サイト名を入力すると、「Delete」ボタンが押せるようになりますので、クリックして削除は完了します。復活はできないので注意して使ってください。



## ♣ ログアウトする

1. ログアウトするためには、右上の「M」ボタン（会員登録した名前によって変わります）をクリックします。いろいろなメニューが表示され、様々な設定を行うことができます。一番下の「Sign out」をクリックし、終了です。





## 付録 A

# 珠玉の名著のご紹介

次の段階へ進むために読んで欲しい、名著をご紹介いたします。 \*1

### ♣ CSS グリッドで作る HTML5 & CSS3 レッスンブック



本書は CSS グリッドを基礎にした Web ページ制作を行うための解説書です。CSS グリッドを基礎にすると、Web ページ制作がシンプルになります。サンプルを作りながら一歩一歩着実に学習することにより、モバイルファーストで本格的なレスポンシブに対応した実践的な Web 制作に関する知識がひと通り得られます。

- これから HTML5 & CSS3 を使ったサイト構築を学ぶ人
  - 最新の CSS グリッドに関する知識を得たいと考える人
- に最適の一冊です。

### ♣ 作って学ぶ HTML & CSS モダンコーディング



モバイルファースト&レスポンシブで、サイトを制作していく過程を実際に操作しながら学びます。

サイトはパーツ単位で作成し、章ごとに 1 つのパーツを作成していきます。ヘッダー / ヒーロー / フッター / 記事 / ナビゲーションなど、各パーツの作成にあたっては、パーツのレイアウトを実現する CSS の選択肢を示し、適切なものを選択して作成します。

HTML は最新の「HTML Living Standard」に準拠し、CSS では従来から活用されてきたメディアクエリの他、Flexbox、CSS Gridなどのレイアウトのコントロール、CSS 関数を使いこなします。

\*1 書籍紹介文から、引用・改変。

## ♣ スラスラ読める JavaScript ふりがなプログラミング



「プログラムの読み方をすべて載せる（ふりがなをふる）」という手法で究極のやさしさを目指した、まったく新しい JavaScript（ジャバスクリプト）の入門書です。本書内に登場するプログラムの読み方をすべて載せ、さらに、漢文訓読の手法を取り入れ、読み下し文を用意。プログラムの1行1行が何を意味していて、どう動くのかが理解できます。

この手法により「プログラムが読めず、自分がいま何をしているか分からぬ」といったプログラミング入門者が挫折する原因を解決しました。また、実際に手を動かしプログラムを考えるため、しっかり JavaScript の基礎文法を身につけられます。

## ♣ JavaScript Primer 迷わないと学ぶ入門書



これから JavaScript を学びたい人が、ECMAScript 2015 以降をベースにして一から JavaScript を学べる書籍です。

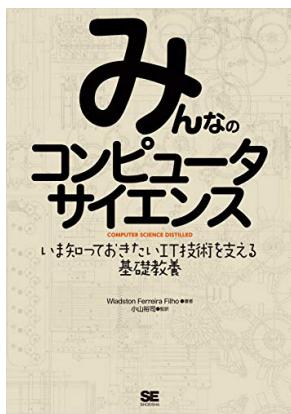
この書籍は、JavaScript の仕様に対して真剣に向き合って書かれています。入門書であるからといって、極端に省略して不正確な内容を紹介することは避けています。そのため、JavaScript の熟練者であっても、この書籍を読むことで発見があるはずです。

## ♣ 改訂2版 わかばちゃんと学ぶ Git 使い方入門



せっかく学ぶなら、やっぱり楽しい方がいい。「Gitって難しそう」「勉強しようとは思っているけど、なかなか一步が踏み出せない」そんな方のために、楽しく Git を理解できる本を作りました。・個性的なキャラクターたちが登場するマンガ・感覚的にわかる図解・丁寧な実践パート 上記3つの特長で、Gitを無理なく学べます。仕事に必要な基本の使い方はもちろん、サンプルデータが使えるので、プルリクエストの練習・GitHub PagesでのWebページ公開もできます。開発現場のリアルな声を反映。セクションごとに応用コマンドを掲載。さらなるレベルアップも可能に。付録として「コマンド操作に挑戦!」も追加されています。

## ♣ みんなのコンピュータサイエンス



コンピュータなしには生活が立ち行かなくなる水準に達しつつある現代社会。その圧倒的な力を課題解決に援用するには小手先の知識では追いつきません。とは言え無闇に全方位に知識を求めるには、その世界は広すぎ、効率も悪すぎます。

本書は計算機科学が扱う「基礎」「効率」「戦略」「データ」「アルゴリズム」「データベース」「コンピュータ」「プログラミング」という8つのジャンルにしぶり、その精髄と背景となる考え方を紹介します。

ステップアップしたいエンジニアや、ライトに全体像を俯瞰したい学生にも最適な1冊です。

## ♣ プログラマの数学



プログラミングに役立つ「数学的な考え方」を身につけよう。

プログラミングや数学に関心のある読者を対象に、プログラミング上達に役立つ「数学の考え方」をわかりやすく解説しています。数学的な知識を前提とせず、たくさんの図とパズルを通して、平易な文章で解き明かしています。

二進数から人工知能に至るまで、ていねいに説明しています。

プログラミングや数学に関心のある読者はいうまでもなく、プログラミング初心者や数学の苦手な人にとっても最良の一冊です。

## ♣ 数学ガール



本書は、三人の高校生が数学の問題に挑戦する物語。題材は「素数」「絶対値」という基本的なものから「フィボナッチ数列」「二項定理」「無限級数」や「テイラーフィーリー展開」「母関数」まで多岐にわたっています。

