

じゃんけんゲームを 創ろう

— 初めてのウェブプログラミング —

[著] アトリエ未来

「技術書典 11 新刊」
令和三年七月七日 ver 1.0.0

■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、TM、[®]、[©]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」に囲まれた生活を送るわたくしたち。多くの先人が築いた歴史の上に今日があります。

ページ数の関係で、二冊に分冊いたしました。姉妹巻となる「計算機の歴史と仕組み - 暮らしに生きるコンピュータ」では、算盤から iPhone まで、コンピュータの歴史と仕組みを学びます。また、アルゴリズムや学校教育におけるプログラミングについても簡単にご紹介しています。

本書では、HTML / CSS / JavaScript を学び、じゃんけんアプリを作り上げていきます。簡単なじゃんけんゲームですが iPhone でも楽しめ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つて、幸せな未来へと大きく羽ばたいてください。

対象読者

パソコンの操作ができる、初心者の方を想定しています。ウェブサイトを作ってみたい方や、プログラミングに興味があって、なにかアプリを作つてみたい人の最初の一歩になればと願っています。

また、誰かに教えようと思っている方にも、生徒の方にお渡しするテキストとして、お使い頂けるかと思います。

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 上で公開しています。以下の URL より、ダウンロードできます。

[https://github.com/Atelier-Mirai/janken^{*1}](https://github.com/Atelier-Mirai/janken)

また、以下の URL で公開しておりますので、お楽しみください。

[https://atelier-mirai-janken.netlify.app^{*2}](https://atelier-mirai-janken.netlify.app)

謝辞

Re:VIEW Starter^{*3}を用いて、綺麗に製本することができました。作者の kauplan 氏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、プログラミングの個人指導やウェブ開発を行う「アトリエ未来^{*4}」を創業。HTML 講座や Ruby 講座、IT パスポート講座等を開催している。

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を歩くことや、たくさんの花に囲まれた日々を愛している。

^{*1} <https://github.com/Atelier-Mirai/janken>

^{*2} <https://atelier-mirai-janken.netlify.app/>

^{*3} <https://kauplan.org/reviewstarter/>

^{*4} <https://atelier-mirai.net/>

目次

始めに	i
対象読者	i
プログラムのダウンロード	i
謝辞	ii
著者紹介	ii
第 1 章 ウェブの歴史と技術	1
1.1 ウェブサイトの発祥	1
1.2 ウェブサーバとブラウザ	2
1.3 HTML とは	3
1.4 CSS とは	4
♣ 文書構造とスタイル指定とを分離する	5
1.5 JavaScript とは	6
♣ 主な特徴	6
♣ ブラウザでの使われ方	6
♣ 他の実行環境	6
♣ 歴史	7
第 2 章 開発環境の準備	9
2.1 21世紀の高性能エディタ Atom	10
♣ お勧めプラグイン	10
2.2 セキュリティ重視のブラウザ Firefox	14
2.3 基本的な作り方	15
♣ 手順のご案内	16
第 3 章 初めての HTML	25
3.1 初めての HTML 【解説】	27
3.2 HTML の基本ルール	28

3.3	じゃんけんゲームの土台を作る	32
第4章	初めての JavaScript	37
4.1	初めての JavaScript	38
4.2	初めての JavaScript 解説	39
	♣ コメント	39
	♣ 厳格モード	39
	♣ alert 関数	39
4.3	変数	40
4.4	ウェブブラウザと DOM	41
4.5	イベントリスナー	42
4.6	関数(function)	43
4.7	開始ボタンを押すと応答するプログラム	44
4.8	繰り返し処理	46
4.9	条件分岐	48
4.10	乱数を利用する	52
	[コラム] 亂数関数を自作する	53
4.11	入れ子になった if 文	53
4.12	定数	55
	♣ 関数化とリファクタリング	57
	♣ じゃんけんのアルゴリズム	58
第5章	初めての CSS	63
5.1	ユーザインターフェース(UI)とは	63
5.2	UI/UX を考慮した HTML	65
	♣ レスポンシブ対応	68
	♣ スタイルシートの読み込み	68
	♣ 画像ファイルの指定	68
	♣ id 属性	69
	♣ class 属性	69
	♣ button 要素	69
	♣ a 要素と URL エンコーディング	70

5.3	CSS ファイルの作成	70
5.4	CSS の基本	76
5.5	CSS セレクタ	79
5.6	CSS グリッドレイアウトとは	82
	♣ グリッドとは何か?	82
	♣ アイテムの配置	83
	♣ グリッドコンテナの作成	83
	♣ グリッド線を使って要素を配置する	83
5.7	色について	84
	♣ CSS での色の指定をする	85
5.8	レスポンシブデザイン	86
第 6 章	じゃんけんゲームの完成	89
6.1	プレイヤーの手を取得する	90
6.2	コンピュータの手を表示する	91
	♣ 配列	92
	♣ 配列内の要素の指定法	93
6.3	アニメーション機能を実装する	94
	♣ 再帰関数	97
	♣ アニメーションの開始と終了処理	97
6.4	名前重要	97
6.5	勝敗更新機能の実装	99
6.6	じゃんけんプログラム完成	101
6.7	ウェブサイトへの公開	104
付録 A	珠玉の名著のご紹介	107
A.1	HTML / CSS / JavaScript を学ぶ	107
	♣ CSS グリッドで作る HTML5 & CSS3 レッスンブック . .	107
	♣ JavaScript Primer 迷わないための入門書	108
	♣ 動く Web デザインアイディア帳	108
A.2	コンピュータサイエンスの基礎を学ぶ	108
	♣ みんなのコンピュータサイエンス	109

♣ プログラマの数学	110
♣ 教養としてのコンピューターサイエンス講義	110
♣ キタミ式イラスト IT 塾 IT パスポート	111
A.3 プログラマとして上達したい方のために	111
♣ 新装版 達人プログラマー 職人から名匠への道	111
♣ コーディングを支える技術	112
A.4 アルゴリズムを学ぶ	112
♣ アルゴリズム図鑑 絵で見てわかる 26 のアルゴリズム	113
♣ C 言語による標準アルゴリズム事典	113
A.5 Ruby を学ぶ	113
♣ 初めてのプログラミング 第 2 版	114
♣ たのしい Ruby 第 6 版	114
♣ プロを目指す人のための Ruby 入門	115
♣ Ruby によるデザインパターン	115
付録 B 付録: 参考リンク集	117
B.1 Mozilla 公式チュートリアル	117
B.2 タイピング	117
B.3 プログラミング練習	118
B.4 技術用語	118
B.5 写真素材	119
B.6 ウェブデザイン	119
B.7 開発マシン	119
B.8 エディタ	120
B.9 ブラウザ	120
B.10 英語翻訳	120
B.11 ウェブサイトの公開	121
B.12 情報技術を夢の架け橋に アトリエ未来	121
終わりに	123

第 1 章

ウェブの歴史と技術

今ではすっかり定着したウェブサイト。その歴史をみていきます。そしてウェブサーバとブラウザとの関係、ウェブサイトを作成する際に用いられる、HTML / CSS / JavaScript の概要をご紹介します。

1.1 ウェブサイトの発祥

HTML でウェブページを作成し、情報を発信するシステムは、平成 2 年(1989 年)に CERN(歐州原子核研究機構) のティム・バーナーズ・リー氏によって提案されました。「どのようなコンピュータからでも情報を閲覧し、共用できるようにする」ことが目的で、正式には WWW(World Wide Web) と呼ばれます。

WWW は、CERN を中心としたケンキュ期間や大学を中心に開発が進み、平成 6 年(1993 年)にオープンな形となりました。同時に、NCSA によってリリースされた Mosaic というブラウザによって多くの研究者が使い始めます。やがて、Netscape(Firefox の前身) というブラウザの誕生と Windows 95 のリリースにより、誰もが使える環境が整い、爆発的に普及していくこととなり

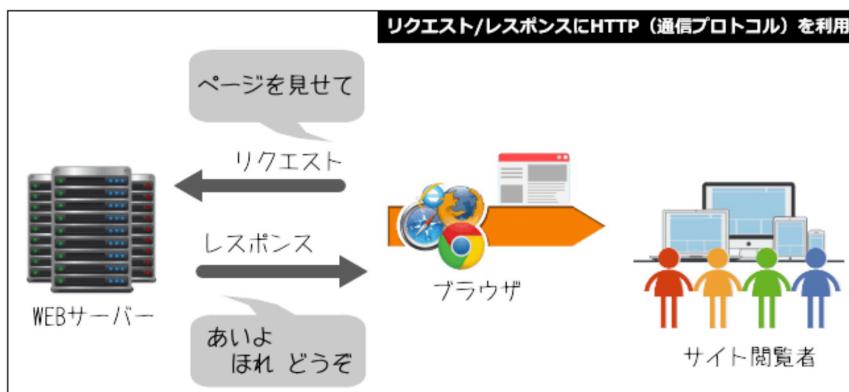
ました。 *1

1.2 ウェブサーバとブラウザ

ウェブサーバとは、利用者側のコンピュータに対し、ネットワークを通じて情報や機能を提供するコンピュータおよびソフトウェアのことです。

ウェブサーバは、利用者のコンピュータから操作されるブラウザからの要求に応え、自身の管理するデータを送信します。より具体的には HTML ファイルや画像ファイルなどウェブページを構成するファイルを送信します。

このやり取りには HTTP (HyperText Transfer Protocol) と呼ばれる通信規約（プロトコル）が用いられるので、わたくしたちがウェブサイトを閲覧する際には、「`http://～`」あるいは「`https://～`」から始まる URL を、ブラウザのアドレスバーに入力します。