数学クイズが好きな一般の方から、理系の大学生、社会人まで幅広い読者に楽しんでもらえる数学物語です。数式が苦手でも大丈夫。登場する高校生自身も数式で悩み、ああでもない、こうでもないと読者と思いを共有します。数式が追えなくても「旅の地図」と称した概念図で読者さんの理解を助けます。「数学は、時を越える」をテーマにおいた本書は本格的な数学の奥深いおもしろみをすべての読者に提供するでしょう。

## ♣ 教養としてのコンピューターサイエンス講義



デジタル時代で活躍するための「教養」をこの1冊で身につけよう。  
プリンストン大学の一般人向け「コンピューターサイエンス」の講義が一冊に。デジタル社会をよりよく生きるために知識を伝説の計算機科学者がやさしく伝えます。(著者ブライアン・カーニハーン氏は、C言語の発明者です)

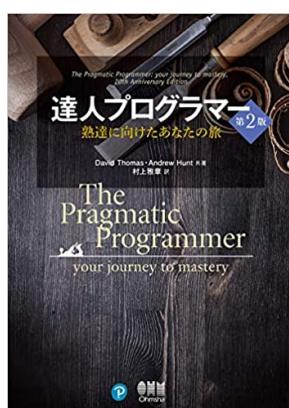
本書は、わたくしたちの世界(デジタル社会)が、どのように動いているのか、なぜそのしくみになっているのかをもっとも明快かつ簡潔に説明しています。

## ♣ キタミ式イラスト IT 塾 IT パスポート



可愛いイラストでとてもわかりやすい解説を行っているため、ITパスポート試験にとって、まず大切な「解説書を一冊読み、用語や計算に慣れる」ことができる書籍です。

## ♣ 達人プログラマー(第2版) 熟達に向けたあなたの旅



本書は、より効率的、そしてより生産的なプログラマーになりたいと願うソフトウェア開発者に向けて、アジャイルソフトウェア開発手法の先駆者として知られる二人により執筆されました。経験を積み、生産性を高め、ソフトウェア開発の全体をより良く理解するための、実践的なアプローチが解説されています。先見性と普遍性に富んだ本書は、入門者には手引きとなり、ベテランでも読み直すたびに得るものがある、座右の一冊です。

## ♣ コーディングを支える技術



本書は、プログラミング言語が持つ各種概念が「なぜ」存在するのかを解説する書籍です。世の中にはたくさんのプログラミング言語があります。そしてプログラミングに関する概念も、関数、型、スコープ、クラス、継承など、さまざまなものがあります。多くの言語で共通して使われる概念もあれば、一部の言語でしか使われない概念もあります。これらの概念は、なぜ生まれたのでしょうか。本書のテーマは、その「なぜ」を理解することです。

そのために本書では、言語設計者の視点に立ち、複数の言語を比較し、そして言語がどう変化してきたのかを解説します。いろいろな概念が「なぜ」生まれたのかを理解することで、なぜ使うべきか、いつ使うべきか、どう使うべきかを判断できるようになるでしょう。

## ♣ アルゴリズム図鑑 絵で見てわかる 33 のアルゴリズム



基本的な 33 のアルゴリズム +7 つのデータ構造をすべてイラストで解説。アルゴリズムはどんな言語でプログラムを書くにしても不可欠ですが、現場で教わることはめったになく、かといって自分で学ぶには難しいものです。

本書は、アルゴリズムを独学する人のために作りました。はじめて学ぶときにはイメージしやすく、復習するときには思い出しやすくなるよう、基本的な 33 のアルゴリズム +7 つのデータ構造をすべてイラストにしています。

よいプログラムを書くために知っておかなきゃいけないアルゴリズムの世界を、楽しく学びましょう。

## ♣ C 言語による標準アルゴリズム事典



コンピュータの算法に関わるアルゴリズムの定石、レトリックを可能な限り収録した定番の書。手元に置いておきたい実用的な本が 30 年弱の時を経て新装改訂版として登場です。定評をいただいている基本的な内容はそのままに、時代にそぐわなくなっていた部分を改訂。これからも末長くご愛顧いただけるようにまとめ直しました。

## ♣ 初めてのプログラミング 第2版



初めてプログラミングを学ぶ入門者を対象に、プログラミングの基礎をていねいに解説した書籍。

教材には、誰でもどんな環境でも気軽に使える Ruby を使い、実際に簡単なコードを書きながら理解を深めます。プログラミングとは何かを無理なく理解してもらうために、要点をひとつひとつていねいに解説。簡単な概念から始めて、かなり高度なプログラミングの知識まで身に付けられます。

プログラミングを学ぶ最初の一冊に最適な入門書です。

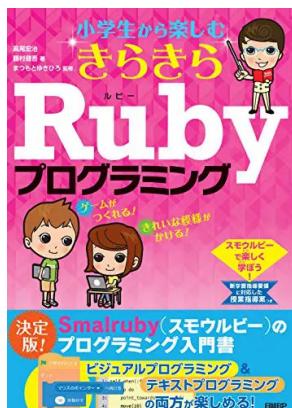
## ♣ プロを目指す人のための Ruby 入門



本書は、プログラミング言語 Ruby の言語仕様や開発の現場で役立つ Ruby の知識を説明した本です。豊富なサンプルコードで文法を学び、例題でプログラミングの流れを体験できます。初心者の目線にたった丁寧な解説が好評で多くの Ruby 初学者に愛読され、いまや Ruby 入門書の定番とも言える存在です。

本書の内容を理解すれば、開発の現場で必要とされる Ruby 関連の知識を一通り習得できます。そして「今まで呪文のようにしか見えなかった不思議な構文」や「実はあまりよくわからないまま見よう見まねで書いているコード」も自信をもって読み書きできるようになるはずです。

## ♣ 小学生から楽しむ きらきら Ruby プログラミング



スマウルビー解説書の決定版! ブロック(ビジュアル)プログラミング言語「Scratch」とテキストプログラミング言語「Ruby」の両方の特徴を持つ「Smalruby」を使ったプログラミング入門書です。Scratch 同様に簡単にプログラミングを始められ、さらにテキスト言語への移行もスムーズに行えるよう、ブロックとテキストの両方でプログラムを書く方法を丁寧に解説。新学習指導要領に対応した、実際の授業でも使われている授業指導案も付属。

この本では小学校でするプログラミングの内容を、音楽、社会、算数、理科といった各教科に分けてできるようになっていて、プログラミングがはじめての人にも経験している人にもバッチリな内容。

## 付録 B

# 付録: 参考リンク集

### ♣ MDN Mozilla 公式チュートリアル<sup>\*1</sup>

Mozilla 公式ウェブサイト。さまざまなチュートリアルとトレーニング用教材へのリンク集。初心者の方からベテランの方まで、学習に役に立つ教材が見つかります。

### ♣ タイピングクラブ<sup>\*2</sup>

「F」「J」のホームポジションから始まり、数字や記号に至るまで滑らかに入力できるよう練習できるサイト。円滑に文字入力できるよう、毎日こつこつ練習しましょう。

### ♣ プログラミング練習

#### ドットインストール<sup>\*3</sup>

すべてのレッスンは 3 分以内の動画で提供されており、無理なく気軽に学べます。

#### Progate<sup>\*4</sup>

紙の本よりも直感的で、動画よりも学びやすい、「スライド学習」を採用した学習サイト。自分のペースで学習できること、復習しやすいことが強みです。

#### Ruby on Rails チュートリアル<sup>\*5</sup>

初心者からの成長を目指す方にお薦め。人気の Ruby on Rails を学ぶことで、クリックパッドや食べログのようなウェブサイトの作成も可能になります。

### ♣ 技術用語

#### IT 用語辞典<sup>\*6</sup>

このサイトは IT 用語のオンライン辞典です。情報・通信技術に関連する用語の意味や読み方、関連用語などを、キーワード検索や五十音索引から調べられます。

#### Wikipedia<sup>\*7</sup>

インターネット百科事典。玉石混交ですが、有益な記事も多く掲載されています。

\*1 <https://developer.mozilla.org/ja/docs/Web/Tutorials>

\*2 <https://www.typing.com/student/lessons>

♣ **ぱくたそ<sup>\*8</sup>, pro.foto<sup>\*9</sup>, BEIZ<sup>\*10</sup>, 足成<sup>\*11</sup>, Pixabay<sup>\*12</sup>, 写真 AC<sup>\*13</sup>**

様々なサイトが素晴らしい写真素材を提供しています。

♣ **WebDesignClip<sup>\*14</sup>**

Web デザインの参考となる品質の高い国内の Web デザイン・クリップ集です。Web 制作におけるアイデア・技術に優れたサイトをクリップしています。

♣ **MacBook Air<sup>\*15</sup>**

美しく洗練されたデザイン、心地よく優れた利用者体験を提供する macOS のもと、プログラミングを始める方への最初の一台として最適な機種です。

♣ **コミュニティ主導の高性能テキストエディタ pulsar<sup>\*16</sup>**

プログラマが心地よくコーディングできるよう、必要な機能が搭載されています。始めての方は設定不要で初日から使え、熟練者は深部まで調整可能です。

♣ **Mozilla 謹製ブラウザ Firefox<sup>\*17</sup>**

インターネット草莽期の Mosaic、NetScape の血を受け継ぐ Mozilla 財団製ブラウザ。プライバシー保護や、CSS グリッドの確認、分かりやすいエラー表示などが特徴。

♣ **DeepL<sup>\*18</sup>, Weblio 英語翻訳<sup>\*19</sup>**

機械学習 (Deep Learning) 技術を用いた高精度な翻訳サイトと、たくさんの例文と発音も確認することができる翻訳サイトです。

♣ **ウェブサイトの公開サービス Netlify<sup>\*20</sup>**

無料で利用でき、Git リポジトリサービスと連携した自動デプロイが特徴です。

---

\*8 <https://www.pakutaso.com/>

\*9 <https://pro-foto.jp>

\*10 <https://www.beiz.jp>

\*11 <http://www.ashinari.com>

\*12 <https://pixabay.com/ja/>

\*13 <https://www.photo-ac.com>

\*14 <https://www.webdesignclip.com>

\*15 <https://www.apple.com/jp/macbook-air/>

\*16 <https://pulsar-edit.dev>

\*17 <https://www.mozilla.org/>

\*18 <https://www.deepl.com/translator>

\*19 <https://translate.weblio.jp>

\*20 <https://www.netlify.com>

## 付録 C

# HTML / CSS / JavaScript 簡易まとめ

HTML / CSS / JavaScript に関する簡易なまとめです。タグを使用して作成される HTML 要素を一覧表示しています。見つけやすいように、機能別にグループ化しています。

HTML 要素リファレンス<sup>\*1</sup>より、抄訳しております。引用元にはより詳細な解説や使用例等が掲載されておりますので、是非ご活用下さい。

### C.1 HTML 簡易まとめ

HTML は Hyper Text Markup Language 超文書印付け言語 の意味で、ウェブサイトにおける文書構造の記述に用います。

#### ♣ コメント

プログラミング言語では、ソースコード中に記述されるがコードとしては解釈されない、人に向けた文字列をコメントといいます。主にコードの記述者が別の開発者などにコードの意味や動作、使い方、注意点等について注釈や説明を加える為に使われます。<sup>\*2</sup>

HTML では、コメントは以下のように記述します。

記述例	説明
<!-- コメント -->	コメント

\*1 <https://developer.mozilla.org/ja/docs/Web/HTML/Element>

\*2 出典：IT 用語辞典

## ♣ メインルート

要素	説明
<html>	HTML 文書においてルート（基点）となる要素（トップレベル要素）であり、ルート要素とも呼ばれます。他の全ての要素は、この要素の子孫として配置します。