▲図 1.1: ウェブサーバとブラウザ

*2

*1 出典: CSS グリッドで作る HTML5&CSS3 レッサンブック

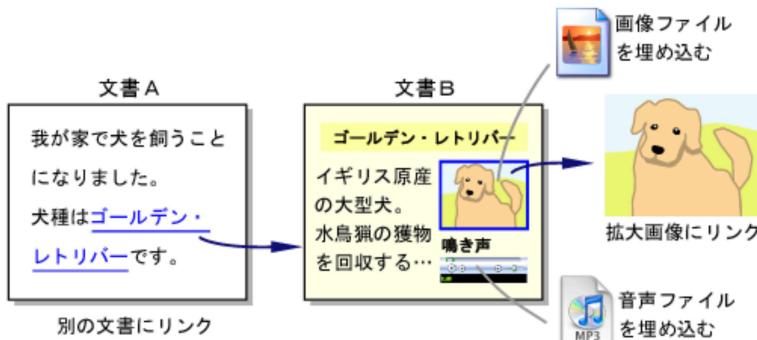
*2 画像出典: <https://www.nkshopping.biz/index.php?レスポンスとは。説明文は、IT用語辞典をもとに改変>

1.3 HTML とは

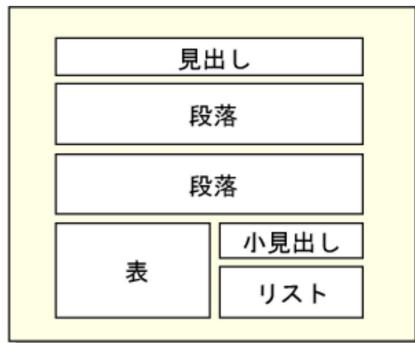
HTML(HyperText Markup Language) は、ウェブページを作成するため開発された言語です。現在、インターネット上で公開されているウェブページのほとんどは、HTML で作成されています。

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。ハイパーテキスト (HyperText) とは、ハイパーリンクを埋め込むことができる高機能な文書です。ハイパーリンクというのは、ウェブページで下線の付いたテキストなどをクリックすると別ページへ移動する、あのリンクのことです。

ハイパーテキストでは、ウェブページから別のウェブページにリンクを貼つたり、ウェブページ内に画像・動画・音声などのデータファイルをリンクで埋め込むことができます。HTML には、このハイパーリンク機能で関連する情報同士を結びつけて、情報を整理するという特徴があります。



また、目印をつける (Markup) というのは、文書の各部分が、どのような役割を持っているのかを示すということです。例えば、見出し・段落・表・リストなど、文書の中で各部分が果たしている役割が分かるように目印をつけます。こうした見出し・段落・表・リストなどの文書内の各部分を要素と呼びます。



▲図 1.2: 文書の各部分を要素に分ける

文書内の各部分に目印をつけて、その部分がどんな要素なのかを明確にすることで、コンピュータがその文書の構造を理解できるようになります。具体的には、検索エンジンがウェブページの構造を把握して解析したり、ブラウザがウェブページ内の各要素の意味を理解して閲覧しやすいように表示することなどが可能になります。

このようにコンピュータに理解できるように文書の構造を定義することこそが、HTML の最も重要な役割と言えるでしょう。この際、目印をつけるための記号として使用されるのが HTML タグです。^{*3}

1.4 CSS とは

CSS(Cascading Style Sheet) とは、Web ページの要素の配置や見栄えなどを記述するための言語です。

指定できる項目は、要素の大きさや配置、要素間の位置関係や空白、要素の境界線や余白、要素間の間の空白や周囲の余白、文字の大きさや文字や行の間隔、書体（フォント）の種類や変形（太字や斜体など）、箇条書きの表示書式、背景色や背景画像など多岐に渡ります。

HTML 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

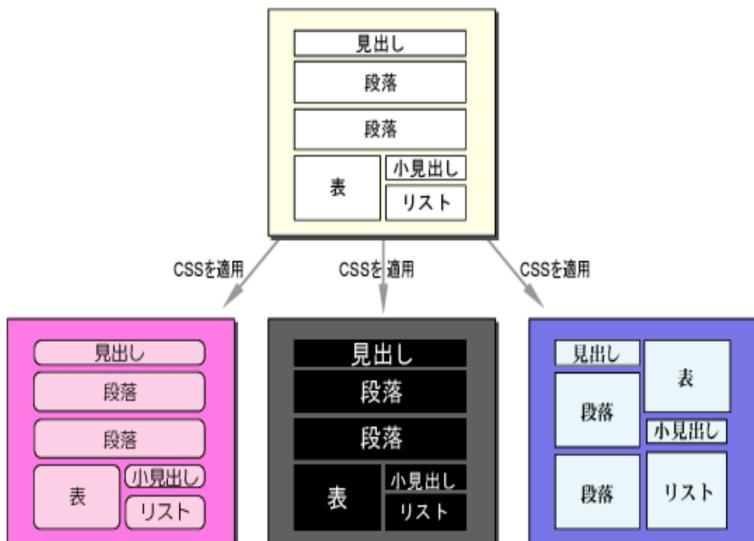
^{*3} 出典：HTML クイックリファレンス

指定されたものが子要素、孫要素に引き継がれ、子要素で指定されたものが追加されていきます。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”（カスケード）になぞらえて、CSS(Cascading Style Sheet) という名称になりました。^{*4}

♣ 文書構造とスタイル指定とを分離する

HTML に文章構造を記述し、CSS に見栄えに関する記述を行うと、対象機器（パソコンや携帯電話など）に応じて CSS を切り替えることで、それぞれに適した表示を行えるようになります。

また、共通する見栄えの部分を他のページにも適用することで、効率的に統一感のあるウェブサイトを作成できるようになります。^{*5}



▲図 1.3: CSS で見た目を切り替える

^{*4} 出典：IT 用語辞典

^{*5} 出典：HTML クイックリファレンス

1.5

JavaScript とは

JavaScript は主にウェブブラウザの中で動くプログラミング言語です。その特徴などをご紹介します。

♣ 主な特徴

C 言語や Java に似た記法や文法を採用した手続き型の言語です。オブジェクト指向にも対応しています、多くの言語で一般的なクラスを用いる方式ではなく、既存オブジェクトを複製し、機能を追加していくプロトタイプベースと呼ばれる手法を採用しています。関数を変数のように（第一級オブジェクトとして）扱ったり、関数を引数に取る高階関数を定義できるなど、関数型プログラミング言語の仕様も取り込んでいます。

♣ ブラウザでの使われ方

JavaScript は、HTML ファイルから JavaScript が書かれたファイルを読み込む形で良く使われます。ブラウザによってページが表示される際に、ページ内の写真などの要素に動きや効果を加えたり、閲覧者の操作に反応して何らかの処理を行ったりする為に用います。

例えば、ウェブサイトで何か操作をしたら表示が書き換わったり、ウェブサイトの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。

♣ 他の実行環境

当初は、ブラウザ上で実行されるプログラミング言語でしたが、Node.js というサーバ側のアプリケーションを作る仕組みでも利用されています。また、デスクトップアプリやスマートフォンアプリ、IoT (Internet of Things) デバイスでも JavaScript を使って動かせるものが現れるなど、幅広い環境で動いているプログラミング言語となっています。

♣ 歴史

JavaScript は 1995 年にネットスケープ・コミュニケーションズ (Netscape Communications) 社（当時）のブランダン・アイク (Brendan Eich) 氏によって開発されました。当時最も人気の高いウェブブラウザだった Netscape Navigator 2.0 に実装されたのが、その始まりです。当初は「LiveScript」という名称でしたが、同社が Java 言語の開発元のサン・マイクロシステムズ (Sun Microsystems) 社と提携していたことから、JavaScript に改称されました。

名称に Java と付いてはいますが、記法や予約語などの一部が共通するのみで、Java 言語との直接的な繋がりや互換性はありません。 *6

*6 出典：IT 用語辞典, JavaScript Primer 迷わないとめの入門書

第 2 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「Atom」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の環境整備を行っていきましょう。整える道具は二つ、「エディタ」と「ブラウザ」です。

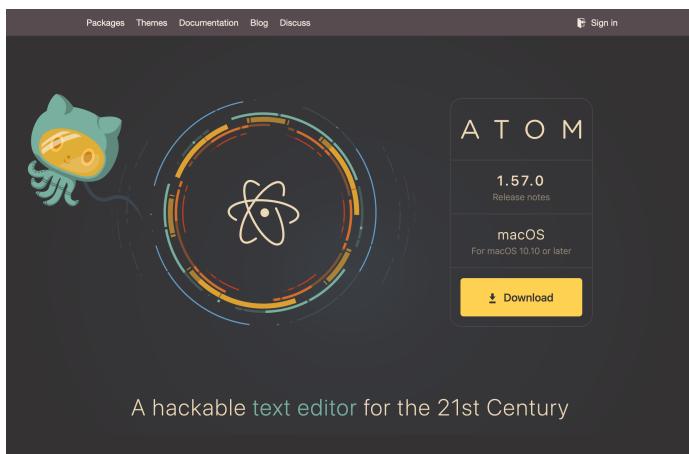
プログラミングを行うために必要となるエディタを準備しましょう。エディタとは、プログラミングが快適に行えるよう様々な支援機能を備えたコーディングのためのソフトウェアです。さまざまなエディタが提供されていますが、ここでは、Atom をご紹介いたします。

「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。じゃんけんゲームは、ウェブアプリとして開発していきますので、コーディング結果を確認するためにブラウザが必要です。さまざまなブラウザが提供されていますが、ここでは、Firefox をご紹介いたします。

2.1

21世紀の高性能エディタ Atom

Atomは、公式サイト^{*1}より、ダウンロードできます。黄色の「Download」ボタンを押すとダウンロードが始まります。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。



▲図2.1: Atom公式サイト

♣ お勧めプラグイン

Atomはそのままでも十分高機能に使えるGitHub謹製の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。^{*2}