## ♣ 文書メタデータ

メタデータは、ページに関する情報のことです。これは検索エンジンやブラウザなどが利用する、およびページの描画を支援するスタイル、スクリプト、データといった情報を含みます。スタイルやスクリプトのメタデータはページ内で定義するか、それらの情報を持つ別のファイルへのリンクとして定義します。

要素	説明
<head>	文書に関する機械可読な情報 (metadata)、たとえば題名、スクリプト、スタイルシートなどを含みます。
<link>	外部リソースへのリンク要素です。現在の文書と外部のリソースとの関係を指定します。この要素は CSS へのリンクに最もよく使用されますが、サイトのアイコン (favicon スタイルのアイコンと、モバイル端末のホーム画面やアプリのアイコンの両方) の確立や、その他のことにも使用されます。
<meta>	他のメタ関連要素 (base / link / script / style / title) で表すことができない任意の metadata を提示します。
<style>	文書あるいは文書の一部分のスタイル情報を含みます。
<title>	題名要素です。ブラウザのタイトルバーやページのタブに表示される文書の題名を定義します。

## ♣ 区分化ルート

要素	説明
<body>	HTML 文書のコンテンツを示す要素で、<body>要素は一つだけ配置できます。

## ♣ コンテンツ区分

コンテンツ区分要素は、文書のコンテンツを論理的な断片に体系づけます。ページのコンテンツでヘッダーやフッターのナビゲーション、あるいはコンテンツのセクションを識別する見出しなどの、大まかなアウトラインを作成するために区分要素を使用します。

要素	説明
<address>	これを含んでいる HTML が個人、団体、組織の連絡先を提供していることを示します。
<article>	文書、ページ、アプリケーション、サイトなどの中で自己完結しており、（集合したものの中で）個別に配信や再利用を行うことを意図した構成物を表します。
<aside>	文書のメインコンテンツと間接的な関係しか持っていない文書の部分を表現します。
<footer>	直近の区分コンテンツまたは <body> 要素のフッターを表します。フッターには通常、そのセクションの著作者に関する情報、関連文書へのリンク、著作権情報等を含めます。
<header>	導入部やナビゲーション等のグループを表すコンテンツです。見出し要素だけでなく、ロゴ、検索フォーム、作者名、その他の要素を含むこともできます。

要素	説明
<h1> <h2> <h3> <h4> <h5> <h6>	セクションの見出しを 6 段階で表します。 <h1>が最上位で、<h6>が最下位です。
<main>	文書の <body> の主要な内容を表します。主要な内容とは、文書の中心的な主題、またはアプリケーションの中心的な機能に直接関連または拡張した内容の範囲のことです。
<nav>	現在の文書内の他の部分や他の文書へのナビゲーションリンクを提供するためのセクションを表します。ナビゲーションセクションの一般的な例としてメニュー、目次、索引などがあります。
<section>	文書の自立した一般的なセクション（区間）を表します。そのセクションを表現するより意味的に具体的な要素がない場合に使用します。

## ♣ テキストコンテンツ

テキストコンテンツ要素は、開始タグ <body> と終了タグ </body> の間にあるコンテンツでブロックやセクションを編成します。これらの要素はコンテンツの用途や構造を識別するものであり、アクセシビリティ や SEO のために重要です。

要素	説明
<div>	フローコンテンツの汎用コンテナです。CSS を用いて何らかのスタイル付けがされる（例えば、スタイルが直接適用されたり、親要素にグリッドなど何らかのレイアウトモデルが適用されるなど）までは、コンテンツやレイアウトには影響を与えません。
<figure>	図表などの自己完結型のコンテンツを表します。任意で figcaption 要素を使用してキャプション（見出し）を付けることができます。
<figcaption>	親の figure 要素内にあるその他のコンテンツを説明するキャプション（見出し）や凡例を表します。
<ol>	項目の順序付きリストを表します。ふつうは番号付きのリストとして表示されます。
<ul>	項目の順序なしリストを表します。一般的に、行頭記号を伴うリストとして描画されます。
<li>	リストの項目を表すために用いられます。
<p>	テキストの段落を表します。
<hr>	段落レベルの要素間において、テーマの意味的な区切りを表します。例えば、話の場面の切り替えや、節内での話題の転換などです。

## ♣ インライン文字列意味付け

インラインテキストセマンティクス要素は、単語、行、あるいは任意のテキスト範囲の意味、構造、スタイルを定義します。

要素	説明
<a>	アンカー要素は、 <code>href</code> 属性を用いて、別のウェブページ、ファイル、メールアドレス、同一ページ内の場所、または他の URL へのハイパーリンクを作成します。
 	文中に改行（キャリッジリターン）を生成します。詩や住所など、行の分割が重要な場合に有用です。
<b>	注目付け要素です。要素の内容に読み手の注意を惹きたい場合で、他の特別な重要性が与えられないものに使用します。
<em>	強調されたテキストを示します。入れ子にすることができ、入れ子の段階に応じてより強い程度の強調を表すことができます。
<i>	興味深いテキスト要素です。何らかの理由で他のテキストと区別されるテキストの範囲を表します。
<strong>	強い重要性要素です。内容の重要性、重大性、または緊急性が高いテキストを表します。ブラウザは一般的に太字で描画します。
<small>	著作権表示や法的表記のような、注釈や小さく表示される文を表します。既定では、 <code>small</code> から <code>x-small</code> のように、一段階小さいフォントでテキストが表示されます。
<span>	記述コンテンツの汎用的な行内コンテナであり、何かを表すものではありません。 <code>class</code> または <code>id</code> 属性を使用して、スタイル付けのために使用することができます。

## ♣ 画像とマルチメディア

HTML は 画像、音声、映像といった、さまざまなマルチメディアソースをサポートします。

要素	説明
<img>	文書に画像を埋め込みます。
<audio>	文書内に音声コンテンツを埋め込むために使用します。
<video>	映像要素です。文書中に映像再生に対応するメディアプレイヤを埋め込みます。

## ♣ SVG と MathML

SVG と MathML のコンテンツを、`<svg>` および `<math>` 要素を使用して直接 HTML 文書に埋め込むことができます。

要素	説明
<svg>	SVG(Scalable Vector Graphics) 形式の図形描画のための要素です。直線、矩形、円、楕円、多角形、折れ線やベジェ曲線の描画が可能です。
<math>	数式を記述する際に用います。

## ♣ スクリプティング

動的なコンテンツやウェブアプリケーションを作成するために、HTML ではスクリプト言語を使用できます。もっとも有名な言語は、JavaScript です。

要素	説明
<canvas>	<canvas>要素と Canvas スクリプティング API や WebGL API を使用して、グラフィックやアニメーションを描画することができます。
<script>	実行できるコードやデータを埋め込むために使用します。ふつうは JavaScript のコードの埋め込みや参照に使用されます。
<noscript>	このページ上のスクリプトの種類に対応していない場合や、スクリプトの実行がブラウザで無効にされている場合に表示する HTML の部分を定義します。

## ♣ 表(テーブル)

以下の要素は、表形式のデータを作成および制御するために使用します。

要素	説明
<table>	表形式のデータ、つまり、行と列の組み合わせによるセルに含まれたデータによる二 次元の表で表現される情報です。
<caption>	表のキャプション(またはタイトル)を指定します。
<colgroup>	表内の列のグループを定義します。
<col>	表内の列を定義して、全ての一般セルに共通の意味を定義するために使用します。この要素は通常、colgroup要素内にみられます。
<thead>	表の列の見出しを定義する行のセットを定義します。
<tbody>	表本体要素(tbody)は、表の一連の行(tr要素)を内包し、その部分が表(table)の本体部分を構成することを表します。
<tr>	表内でセルの行を定義します。行のセルはtd(データセル)およびth(見出しセル)要素を混在させることができます。
<th>	表のセルのグループ用のヘッダーであるセルを定義します。
<td>	表でデータを包含するセルを定義します。これは表モデルに関与します。
<tfoot>	表の一連の列を総括する行のセットを定義します。

## ♣ フォーム

利用者がデータを記入してウェブサイトやアプリケーションに送信することを可能にするフォームを作成するために組み合わせて用いる要素です。<sup>\*3</sup>

要素	説明
<fieldset>	フォーム内のラベル(label)などのようにいくつかのコントロールをグループ化する ために使用します。
<legend>	fieldsetの内容のキャプション(見出し)を表すために用います。
<form>	サーバに情報を送信するための対話型コントロールを含む文書の区間を表します。
<button>	クリックできるボタンを表し、フォームや、文書で単純なボタン機能が必要なあらゆ る場所で使用することができます。
<input>	利用者からデータを受け取るための、フォーム用の対話的なコントロールを作成する ために使用します。
<textarea>	複数行のプレーンテキスト編集コントロールを表し、問い合わせフォーム等、利用者 が大量の自由記述テキストを入力できるようにするときに便利です。
<label>	ユーザーインターフェイスの項目のキャプション(見出し)を表します。
<datalist>	他のコントロールで利用可能な値を表現する一連のoption要素を含みます。
<select>	選択式のメニューを提供するコントロールを表します。
<optgroup>	select要素内の、選択肢(option)のグループを作成します。
<option>	select要素、optgroup要素、datalist要素内で項目を定義するために使われます。
<output>	出力要素(output)です。サイトやアプリが計算結果やユーザー操作の結果を挿入す ることができるコンテナ要素です。
<meter>	既知の範囲内のスカラ値、または小数値を表します。
<progress>	タスクの進捗状況を表示します。プログレスバーとしてよく表示されます。

<sup>\*3</sup> フォームに関する多くの情報が、HTML フォームガイド (<https://developer.mozilla.org/ja/docs/Learn/Forms>) に掲載されています。

## C.2 CSS 簡易まとめ

CSS は、Cascading Style Sheets の略で、ウェブサイトにおける装飾などの為に用います。CSS: カスケーディングスタイルシート<sup>\*4</sup>より、抄訳しています。引用元には詳細な解説や使用例等が掲載されています。是非ご活用下さい。

### ♣ 構文

カスケーディングスタイルシート (CSS) 言語の基本的な狙いは、ブラウザがページの要素を、色、位置、装飾などの特定の特性をもって描けるようにすることです。その為に、**プロパティ** (人がどのような特性を考えることのできる名前) と、その特性をどのようにブラウザが操作しなければならないか表す**値**の組で表現します。これを**宣言**と呼びます。ページの要素を選択する条件である**セレクタ**により、それぞれの宣言を文書のそれぞれの部品に適用できるようにします。

#### ▼ CSS 構文

```
セレクタ {
  プロパティ1: 値;
  プロパティ2: 値;
  プロパティ3: 値;
}
```

#### ▼ CSS の例

```
header, p.intro {
  background-color: red;
  border-radius: 3px;
}
```

### ♣ セレクタ

#### 基本セレクタ

##### 全称セレクタ

全ての要素を選択します。任意で、特定の名前空間に限定したり、全ての名前空間を対象にしたりすることができます。

例: \* は文書の全ての要素を選択します。

##### 要素型セレクタ

指定されたノード名を持つ全ての要素を選択します。

例: input はあらゆる <input> 要素を選択します。

##### クラスセレクタ

指定された class 属性を持つ全ての要素を選択します。

例: .index は "index" クラスを持つあらゆる要素を選択します。

<sup>\*4</sup> <https://developer.mozilla.org/ja/docs/Web/CSS>

## ID セレクタ

ID 属性の値に基づいて要素を選択します。文書中に指定された ID を持つ要素は 1 つしかないはずです。

例: `#toc` は "toc" という ID を持つ要素を選択します。

## 属性セレクタ

指定された属性を持つ要素を全て選択します。

構文: `[attr] [attr=value] [attr~=value] [attr|=value] [attr^=value]`  
`[attr$=value] [attr*=value]`