プラグインのことを、Atom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。

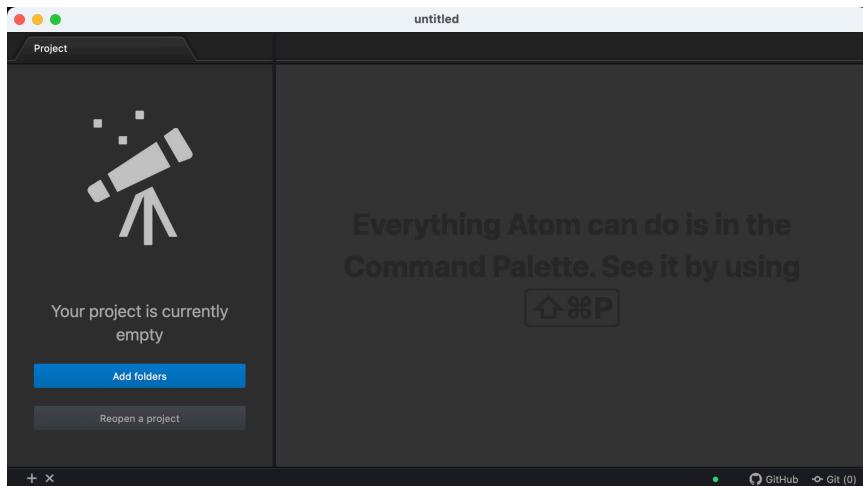
^{*1} <https://atom.io>

^{*2} 出典：IT用語辞典

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

それでは、インストールしていきましょう。

1. Atom を起動します。



▲図 2.2: 起動直後

^{*3} <https://atom.io/packages/japanese-menu>

^{*4} <https://atom.io/packages/color-picker>

^{*5} <https://atom.io/packages/pigments>

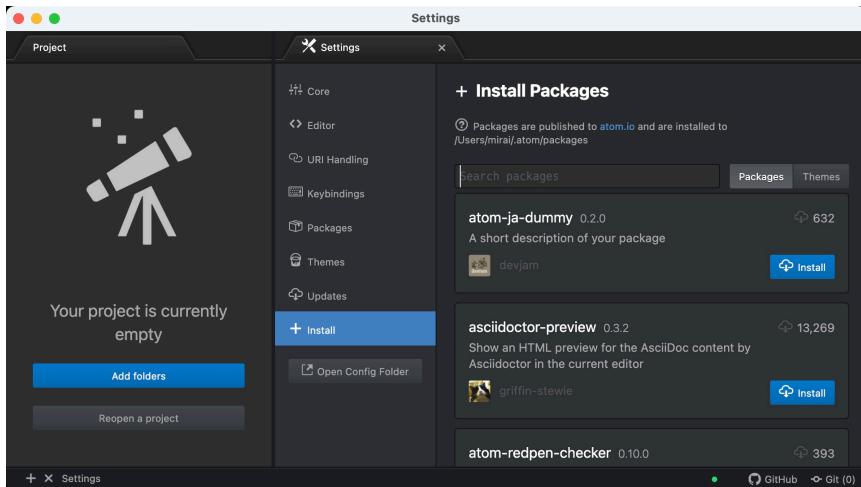
^{*6} <https://atom.io/packages/file-icons>

^{*7} <https://atom.io/packages/highlight-column>

^{*8} <https://atom.io/packages/selection-highlight>

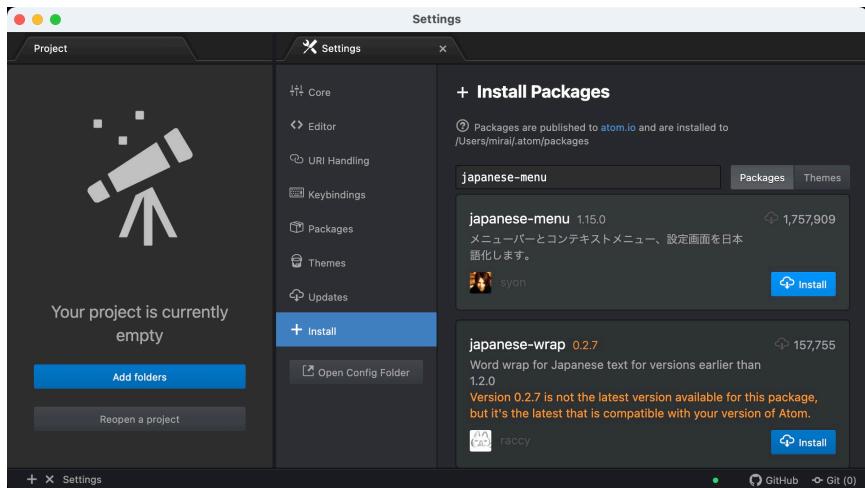
^{*9} <https://atom.io/packages/tabcnine>

- 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。



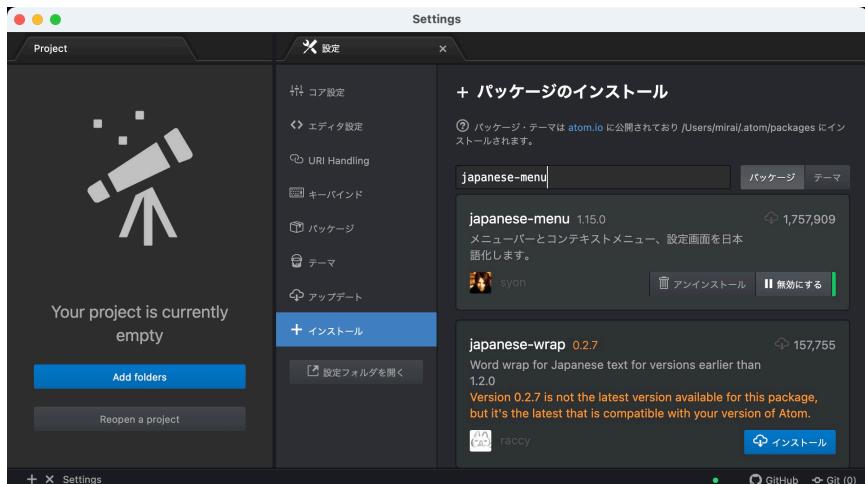
▲図 2.3: 設定画面

- 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を入力します。ここでは、`japanease-menu` と入力します。`japanese-menu` というパッケージが表示されますので、「Install」ボタンを押します。



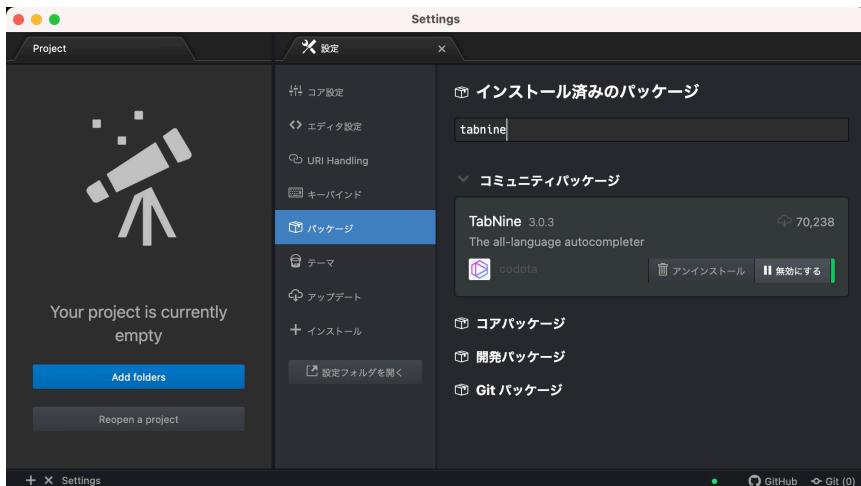
▲図 2.4: パッケージ名を入力

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。



▲図 2.5: インストール完了

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。



▲図 2.6: パッケージの削除や無効化の方法

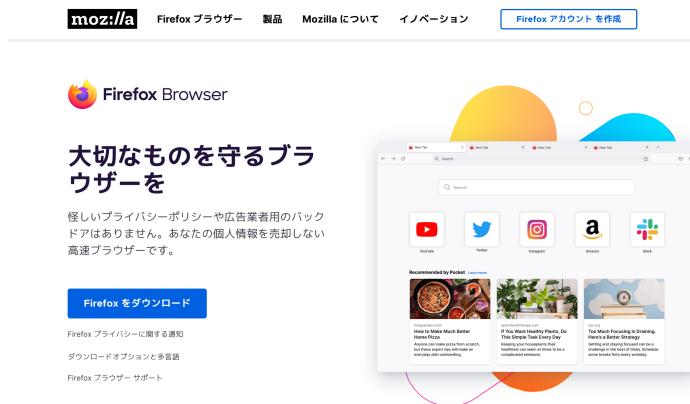
この他にも、いくつかお勧めのパッケージを見繕いましたので、Rails での Web アプリ開発に愛用している Atom プラグイン^{*10} にご紹介しています。いろいろなパッケージをインストールして自分の使いやすいエディタに育てていって下さい。

2.2 セキュリティ重視のブラウザ Firefox

Firefox は、Mosaic, Netscape の血を受け継ぐ、ブラウザです。CSS グリッ

*10 https://zenn.dev/atelier_mirai/articles/c3ed79af5ba395

ドの確認がし易く、丁寧なコメントの開発者ツールが特徴です。Firefox は、公式サイト^{*11}より、ダウンロードできます。青色の「Firefox をダウンロード」ボタンを押すとダウンロードが始まります。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。



▲図 2.7: Firefox 公式サイト

2.3

基本的な作り方

Atom と Firefox がインストールできましたので、プログラムの基本的な作り方をご案内します。

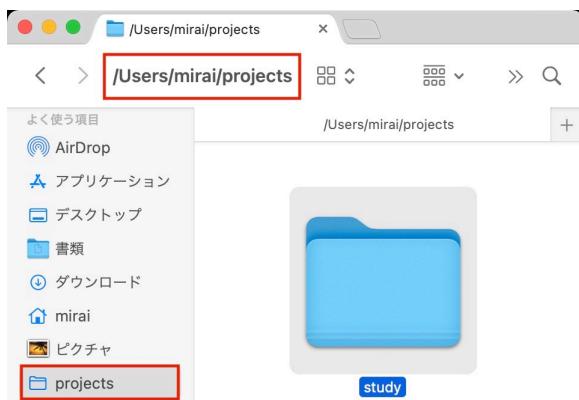
0. 左側に Atom を配置、右側に Firefox を配置します。
1. Atom で、プログラムをいろいろ書いていきます。
2. Firefox で、作ったプログラムの表示や動作を確認します。
3. 出来上がりが良ければ、プログラムを公開して完了です。
もう少し改良したければ、1. に戻ります。

^{*11} <https://www.mozilla.org/ja/firefox/new/>

♣ 手順のご案内

作業ディレクトリの作成方法や、Atom や Firefox の簡単な使い方のご紹介です。（Mac 利用者向けです。Windows の方は適宜読み替えてください。）
より、詳細な解説は テキストエディタ Atom 入門^{*12} がお勧めです。