例: `[autoplay]` は `autoplay` 属性が (どんな値でも) 設定されている全ての要素を選択します。

## グループ化セレクタ

### セレクターリスト

, (カンマ) はグループ化の手段であり、一致する全てのノードを選択します。

例: `div, span` は `<span>` と `<div>` の両要素に一致します。

### 子孫結合子

半角空白 結合子は、第 1 の要素の子孫にあたるノードを選択します。

例: `div span` は `<div>`要素内の`<span>`要素を全て選択します。

### 子結合子

> 結合子は、第 1 の要素の直接の子に当たるノードを選択します。

例: `ul > li` は `<ul>` 要素の内側に直接ネストされた `<li>` 要素を全て選択します。

### 一般兄弟結合子

~ 結合子は兄弟を選択します。つまり、第 2 の要素が第 1 の要素の後にあり (直後でなくとも構わない)、両者が同じ親を持つ場合です。

例: `p ~ span` は `<p>` 要素の後にある `<span>` 要素を全て選択します。

### 隣接兄弟結合子

+ 結合子は隣接する兄弟を選択します。つまり、第 2 の要素が第 1 の要素の直後にあり、両者が同じ親を持つ場合です。

例: `h2 + p` は `<h2>` 要素の後にすぐに続く `<p>` 要素を全て選択します。

## 擬似表記

### 擬似クラス

: 表記により、文書ツリーに含まれない状態情報によって要素を選択できます。

例: `a:visited` は利用者が訪問済みの `<a>` 要素を全て選択します。

### 疑似要素

:: 表記は、HTML に含まれていない存在 (エンティティ) を表現します。

例: `p::first-line` は全ての `<p>` 要素の先頭行を選択します。

## コメント

CSS では、コメントは以下のように記述します。

記述例	説明
/* コメント */	コメント

## ♣ 良く使う CSS プロパティのご案内

CSS には、100 以上ものプロパティがあり、そしてそれぞれのプロパティが取り得る値も個々に決められています。CSS リファレンス<sup>\*5</sup>に全てが紹介されていますので、詳しくはそちらをご覧ください。ここでは、主なもののみを簡単にご紹介します。

要素	説明
background	色、画像、原点と寸法、反復方法など、背景に関する全てのスタイルプロパティを一括で設定します。
border	要素の境界（枠線）を設定します。これは border-width / border-style / border-color の値を設定します。
border-radius	要素の境界（枠線）の外側の角を丸めます。1 つの半径を設定すると円の角になり、2 つの半径を設定すると橢円の角になります。
box-shadow	要素のフレームの周囲にシャドウ（影）効果を追加します。
color	要素のテキストやテキスト装飾における前景色の色の値を設定します。
display	要素をブロック要素とインライン要素のどちらとして扱うかを設定します。およびその子要素のために使用されるレイアウト、例えば フローレイアウト、グリッド、フレックスなどを設定します。
filter	ぼかしや色変化などのグラフィック効果を要素に適用します。フィルターは画像、背景、境界の描画を調整するためによく使われます。
font-family	選択した要素に対して、フォントファミリ名や総称ファミリ名の優先順位リストを指定します。明朝体、ゴシック体など、書体名を設定します。
font-size	フォントの大きさを定義します。
font-style	通常体（normal）、筆記体（italic）、斜体（oblique）のどれでスタイル付けるかを設定します。
font-weight	フォントの太さ（あるいは重み）を指定します。
height	要素の高さを指定します。
line-height	行ボックスの高さを設定します。これは主にテキストの行間を設定するために使用します。

<sup>\*5</sup> <https://developer.mozilla.org/ja/docs/Web/CSS/Reference>

要素	説明
<code>list-style-type</code>	リスト項目要素のマーカーを設定します(円、文字、独自のカウンタースタイルなど)。
<code>margin</code>	要素の全四辺のマージン領域を設定します。
<code>max-height</code>	要素の最大高を設定します。
<code>max-width</code>	要素の最大幅を設定します。
<code>object-fit</code>	<code>&lt;img&gt;</code> や <code>&lt;video&gt;</code> などの中身を、コンテナにどのようにはめ込むかを設定します。
<code>object-position</code>	<code>object-fit</code> プロパティと併用し、ボックス内における置換要素の配置を指定することができます。
<code>padding</code>	要素の全四辺のパディング領域を一度に設定します。
<code>position</code>	文書内で要素がどのように配置されるかを設定します。
<code>text-align</code>	ブロック要素または表セルボックスの水平方向の配置を設定します。
<code>text-decoration</code>	テキストの装飾的な線の表示を設定します。
<code>text-shadow</code>	テキストに影を追加します。文字列及びその装飾に適用される影のカンマで区切られたリストを受け付けます。
<code>vertical-align</code>	インライنبックス、インラインブロック、表セルボックスの垂直方向の配置を設定します。
<code>width</code>	要素の幅を設定します。
<code>z-index</code>	位置指定要素とその子孫要素、またはフレックスアイテムの z 順を定義します。より大きな <code>z-index</code> を持つ要素はより小さな要素の上に重なります。

### グリッドレイアウト関係のプロパティ

要素	説明
<code>grid-template-columns</code>	列の線名と列幅のサイズ変更機能を定義します。
<code>grid-template-rows</code>	行の線名と行高のサイズ変更機能を定義します。
<code>grid-auto-flow</code>	自動配置されたアイテムがどのようにグリッドに流れていくかを指定します。
<code>grid-column</code>	グリッド列の中におけるグリッドアイテムの寸法と位置を指定し、線、区間、なし(自動)をグリッド配置に適用されることで、グリッド領域の列の開始と終了の端を指定します。
<code>grid-row</code>	グリッド行の中におけるグリッドアイテムの寸法と位置を指定し、線、区間、なし(自動)をグリッド配置に適用されることで、グリッド領域の行の開始と終了の端を指定します。
<code>gap</code>	行や列の間のすき間(溝)を定義します。これは <code>row-gap</code> および <code>column-gap</code> の一括指定です。
<code>align-self</code>	グリッド領域内のアイテムの垂直方向の配置を指定します。
<code>justify-self</code>	グリッド領域内のアイテムの水平方向の配置を指定します。

## C.3

# JavaScript 簡易まとめ

JavaScript は、主にブラウザ上で動くプログラミング言語で、動的サイトの作成に用います。<sup>\*6</sup>

## ♣ コメント

JavaScript は、一行コメントと複数行コメントが用意されています。

コード例	説明
// xxx	一行コメント
/* xxx */	複数行コメント

## ♣ データ構造

変数とは、コンピュータプログラムのソースコードなどで、データを一時的に記憶しておくための領域に固有の名前を付けたもの。<sup>\*7</sup>

JavaScript では、定数宣言用の「`const`」と、変数宣言用の「`let`」が用意されています。

コード例	説明
<code>const x</code>	変数宣言。x に値の再代入はできない
<code>let x</code>	変数宣言。const と似ているが、x に値を再代入できる
<code>var x</code>	変数宣言。古い変数宣言方法（今は使わない）

## ♣ リテラル

リテラル (literal) とは、直値、直定数とも呼ばれ、コンピュータプログラムのソースコードなどの中に、特定のデータ型の値を直に記載したものである。また、そのように値をコードに書き入れるために定められている書式のことを行う。<sup>\*8</sup>

コード例	説明
<code>true</code> または <code>false</code>	真偽値
<code>123</code>	10進数の整数リテラル
<code>123n</code>	巨大な整数を表す BigInt リテラル
<code>1_2541_0000</code>	日本の人口など、大きな数は _ で区切ると読みやすくなる
<code>0b10</code>	2進数の整数リテラル
<code>0x30A2</code>	16進数の整数リテラル
<code>[x, y]</code>	x と y を初期値にもつ配列オブジェクトを作成
<code>{ k: v }</code>	プロパティ名が k、 プロパティの値が v のオブジェクト（連想配列）を作成

\*6 出典：JavaScript Primer 迷わないための入門書 (<https://jsprimer.net>) など

## ♣ 文字列

文字列とは、文字を並べたもの。コンピュータ上では、数値など他の形式のデータと区別して、文字の並びを表すデータを文字列という。<sup>\*9</sup>

コード例	説明
"xxx"	ダブルクオートの <b>文字列リテラル</b> 。
'xxx'	シングルクオートの <b>文字列リテラル</b> 。
`xxx`	テンプレート文字列リテラル。改行を含んだ入力が可能
`\${x}`	テンプレート文字列リテラル中の変数 x の値を展開する

## ♣ 制御構造

プログラムの流れを制御するための構文です。繰り返しのための「**for 文**」、条件分岐のための「**if 文**」などが用意されています。

例	説明
<code>while(x){}</code>	<b>while ループ</b> 。 x が true なら反復処理を行う。 繰り返し回数が不明な際に用いると効果的
<code>for(let x=0;x &lt; y ;x++){}</code>	<b>for ループ</b> 。 x < y が true なら反復処理を行う。 繰り返し回数が分かる時に使うと効果的
<code>for(const p in o){}</code>	<b>for...in ループ</b> 。 オブジェクト (o) のプロパティ (p) に対して反復処理を行う
<code>for(const x of iter){}</code>	<b>for...of ループ</b> 。 イテレータ (iter) の反復処理を行う
<code>if(x){/*A*\/}else{/*B*\/}</code>	<b>条件式</b> 。 x が true なら A の処理を、 それ以外なら B の処理を行う
<code>switch(x){case "A":{/*A*\/} "B":{/*B*\/}}</code>	<b>switch 文</b> 。 x が "A" なら A の処理を、 "B" なら B の処理を行う
<code>x ? A: B</code>	<b>条件 (三項) 演算子</b> 。 x が true なら A の処理を、 それ以外なら B の処理を行う
<code>break</code>	<b>break 文</b> 。 現在の反復処理を終了しループから抜け出す。
<code>continue</code>	<b>continue 文</b> 。 現在の反復処理を終了し次のループに行く。
<code>try{}catch(e){}finally{}</code>	<b>try...catch 構文</b>
<code>throw new Error("xxx")</code>	<b>throw 文</b>

## ♣ 演算子

演算子とは、数学やプログラミングなどで式を記述する際に用いる、演算内容を表す記号などのこと。様々な演算子が定義されており、これを組み合わせて式や命令文を構成する。<sup>\*10</sup>