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、ご自身で作成するいろいろなプロジェクト（案件）を格納すると良いと思います。良く使うことになるので、 Finder のサイドバーに登録しておくと便利です。projects ディレクトリが作成できたら、その下に今回の学習で用いる study ディレクトリを作成します。



▲図 2.8: study ディレクトリの作成

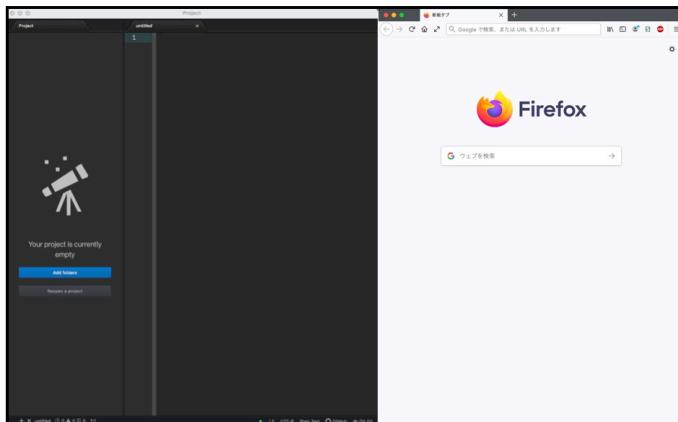
1. Atom のアイコンと Firefox のアイコンです。
良く使うのでドックに登録しておくと便利です。

^{*12} <https://books.oiax.jp/items/atom>



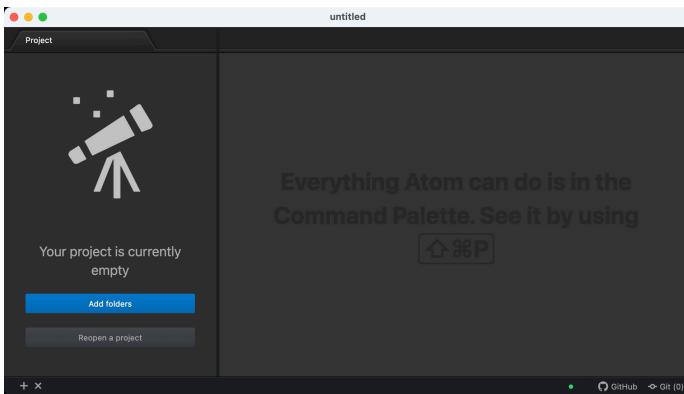
▲図 2.9: Atom(左)と Firefox(右)のアイコン

2. Atom と Firefox を左右に配置します。



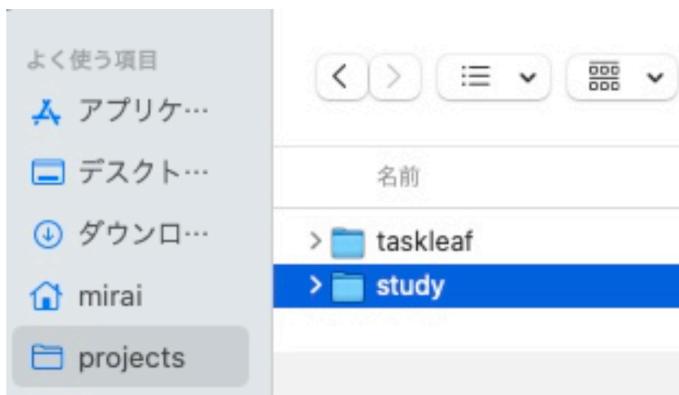
▲図 2.10: Atom(左)と Firefox(右)を配置

3. Atom 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、Atom の画面左側に配置されている「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るので、とても便利です。プロジェクトフォルダを追加するために、左側の青色の「Add folders」ボタンを押します。



▲図 2.11: 左側の青色の「Add folders」ボタンを押す

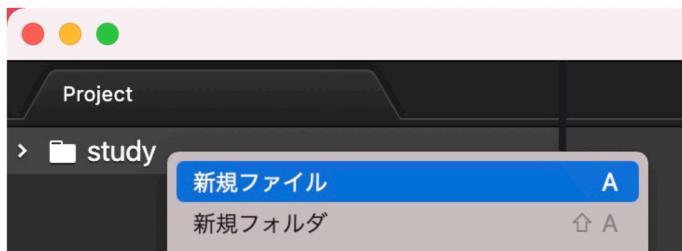
4. Finder が開くので、サイドバーから projects を選びます。projects ディレクトリ内には、taskleaf と study の二つのディレクトリがありますが、ここでは study ディレクトリを開きます。



▲図 2.12: projects 内の study を選ぶ

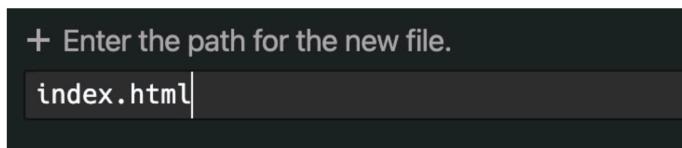
5. 左側の「ツリービュー」に study ディレクトリが表示されました。この study ディレクトリ内にいろいろなファイルを納めていき、最終的に一本の作品を創り上げていきます。今は何もないディレクトリな

ので、新しく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ます。コンテンツメニューが表示されるので、「新規ファイル」をクリックします。



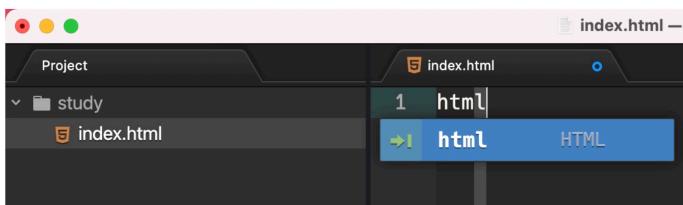
▲図 2.13: `study` を右クリックし 新規ファイルをクリックする

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。



▲図 2.14: 新しいファイル名として `index.html` と入力する

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。早速これをクリックします。すると右側の編集領域（「ペイン」と呼ばれています）に、今作成した `index.html` が表示されます。当然、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。Atom 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。



▲図 2.15: 「html」と入力して「入力支援機能」を活用する

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>
```

A screenshot of the Atom code editor showing a generated HTML template. The code is displayed in a dark-themed editor with syntax highlighting. The template includes a DOCTYPE declaration, an HTML element with attributes 'lang="en"' and 'dir="ltr"', a head section containing a meta element with 'charset="utf-8"', a title element, a body section, and a closing HTML element.

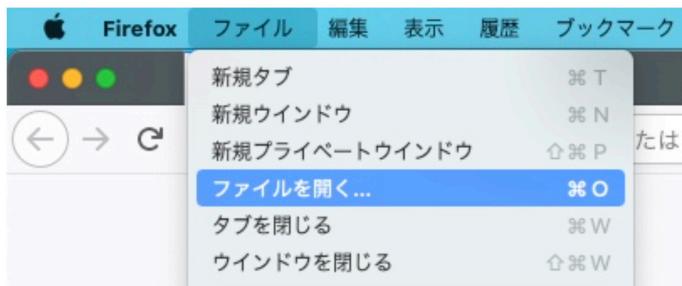
▲図 2.16: 入力支援機能により入力された雛形

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。それでは今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。Atom 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
      成ります。
    </p>
  </body>
</html>
```

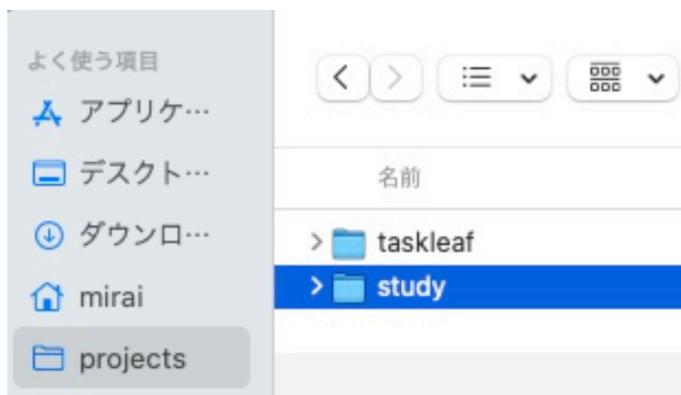
▲図 2.17: 適宜編集して完成した HTML

10. HTML を入力して、`index.html` ファイルが完成しました。Atom の「ファイル」メニューから「保存」をクリックして、保存しましょう (`Command + S`) を押しても保存することができます)。保存ができたら、どのように表示されるのか、Firefox を使って確認してみましょう。Firefox の「ファイル」メニューから、「ファイルを開く」をクリックします (`Command + O`) を押しても開くことができます)。



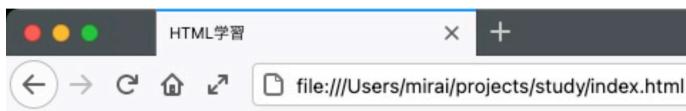
▲図 2.18: ファイルメニューから「ファイルを開く」をクリック

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。



▲図 2.19: projects 内の study を選ぶ

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

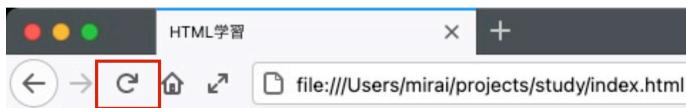


初めてのHTML

HTMLを学習して、素敵なサイトを作れるように成ります。

▲図 2.20: index.html を Firefox で閲覧する

13. 綺麗に出来上がっていたら完成です。もしもう少し改善したい点があるなら、Atomに戻って、`index.html`を編集します。編集が終わったら、`Command + S`を押して保存しましょう。保存が終わったら、編集結果をFirefoxで見てみましょう。「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。



▲図 2.21: 再読み込みボタン

第3章

初めてのHTML

じゃんけんゲームの土台となるHTMLの基本文法をご紹介します。HTMLの文法はとても簡潔で、理解しやすいものとなっています。何事も基礎がとても大切です。しっかり習得なさってください。

前章で、テキストエディタAtomの使い方をご説明する過程で、以下のHTMLを作成いたしました。15行の短いコードですが、HTMLの基本が詰まっていますので、一つずつ解説していきます。HTMLに関しては、(準)公式サイトMDNによる、HTMLの基本^{*1}が分かりやすいです。また巻末に挙げた参考書籍「CSSグリッドで作るHTML5&CSS3レッスンブック」も大変良い本ですので、一読をお勧めいたします。この本でのHTML/CSSの記述も多くの部分を負っています。

▼index.html

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

^{*1} https://developer.mozilla.org/ja/docs/Learn/Getting_started_with_the_web/HTML_basics

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

3.1 初めての HTML 【解説】

```
<!DOCTYPE html> DOCTYPE宣言です。  
<html> HTML5で書かれていることを明示します。  
  <head> Webページに関する設定の記述場所  
    <meta charset="utf-8">  
    <title>HTML学習</title>  
  </head>  
  <body> コンテンツ(内容・出し物)の記述場所  
    <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なります。  
    </p>  
  </body>  
</html>
```

- 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。

- 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

ファイル名を、index.html としました。index は、**ファイル名**と呼ばれます。後述する**拡張子**と区別する意図で、**主ファイル名**と呼ばれることもあります。

index は、「見出し」「索引」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**と呼ばれており、ファイルの種類が HTML であることを示します。拡張子が .html であるファイルを開くと、OS により、良くブラウザが開くように設定されています。

文書型宣言

全ての 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 で書かれている旨をブラウザに伝えることです。今日ほとんどのウェブサイトが HTML5 に従って書かれていますので、意味のない記述ですが、過去の歴史的な経緯により、先頭に書く必要があります。

<html>, <head>, <body>

<html> 全体は、大きく二つの部分 <head> と <body> に分かれます。

<head> は、Web ページに関する設定の記述場所です。

- 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

を記述しています。

<body> は、コンテンツ (内容・出し物) の記述場所です。

- 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。

3.2 HTML の基本ルール

要素とは

HTML では、コンテンツ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、中身がなんであるかを明確にします。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。

この例ですと、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。

要素名(h1)

<h1>初めてのHTML</h1>

開始タグ コンテンツ(内容) 終了タグ

要素

▲図 3.1: HTML の要素

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。

左は正しい HTML で、右は誤った HTML です。タグの対応状態を注意深く観察してみてください。

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

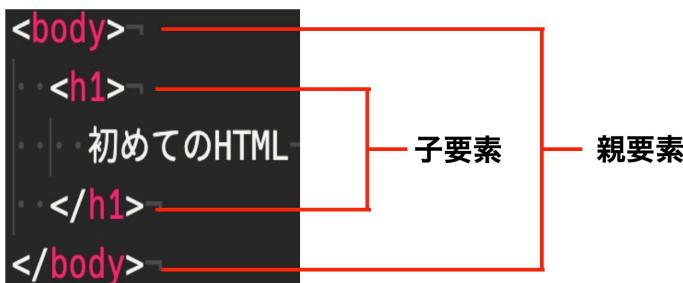
```
<body>
|· · <h1>
|· · · 初めてのHTML
|· · </body>
</h1>
```

▲図3.2: 正しいHTML(左)と誤ったHTML(右)

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。



▲図3.3: 階層構造になったHTML

コメント(覚書・説明書き)

コメント(覚書・説明書き)と呼ばれる構文があります。`<!-- -->`の間にコメントを書くことができます。ソースコードを分かり易くするための覚書、説明書きとして活用しましょう。

Atom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、もう一度押すとコメントを解除できるようになっています。

```
<!-- コメント(覚書・説明書き) -->
```

▲図 3.4: コメント(覚書・説明書き)

字下げ(インデント)

字下げ(インデント)はとても大切です。HTMLでは、改行や空白は無視されますが、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)しましょう。

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

▲図 3.5: 好ましい字下げ その1

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

▲図 3.6: 好ましい字下げ その2

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

▲図3.7: 好ましい字下げ その3

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

▲図3.8: 不適切な字下げ

3.3

じゃんけんゲームの土台を作る

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。じゃんけん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      </head>
7
8      <body>
9          <header>
10             <h1>じゃんけんゲーム</h1>
11         </header>
12
13         <main>
14             <p>
15                 みんな知っているじゃんけん。  

16                 グーは 0, チョキは 1, パーは 2 を入れてね。
17             </p>
18
19             <input type="number" id="janken_number" value="0">
20             <button id="play">開始</button>
21         </main>
22
23         <footer>
24             <p>
25                 情報技術を夢の架け橋に  

26                 &copy; アトリエ未来
27             </p>
28         </footer>
29
30         <script src="janken.js"></script>
31     </body>
32 </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 --> 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         <!-- このウェブページのタイトルです -->
8         <!-- ブラウザのタブに表示されます -->
9         <title>じゃんけんゲーム</title>
10    </head>
11
12    <!-- body タグです -->
13    <!-- 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 です -->
17        <!-- サイトの付帯情報やサイト名などの表示に用います -->
18        <header>
19            <!-- h1 タグ です -->
20            <!-- 大見出しを表すタグです -->
21            <!-- 他に 中見出しを表す h2 タグや
22                小見出しを表す h3 タグなど
23                h6 タグまであります -->
24            <h1>じゃんけんゲーム</h1>
25        </header>
26
27        <!-- main タグです -->
28        <!-- このページの主題(main)となる
29                    コンテンツ（内容・出し物）を記述します-->
30        <main>
31            <!-- p は 段落(paragraph) を表す際に用いるタグです -->
32            <p>
33                <!-- br タグを入れると、途中で、改行できます -->
34                みんな知っているじゃんけん。<br>
35                ゲーは 0, チョキは 1, パー は 2 を入れてね。
36            </p>
37
38            <!-- input タグです -->
39            <!-- 利用者が、じゃんけんの手を入力する枠を用意します -->
40            <!-- type="number" と書いて数字用の入力枠にします -->
41            <!-- 後ほど、JavaScript から 扱いやすいよう、
42                この入力枠に、id="player_hand" と ID を付与します
```

```

43      ID を付与することで、多数のHTML要素の中から
44      「一意」に特定の要素を取得することができます -->
45      <!-- value="0" と書き 枠の中に「0」を表示します
46      つまりプレーヤーの手は ゲー(0) と言うことです -->
47      <input type="number" id="player_hand_type" value="0">
48
49      <!-- button タグです
50      画面に「開始」ボタンが表示されます -->
51      <!-- このボタンにも、id="play" と ID を付与します
52      ID を付与することで、多数のHTML要素の中から
53      「一意」に特定の要素を取得することができます -->
54      <button id="play">開始</button>
55  </main>
56
57      <!-- footer タグ です -->
58      <!-- サイトの付帯情報や著作権表示などに用います -->
59  <footer>
60      <p>
61          情報技術を夢の架け橋に<br>
62          <!-- &copy; は「実体参照」と呼ばれる記法です -->
63          <!-- 「こぴーらいと」とタイプして 変換すると
64          「©」となりますが「©」の入力が難しいときや
65          HTMLのタグと紛らわしい時には
66          「実体参照」が便利です -->
67          &copy; アトリエ未来
68      </p>
69  </footer>
70
71      <!-- JavaScriptの読み込み -->
72      <script src="janken.js"></script>
73  </body>
74 </html>
```

紙幅の関係上、細かい説明は省かせていただきますが、大まかにそれぞれのコードが果たしている役割は理解できるのではないかでしょうか。より詳しく知りたい方には、「HTML 要素リファレンス^{*2}」が、(準)公式サイトとして、とても良くまとめていますので、是非ご覧になってください。

作成した index.html をブラウザで開くと、次のようになります。

^{*2} <https://developer.mozilla.org/ja/docs/Web/HTML/Element>



▲図3.9: 初めてのじゃんけんゲーム

絵も色もなくて少し寂しいですが、最後には綺麗に仕上げていきますので、お楽しみに。

第 4 章

初めての JavaScript

JavaScript の文法を学びつつ、少しずつコードを追加していくことで、よりじゃんけんゲームらしく、作り上げていきます。

前章では、HTML の基本を学びました。そして、JavaScript は、HTML に組み込まれて使われることが多いプログラミング言語です。

先に書いた HTML の一番最後に、次のような記述があったことを思い出してください。

▼ JavaScript の読み込み

```
71 <!-- JavaScript の読み込み -->
72 <script src="janken.js"></script>
```

これを書くことにより、`janken.js` という JavaScript のプログラムを読み込んで実行することができるようになります。

HTML, JavaScript と二つの言語が登場しました。後ほど CSS という言語も登場します。

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

HTML, JavaScript, CSS と三つの言語が一緒になって、一つのウェブアプ

りが出来上がります。index.html, janken.js, janken.css と三つのファイル、それぞれにコードを書いていきますので、少し大変かもしれませんね、進んでいきましょう。

4.1 初めての JavaScript

それでは、初めての JavaScript のプログラムを書き始めていきましょう。
janken.js というファイルを作り、次のように書きましょう。 *1

▼ janken.js

```
1 // 潜在的なバグを減らす。
2 'use strict';
3
4 // ジャンケンの勝ち負けの結果を表示する
5 alert("あなたの勝ちです!");
```

プログラムが書き終わったら、保存しましょう。そしてブラウザを再読み込みすると、次のようになります。



「あなたの勝ちです!」と、メッセージが表示されました。おめでとうございます。初めての JavaScript は、これで完成です!!

*1 出典：IT用語辞典

4.2 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。

♣ 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。大まかな処理の塊ごとに、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

♣ 厳格モード

JavaScript には、歴史的な経緯により、陥りやすい意図しない動作が多々あります。