以下の表は優先順位の最も高いもの(21)から最も低いもの(1)の順に並べられている。<sup>\*11</sup>

優先順位	演算子の種類	結合性	演算子	優先順位	演算子の種類	結合性	演算子
21	グループ化	n/a	( … )		小なり		… < …
	メンバーへのアクセス	左から右	… . …		小なりイコール		… <= …
	計算値によるメンバーへのアクセス	左から右	… [ … ]	12	大なり	左から右	… > …
20	new (引数リスト付き)	なし	new … ( … )		大なりイコール		… >= …
	関数呼び出し	左から右	… ( … )		in		… in …
	オプショナルチェイニング	左から右	?.		instanceof		… instanceof …
19	new (引数リストなし)	右から左	new …		等価		… == …
18	後置インクリメント	なし	… ++		不等価	左から右	… != …
	後置デクリメント		… --		厳密等価		… === …
	論理 NOT		! …		厳密不等価		… !== …
	ビットごとの NOT		~ …				
	単項 +		+ …	10	ビット単位 AND	左から右	… & …
	単項 -		- …	9	ビット単位 XOR	左から右	… ^ …
17	前置インクリメント	右から左	++ …	8	ビット単位 OR	左から右	…   …
	前置デクリメント		-- …	7	論理 AND	左から右	… && …
	typeof		typeof …	6	論理 OR	左から右	…    …
	void		void …	5	Null 合体	左から右	… ?? …
	delete		delete …	4	条件	右から左	… ? … : …
	await		await …				… = …
16	べき乗	右から左	… ** …				… += …
	乗算		… * …				… -= …
15	除算	左から右	… / …				… **= …
	剰余		… % …				… *= …
14	加算	左から右	… + …				… /= …
	減算		… - …				… %= …
	左ビットシフト		… << …				… <<= …
13	右ビットシフト	左から右	… >> …				… >>= …
	符号なし		… >>> …				… >>>= …
	右ビットシフト			3	代入	右から左	… &= …
							… ^= …
							…  = …
							… &&= …
							…   = …
							… ??= …
				2	yield	右から左	yield …
					yield*		yield* …
				1	カンマ / シーケンス	左から右	… , …

\*11 出典: 演算子の優先順位 ([https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Operators/Operator\\_Precendence](https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Operators/Operator_Precendence))

## ♣ データアクセス

プログラミング言語 Pascal の開発者 ニクラウス・ヴィルト氏による、「プログラミング」 = 「データ構造」 + 「アルゴリズム」は、広く知られています。

配列とオブジェクト(=連想配列)という主要なデータ構造にアクセスするために、次の構文が用意されています。

コード例	説明
array[0]	配列へのインデックスアクセス
obj["x"]	オブジェクトへのプロパティアクセス(ブラケット記法)
obj.x	オブジェクトへのプロパティアクセス(ドット記法)

## ♣ 関数宣言

関数とは、コンピュータプログラム上で定義されるサブルーチンの一種で、数学の関数のように与えられた値(引数)を元に何らかの計算や処理を行い、結果を呼び出し元に返すもののこと。<sup>\*12</sup>

サンプル	説明
function f(){}	関数宣言
const f = function(){};	関数式
const f = () => {};	Arrow Function の宣言
function f(x, y){}	関数における仮引数の宣言
function f(x = 1, y = 2){}	デフォルト引数、引数が渡されていない場合の初期値を指定する。
clasX{}	クラス宣言
const X = clasX{};	クラス式

## ♣ モジュール

大きなプログラムを作る際、小さな部品(モジュール)を組み合わせて作ると、管理しやすく、部品の再利用もできるので便利です。JavaScriptにも、特定のファイルで定義した関数を、他のファイルでも使えるようにする仕組みが用意されています。

コード	説明
import x from "./x.js"	デフォルトイントポート
import { x } from "./x.js"	名前付きイントポート
export default x	デフォルトエクスポート
export { x }	名前付きエクスポート

## ♣ その他

コード	説明
x;	文
{ }	ブロック文

## 【コラム】金の延棒クイズ 【解答】

最後までお読みくださり、ありがとうございます。金の延棒クイズの解答です。

2回鋏を入れて、金の延棒を1と2と4の大きさに分割します。

一日目のお支払いには、1の延棒を渡します。

二日目のお支払いには、2の延棒を渡して、先に渡した1の延棒は返してもらいます。

三日目のお支払いには、1の延棒も渡します。

四日目のお支払いには、大きな4の延棒を渡し、2と1の延棒は返してもらいます。

五日目のお支払いには、1の延棒も渡します。

六日目のお支払いには、2の延棒を渡して、先に渡した1の延棒は返してもらいます。

七日目のお支払いには、全ての延棒を渡します。

延棒の有無を0と1で表すと二進数と対応しています。

意外なところに潜む二進数。探してみてくださいね。

金の延棒	日当
<b>421</b>	
001	1
010	2
011	3
100	4
101	5
110	6
111	7

# 終わりに

本書は、開発環境を整えるところから始まり、HTML / CSS / JavaScript の基礎を学び、iPhone でも遊べるじゃんけんゲームを作り上げ、全世界に公開いたしました。

全部で、百行余りのじゃんけんプログラム、要所要所にコメントも付けていますので、今まで学んできた知識で読解できるはずです。ぜひ、遊んでみてください。自分で作ったプログラムの体験はいかがでしょうか。いろいろ創意工夫して、さまざまなアプリを作っていくうすですね。

「福祉」。「福」「祉<sup>\*13</sup>」どちらも「めぐみ、さいわい」という意味を持ちます。

「熱き心、<sup>たくま</sup>逞<sup>かいな</sup>しき腕、冷静な頭脳」

学生時代に言われた言葉ですが、福祉を生きる者は、人としての熱い思い、暖かい心を持ち、その上で、冷静な判断力を以て、力強く行動するのだと。

「工学」の「工」は、「天の<sup>ことわり</sup>理<sup>り</sup>」を、地に下ろす」意味です。

技術の産物としての社会ではなく、世界を<sup>かがや</sup>耀<sup>かがや</sup>かせるために技術を用いてください。技術に使われるのではなく、技術を使いこなし、人の道に役立てる人となってください。

令和の御世を生きる皆さんに素晴らしい人生を生き、素晴らしい日本を創ることを願って筆を置きます。

いやさか  
彌榮

---

\*13 「祉い」と書いて、「さいわい」と読みます。天からの恵みがその身に止まる意味です。

# ウェブアプリ簡単入門

じゃんけんゲームを創ろう

---

令和五年五月五日

著 者 アトリエ未来

発行者 早乙女 遙香

連絡先 contact@atelier-mirai.net

<https://atelier-mirai.net>

---

© 令和五年 アトリエ未来