```
1 // 潜在的なバグを減らす。
2   'use strict';
```

'use strict'; と、プログラムの先頭に書くことで、厳格 (strict) モードになりますので、予期しない不具合を防ぐ効果があります。

♣ alert 関数

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

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。)

2 / コメント */ というコメントの書き方もあります。

メッセージを表示する機能はあると便利ですので、JavaScriptに標準で用意されており、`alert`関数といいます。

`alert`関数を使うには、`alert('表示させたいメッセージ');`と書くことで使えます。

HTMLを書き、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。

どのようにすれば、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ね。

そのために、プログラミングに関する基礎知識を準備していきましょう。

4.3

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。 *3

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。変数に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。

*3 出典：IT用語辞典

JavaScript には「これは変数です」という宣言をするキーワードとして、`const` `let` `var` の 3 つがあります。

`const` は、「定数」^{*4} を宣言する時に使います。一度決めたらその値を変更することはできません。

`let` は、「変数」を宣言する時に使います。何度でもその値を変更することができます。

`var` は、以前は使用されていましたが、数々の不具合があるので、今は使用してはなりません。

4.4 ウェブブラウザと DOM

HTML ドキュメントをブラウザで読み込むとき、DOM と呼ばれるプログラミング用のデータ表現が生成されます。DOM (Document Object Model) とは、HTML ドキュメントのコンテンツと構造を JavaScript から操作できるオブジェクトです。DOM では HTML ドキュメントのタグの入れ子関係を木構造で表現するため、DOM が表現する HTML タグの木構造を DOM ツリーと呼びます。

たとえば、DOM には HTML ドキュメントそのものを表現する `document` グローバルオブジェクトがあります。`document` グローバルオブジェクトには、指定した HTML 要素を取得したり、新しく HTML 要素を作成するメソッドが実装されています。`document` グローバルオブジェクトを使うことで、先ほどの `index.html` に書かれた HTML を JavaScript から操作できます。^{*5}

▼ HTML の操作

```
// CSSセレクタを使ってDOMツリー中のh1要素を取得する
const heading = document.querySelector("h1");

// h2要素に含まれるテキストコンテンツを取得する
const headingText = heading.textContent;
```

^{*4} 厳密には「再代入できない変数」の宣言です。

^{*5} 出典：JavaScript Primer 迷わないための入門書

```
// button要素を作成する
const button = document.createElement("button");
button.textContent = "ボタンを押してください";

// body要素の子要素としてbuttonを挿入する
document.body.appendChild(button);
```

JavaScript と DOM は Web アプリケーション開発において切っても切り離せない関係です。動的な Web アプリケーションを作るためには、JavaScript による DOM の操作が不可欠です。

少し、難しい説明となりましたが、ここで抑えておいて欲しい要点は「JavaScript から HTML を操作できる」です。後ほどまた解説する機会もありますので、先へ進みましょう。

4.5 イベントリスナ

先程のプログラムでは、利用者（ユーザ）が何もすることなく、勝利メッセージが表示されました。これではゲームらしくありません。「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、「何か操作を行った時」に、指定した動作をする機能があります。**イベントリスナ** と呼ばれる仕組みです。

イベントとはなんでしょうか？

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。*6

リスナーとはなんでしょうか？

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。一般的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。IT の分野では「リスナ」と長音を省略して表記することも多い。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。^{*7}

つまり、イベントリスナーとは次のようになります。

「ページの読み込みやクリック動作など、ウェブページ上で行われる様々な動作を常時起動し待ち受けて（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」^{*8}

4.6 関数 (function)

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。

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` からはじ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);
console.log(関数の結果); // => 関数の返り値
```

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	- 関数の呼び出し時に渡された値が入る変数。
	複数ある場合は、(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()と関数名にカッコをつけることで呼び出せます。関数を引数と共に呼ぶ際は、関数名(引数1, 引数2)とし、引数が複数ある場合は、(カンマ)で区切ります。関数の中身ではreturn文によって、関数の実行結果として任意の値を返せます。^{*9}

4.7

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。

▼ janken.js

```
1 // 潜在的なバグを減らす。
2 'use strict';
3
4 // イベントリスナーの設定
5 const playButton = document.getElementById("play");
6 playButton.addEventListener('click', jankenHandler);
7
8 // ジャンケンの勝ち負けの結果を表示する関数
9 function jankenHandler(event) {
10   alert("あなたの勝ちです!");
11 }
```

^{*9} 出典：JavaScript Primer 迷わないとめの入門書

```

12 // totalの初期値は0
13 let total = 0;
14
15 // for文の実行の流れ
16 // まず、iを1で初期化
17 // iが10以下（条件式を満たす）ならfor文の処理を実行
18 // iに1を足し(i++)、再び条件式の判定へ
19 for (let i = 1; i <= 10; i++) {
20   // 1から10の値をtotalに加算している
21   total = total + i;
22 }
23
24
25 // コンソールに55が出力される
26 console.log(total);

```

一行ずつ解説していきます。

5行目の `const playButton = document.getElementById("play");` ですが、`document.getElementById("play")` で、html に書いた play ボタン要素を取得します。(DOM という仕組みがあり、JavaScript から、HTML の要素を操作できたことを思い出してください)。そして取得した要素をプログラムの中で扱いやすくするために、`playButton` という変数名を付けています。

6 行 目 の `playButton.addEventListener('click', jankenHandler);` ですが、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。^{*10}

9行目から11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。じゃんけん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` にイベントハンドラ

^{*10} `jankenHandler` という関数が、クリックして欲しいなと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かもしれません。

を設定したので、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、janken.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。

4.8 繰り返し処理

それでは、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!");
}
```

こういうふ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？ 百行書くのはとても大変です。

プログラムでは、同じ処理を繰り返したいときもよくあります。ですので、そのための専用の構文（書き方）が用意されています。基本の for 文を使って、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ね。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。

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。たびたび

登場している名著「JavaScript Primer 迷わないとめの入門書」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 文の実行フローは次のようになります。

1. 初期化式 で変数の宣言
2. 条件式 の評価結果が true なら次のステップへ、false 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

次のコードは、for 文 で 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です。今まで作って来た、janken.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;

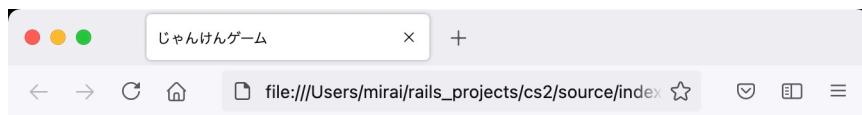
// for文の実行の流れ
// まず、iを1で初期化
// iが10以下（条件式を満たす）ならfor文の処理を実行
// iに1を足し(i++)、再び条件式の判定へ
for (let i = 1; i <= 10; i++) {
    // 1から10の値をtotalに加算している
    total = total + i;
}

// 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、`console.log()` という命令を使って、ブラウザのコンソール画面という開発者用の画面に結果を表示できます。Firefox では、次のように

すると、開発者ツールのウェブコンソールと呼ばれる画面を見ることができます。

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



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。

情報技術を夢の架け橋に
©アトリエ未来



コンソール画面に55と出力されているはずです。

4.9 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) 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;
}
```

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 の中身が実行されます。

▼if 文の条件式が true

```
// 年齢は20歳
const age = 20;

if (age >= 18) {
    console.log("大人料金です");
}
```

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。

▼else if 文

```
// 年齢は10歳
const age = 10;

if (age >= 18) {
    console.log("大人料金です");
} else if (age >= 12) {
    console.log("中高生料金です");
} else if (age >= 6) {
    console.log("小学生料金です");
}
```

`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。一方で条件に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。

▼ else if 文

```
// 年齢は5歳
const age = 5;

if (age >= 18) {
    console.log("大人料金です");
} else if (age >= 12) {
    console.log("中高生料金です");
} else if (age >= 6) {
    console.log("小学生料金です");
} else {
    console.log("無料です");
}
```

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。いつも勝ちでは、面白くないですね。勝ったり負けたりするからこそ、やりがいもあるというものです。

今、学んだ if 文 を使うと、勝ったり負けたりあいこだつたりが実現できそうです。以下のように書いてみましょう。

▼ janken.js

```
1 // 潜在的なバグを減らす。
2 'use strict';
3
4 // イベントリスナーの設定
5 // 開始ボタンを押されるとゲーム開始
6 const playButton = document.getElementById("play");
7 playButton.addEventListener('click', jankenHandler);
8
9 // player の手を取得
10 const inputBox = document.getElementById("player_hand_type");
11 let player = parseInt(inputBox.value);
12
13 // computer の手を設定
14 let computer = 0; // グー
15
16 // じゃんけんの勝ち負けの結果を表示する関数
17 function jankenHandler(event) {
18     // === は「等価演算子」で、「等しい」ことを調べます。
19     if (player === 0) {
20         // プレイヤーがグーの時に行う処理を記します。
```

```

21 // ここでは、alert文を使い、画面表示します。
22 alert("あいこです!");
23 } else if (player === 1) {
24 // プレイヤーがチョキの時の処理を記します。
25 alert("あなたの負けです!");
26 } else {
27 // プレイヤーがパーの時の処理を記します。
28 alert("あなたの勝ちです!");
29 }
30 }
```

7行目までは、前回と一緒にですね。10行目と11行目で、プレイヤーが入力した手を取得しています。index.html 内で、`<input type="number" id="player_hand_type" value="0">` と書いたことを覚えていますでしょうか。

`document.getElementById("player_hand_type");` と書くことで、このプレイヤーの手の入力枠を、JavaScript で取得することができます。取得した入力枠要素を、プログラム中で、扱いやすくするために、`const inputBox =` と書いて、変数に代入しています。そして、`inputBox.value` と書くことで、プレイヤーが枠に入力した値を得ることができます。

`parseInt` は、文字列としての "0" を解析 (parse) し、整数値としての 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。

14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 という約束でプログラムを作っていますので、コンピュータは必ず「グー」を出すことにしています。

それでは、プレイヤーの手とコンピュータの手が出そろったので、勝ち負けの判定を行いましょう。先ほど学んだ `if` 文を使うと良いですね。ここでは、「`==`」厳密等価演算子を使って、プレイヤーの手とコンピュータの手を比較しています。19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。ブラウザを再読み込みしてやってみましょう。ぐつと、じゃんけんゲームらしくなってきましたね。

4.10 亂数を利用する

さて、コンピュータの手は、「0」つまりいつも「グー」でした。なので、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」の手を変えるようにしましょう。

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。

```
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  
1  
0
```

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

【コラム】乱数関数を自作する

関数化して、分かりやすい名前をつけると、一目でプログラムの処理を理解することもできます。また、将来、双六ゲームを作りたければ、1～6の乱数が欲しいでしょうから、ここで作っておくと便利そうです。

```
// 亂数を返す関数
// rand(0, 2)と呼ぶと、
// 0, 1, 2 と グーチョキパー の乱数を返す。
// rand(1, 6)と呼ぶと、
// 1, 2, 3, 4, 5, 6 と サイコロの乱数を返す。
function rand(min, max){
    return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

こうすることで、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);
```

分かりやすくなりましたね。

4.11 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。

▼ janken.js

```
1 // 潜在的なバグを減らす。
2 'use strict';
3
4 // イベントリスナーの設定
5 // 開始ボタンを押されるとゲーム開始
6 const playButton = document.getElementById("play");
7 playButton.addEventListener('click', jankenHandler);
8
9 // player の手を取得
10 const inputBox = document.getElementById("player_hand_type");
11 let player = parseInt(inputBox.value);
12
13 // computer の手を 乱数で設定
14 let computer = rand(0, 2);
15
16 // ジャンケンの勝ち負けの結果を表示する関数
17 function jankenHandler(event) {
18     if (player === 0) {
19         // === は「等価演算子」です。
20         // 「等しい」ことを調べます。
21         // プレイヤーがグーの時なら
22         if (computer === 0) {
23             // コンピュータがグーを出した場合、
24             alert("あいこです!");
25         } else if (computer === 1) {
26             // コンピュータがチョキを出した場合
27             alert("あなたの勝ちです!");
28         } else {
29             // コンピュータがパーを出した場合
30             alert("あなたの負けです!");
31         }
32     } else if (player === 1) {
33         // プレイヤーがチョキの時に、
34         if (computer === 0) {
35             // コンピュータがグーを出した場合
36             alert("あなたの負けです!");
37         } else if (computer === 1) {
```

```

38 // コンピュータがチョキを出した場合
39 alert("あいこです!");
40 } else {
41 // コンピュータがパーを出した場合
42 alert("あなたの勝ちです!");
43 }
44 } else {
45 // プレイヤーがパーの時に、
46 if (computer === 0) {
47 // コンピュータがグーを出した場合
48 alert("あなたの勝ちです!");
49 } else if (computer === 1) {
50 // コンピュータがチョキを出した場合
51 alert("あなたの負けです!");
52 } else {
53 // コンピュータがパーを出した場合
54 alert("あいこです!");
55 }
56 }
57 }
58
59 // 亂数を返す関数
60 // rand(0, 2)と呼ぶと 0, 1, 2 と グーチョキパー の乱数を返す
61 function rand(min, max){
62 return Math.floor(Math.random() * (max - min + 1)) + min;
63 }
```

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。

4.12 定数

だいぶプログラムが長くなってしまった。分かりやすくするために、ここで「定数」を導入しましょう。定数とは、「プログラム内で共通して使う値」のこととで、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、人間にとっては、グーチョキパーのほうが分かりやすいものです。単なる数値にも名

前をつけることで、より大きなプログラムになった際にも分かりやすいコードを書くことができるようになります。

```
// 先頭に「'use strict';」と書くことで、潜在的なバグを減らす。
'use strict';

// 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 DRAW  = 0; // あいこ
const LOSE  = 1; // 負け
const WIN   = 2; // 勝ち

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。

```
// ジャンケン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("あいこです!");
        } 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("あいこです!");
        } 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("あいこです!");
    }
}
}

```

`0, 1, 2`といった数値から、`GUU, CHOKI, PAA`という定数に変わったことで、プログラムの機能は変わりませんが、数値ではなく、人にとって分かりやすい`GUU, CHOKI, PAA`という文字になりましたので、コメントもいろいろいくらいに、とっても読みやすくなりました。

♣ 関数化とリファクタリング

じゃんけんの勝敗判定の部分が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られる機能です。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、9通りの`if`文を書きましたが、より良いアルゴリズムを考えて、綺麗に書き直してみましょう。

「コンピュータプログラミングにおいて、プログラムの外部から見た動作を変えずにソースコードの内部構造を整理すること」を、「リファクタリング」^{*11}と言います。

*¹¹ 出典：Wikipedia

♣ 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^{*12} というブログがあります。こちらを参考に、もう少し簡潔に書けるよう、じゃんけんのアルゴリズムを考察していきましょう。

普通に `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、それぞれについて、コンピュータがグーの場合、チョキの場合、パーの場合と全部で九通りの勝敗判定が必要でした。

`if` 文 を書き連ねるのではなく、もう少し簡潔に書けるかどうか調べるために、勝敗表にまとめてみます。

じゃんけんの勝敗表

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

自分の手と、相手の手が、等しい時に「相子」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**で分かりますので、**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	相子 0	勝ち -1	負け -2
チョキ 1	負け 0	相子 0	勝ち -1
パー 2	勝ち 2	負け 1	相子 0

相子になるのは、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 の時であることが分かります。これで九通りの `if` 文 ではなく、五通りの `if` 文 で良いことが分かりました。

^{*12} <https://staku.designbits.jp/check-janken/>

勝ち負け相手の三通りの判定をするのに、五通りの `if` 文 なのは、いまいち賢くないので、もう少し考察を加えます。

よく見ると、 -2 と 1 は 3 つ離れていますし、 -1 と 2 も 3 つ離れています。ですので、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、

- 相手になるのは、3 で割って余りが 0 の時、
- 贠けになるのは、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「剩余演算」と言いますが、JavaScript では、`%` を使うと、剩余演算を行うことができます。

下に演算子の表を掲載いたします。

演算子の種類

	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。次のようになります。

▼ janken.js

```
1 // 潜在的なバグを減らす。
2 'use strict';
3
4 // 定数宣言
5 // プログラム内で共通して使う定数を宣言する。
6 // 慣習的に定数名は全て大文字で書かれる。
7 const DRAW = 0; // あいこ
8 const LOSE = 1; // 负け
9 const WIN = 2; // 勝ち
```

```
10 const GUU    = 0; // グー
11 const CHOKI = 1; // チョキ
12 const PAA    = 2; // パー
13
14 // イベントリスナーの設定
15 // 開始ボタンを押されるとゲーム開始
16 const playButton = document.getElementById("play");
17 playButton.addEventListener('click', jankenHandler);
18
19 // player の手を取得
20 const inputBox = document.getElementById("player_hand_type");
21 let player = parseInt(inputBox.value);
22
23 // computer の手を 亂数で設定
24 let computer = rand(0, 2);
25
26 // じゃんけんの勝ち負けの結果を表示する関数
27 function jankenHandler(event) {
28     // judge関数に、プレイヤーとコンピュータの手を渡して、
29     // 勝敗(相手なら0 , 負けなら1, 勝ちなら2)を得ます。
30     const result = judge(player, computer);
31
32     if (result === DRAW) {
33         alert('引き分けです！');
34     } else if (result === LOSE) {
35         alert('あなたの負けです！');
36     } else {
37         alert('あなたの勝ちです！');
38     }
39 }
40
41 // 亂数を返す関数
42 // rand(0, 2)と呼ぶと 0, 1, 2 と グーチョキパー の乱数を返す
43 function rand(min, max){
44     return Math.floor(Math.random() * (max - min + 1)) + min;
45 }
46
47 // プレイヤーの手とコンピュータの手が与えられると、
48 // 0: 引き分け 1: 負け 2: 勝ち を返す関数
49 function judge(player, computer) {
50     return (player - computer + 3) % 3;
51 }
52 }
```

とっても分かりやすくなりましたね。

第 5 章

初めての CSS

この章では、CSS を用いることでじゃんけんゲームの意匠を整えていきます。そして利用者に快適に使ってもらえるよう、UI/UX も配慮したウェブアプリに仕上げていきます。

前回までで、じゃんけんの大まかな機能の実装は完了いたしましたが、いまいち見た目が冴えない気がいたします。せっかく作ったじゃんけんゲームですから、綺麗に意匠も整えたいものです。

第一章で簡単に触れましたが、文書構造を HTML で書き、JavaScript で、利用者の入力に応じた処理を行うことができました。CSS は飾り付けや配置などの指定を行うための言語です。CSS を使って、じゃんけんアプリを綺麗にしていきましょう。

5.1

ユーザインターフェース (UI) とは

人間とコンピュータの間にある窓口のようなものです。コンピュータを操作する場合、利用者はコンピュータがどういった状態にあるのかを確認する必要があります。このために、コンピュータの画面上には、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。そして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することができます。^{*1}

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なります。ソフトウェアの開発では、利用者体験(UX)に配慮することも大切です。

利用者に快適に使ってもらえるよう、完成形は次のようにしましょう。

1. プレイヤーがどの手を出すのか、今まででは、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. 今まででは、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。
3. グーチョキパーどれを出しているのか、分かるようにし、アニメーション機能も実装します。
4. ○勝○敗と、今までの勝敗を表示できるようにします。
5. 開始ボタンを分かりやすくするために、色を付け、大きくします。
6. じゃんけんを知らない人はいないとは思いますが、簡単な紹介文と Wikipediaへのリンクを用意します。
7. じゃんけんゲームを見る端末はさまざまです。iPhoneで見る人もいますし、Macで見る人もいます。さまざまな端末で好ましい見た目を提供できるよう「レスポンシブデザイン」を行っていきます。

^{*1} 出典：IT用語辞典

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



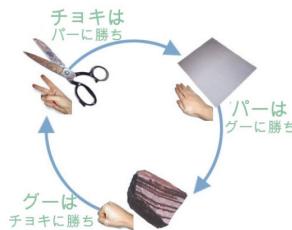
0 勝 0 敗

開始

じゃんけんは、3種類の指の出方
(グー・パー・チョキ) で三すくみの
関係を構成し、出した種類により勝敗
を決める遊戯です。古来伝承されてき
た虫拳などをもとに、明治期に九州で
発明されたと言われています。

詳しくは [ウィキペディア](#)を見てね。

じゃんけん相関図



情報技術を夢の架け橋に

© アトリエ未来

5.2

UI/UX を考慮した HTML

UI/UX に考慮して、index.html を、以下のように追加、変更します。これ
が完成形の HTML になります。

▼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     <!-- iPhoneで見た際にレイアウトを整えるための設定です -->
7     <meta name="viewport" content="width=device-width">
8     <!-- スタイルシートを読み込みます
9       これによりサイトを飾り付けします。-->
10    <link rel="stylesheet" href="janken.css">
11  </head>
12
13  <body>
14    <header>
15      <h1>じゃんけんゲーム</h1>
16    </header>
17
18    <main>
19      <!-- figure タグは 図版(figure)を示すタグです -->
20      <figure>
21        <!-- img タグは、絵(image)を表示するためのタグです -->
22        
23      </figure>
24
25      <p>
26        みんな知っているじゃんけん<br>
27        ゲー・チョキ・パー どれかを押してね
28      </p>
29
30      <!-- div は汎用タグで、適切なタグがない時に用います -->
31      <!-- クラス名をcontrol_areaとし、配置を整え易くします -->
32      <div class="control_area">
33        <!-- プレイヤーの手を選ぶ為のボタンです。 -->
34        <div class="player_hand_type">
35          <!-- JavaScript から操作しやすいよう、
36              id属性をguuに、値を0にしています。 -->
37          <button id="guu" value="0"></button>
38          <button id="choki" value="1"></button>
39          <button id="paa" value="2"></button>
40        </div>
41        <!-- 得点と開始ボタンの領域です -->
42        <div class="score_and_replay">
```

```

43         <p class="score">
44             <!-- JavaScript から操作しやすいよう、
45                 id属性をwinにしています。 -->
46             <span id="win">0</span> 勝
47             <span id="lose">0</span> 敗
48         </p>
49         <button id="play" class="button">開始</button>
50     </div>
51 </div>
52 </main>
53
54 <!-- article は 記事(article) を表す際に用いるタグです -->
55 <article>
56     <div>
57         <p>
58             じゃんけんは、3種類の指の出し方（グー・パー・チョキ）で三すくみの関係を構成し、出した種類により勝敗を決める遊戯です。古来伝承されてきた虫拳などをもとに、明治期に九州で発明されたと言われています。
59         </p>
60     <p>
61         <!-- a はanchor(錨)の意味で、他の文書へのリンクを示すウェブサイトを特徴づけるタグです。 -->
62         詳しくはウィキペディアを見てね。
63     </p>
64 </div>
65
66 <figure>
67     <!-- figcaptionタグを使うと、
68         図版.figureに標題(caption)を設定できます -->
69     <figcaption>じゃんけん相関図</figcaption>
70     
71
72 <!-- ul は箇条書き項目(Unorderd List)に用いるタグです-->
73 <ul>
74     <!-- li は 項目(List)を示す際に用いるタグです -->
75     <li>紙 > 石 : 紙は石を包む</li>
76     <li>鉄 > 紙 : 鉄は紙を切る</li>
77     <li>石 > 鉄 : 石は鉄に克つ</li>
78 </ul>
79 </figure>
80
81

```

```
82     </article>
83
84     <footer>
85         <div class="container">
86             <p>情報技術を夢の架け橋に</p>
87             <p>&copy; アトリエ未来</p>
88         </div>
89     </footer>
90
91     <script src="janken.js"></script>
92
93 </body>
94 </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。

♣ レスポンシブ対応

```
6 <!-- iPhoneで見た際にレイアウトを整えるための設定です -->
7 <meta name="viewport" content="width=device-width">
```

と新しい記述があります。レスポンシブ対応と呼ばれる iPhone で見た際にレイアウトを整えるための設定です。これにより、CSS 側で、iPhone 用、Mac 用と、端末に応じて CSS を変更することができるようになります。

♣ スタイルシートの読み込み

```
8 <!-- スタイルシートを読み込みます
9     これによりサイトを飾り付けします。-->
10    <link rel="stylesheet" href="janken.css">
```

デザインしたスタイルシートを読み込むための指定です。janken.css ファイル内にいろいろ記述し、ウェブサイトの見た目を整えていきます。一般にウェブサイトは複数のページがありますが、同じ css を使うことで、ウェブサイト全体の統一感を持たせることもできます。また、次回別のウェブサイトの作成の際に転用することも可能となり、生産性の向上につながります。

♣ 画像ファイルの指定

```
21 <!-- img タグは、絵(image)を表示するためのタグです -->
22 
```

`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.png"` として、グーの絵を表示させています。

♣ id 属性

```
22 
```

`id="computer_hand_type"` と `id` 属性 を付与しています。 `id` 属性 は
 ページ内で一つしか存在しない要素に対して用います。ここでは、JavaScript
 から要素を取得しやすくするために `id` 属性 を付与しています。

♣ class 属性

```
30 <!-- div は汎用タグで、適切なタグがない時に用います -->
31 <!-- クラス名をcontrol_areaとし、配置を整え易くします -->
32 <div class="control_area">
```

`class="control_area"` と `class` 属性 を付与しています。 `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

```
33 <!-- プレイヤーの手を選ぶ為のボタンです。 -->
34 <div class="player_hand_type">
35   <!-- JavaScript から操作しやすいよう、
36     id属性をguuに、値を0にしています。 -->
37   <button id="guu" value="0"></button>
38   <button id="choki" value="1"></button>
39   <button id="paa" value="2"></button>
40 </div>
```

UI 改善のため、ボタン要素を導入しています。CSS でグーの絵を表示させ
 ているので分かりやすくなります。これにより、今までグーの時は `0` と入力

していましたが、グーのボタンを押すだけで良くなるので、利用者が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。

♣ a 要素と URL エンコーディング

63 詳しくは
64 ウィキペディアを見てね。

a は、 anchor (錨) 要素です。他の文書へのリンクを示します。ウェブサイトを特徴づけるタグです。じゃんけんに関する Wikipedia へのリンクを用意しています。

そして、歴史的な経緯により、URL に使える文字は、半角英数文字です。

```
<a href="https://ja.wikipedia.org/wiki/じゃんけん">ウィキペディア</a>
```

と書けると良いのですが、半角英数文字のみが利用可能なため、「URL エンコーディング」という方式により、「じゃんけん」を「%E3%81%98%E3%82%83%E3%82%93%E3%81%91%E3%82%93」に変換しています。

5.3 CSS ファイルの作成

HTML の変更に応じて、新しく `janken.css` というファイルを作り、次のように記述します。また同様のスタイル指定が繰り返し登場するため、少し長くなっていますが、理解しやすいようコメント (説明書き) もつけていますので、ご覧ください。

▼janken.css

```

1  /* 色の指定 (CSSカスタムプロパティ) */
2  :root {
3      --amairo:          #2ca9e1; /* 天色(あまいろ) */
4      --nibiro:           #9ea1a3; /* 鈍色(にびいろ) */
5      --kurohairo:        #0d0d0d; /* 黒羽色(くろはいろ) */
6      --sakurairo:        #fef4f4; /* 桜色(さくらいろ) */
7  }
8
9  /* 基本設定 */
10 * { /* 全ての要素(*)を対象に */
11     margin: 0; /* 余白を0にする */
12     box-sizing: border-box;
13 }
14
15 img { /* 全てのimg要素を対象に */
16     width: 100%; /* 幅を画面幅にする */
17     height: auto; /* 高さは縦横比を保つようにする */
18 }
19
20 /* ページ全体の設定 */
21 body { /* body は全ての要素の親なので */
22     display: grid; /* グリッド(格子)線を使うモードにする */
23     /* column(列) 左右に20px 残りは中央 */
24     grid-template-columns: 20px 1fr 20px;
25     /* row(行)を用意する */
26     grid-template-rows:
27         [head]      100px /* 一行目の高さは100px headと命名 */
28         [main]      auto  /* 二行目の高さは自動 titleと命名 */
29         [article]   auto  /* 三行目の高さは自動 subtitleと命名 */
30         [foot]      100px; /* 四行目の高さは100px footと命名 */
31 }
32
33 /* 部品の配置 */
34 body > * { /* body の直下(>) にある全ての要素(*)を対象に */
35     grid-column: 2 / -2; /* 列配置は 左から2番目の線から
36                           右から数えて二番目(-2)の線まで */
37 }
38
39 /* ヘッダー */
40 header { /* header 要素を対象に */
41     grid-row: head; /* 行の配置は先ほど命名したheadの線の下に */
42     justify-self: center; /* 左右中央揃えで配置する */

```

```
43 align-self: center; /* 上下中央揃えで配置する */
44 font-size: 20px; /* 書体の大きさは40px */
45 color: var(--kurohairo); /* 文字の色は、黒羽色 #0d0d0d; */
46 /* 右に5px 下に5px ずれた場所にぼかし幅5px で
47 鈍色 #9ea1a3 の影を付ける */
48 text-shadow: 5px 5px 5px var(--nibiiro);
49 }
50
51 header h1 { /* header 内の h1 (大見出しの指定) */
52   text-align: center; /* 文字は中央揃えにする */
53 }
54
55 /* メイン(ウェブサイトの主要機能部)用のスタイル */
56 main { /* main 要素を対象に */
57   grid-row: main;
58 }
59
60 main figure img { /* コンピュータのじゃんけんの絵です */
61   max-height: 300px; /* 絵が歪まぬよう、最大高さを指定します */
62 }
63
64 main p { /* じゃんけんの説明文です */
65   text-align: center; /* 真ん中揃えにします */
66   margin-bottom: 20px; /* p 要素の下側に少し余白を設けます */
67 }
68
69 /* class="control_area" とクラス属性を付与した要素を対象に */
70 .control_area {
71   /* control_area内部の要素もグリッドレイアウトで配置します */
72   display: grid;
73   grid-template-columns: 1fr; /* column(列) は 一列 */
74   grid-template-rows: 1fr auto; /* row(行) は 二行用意します */
75   row-gap: 20px;
76   /* 行の間隔は20px開けます */
77 }
78
79
80 /* control_area内でplayer_hand_typeと
81 クラス属性を付与した要素を対象に */
82 .control_area .player_hand_type {
83   /* 内部の要素もグリッドレイアウトで配置します */
84   display: grid;
85   grid-template-columns: 1fr 1fr 1fr; /* 三列用意します */
```

```
86     grid-template-rows: 1fr; /* 一行用意します */
87 }
88
89 .control_area .player_hand_type button {
90     background-color: transparent; /* ボタンの背景色は透明に */
91 }
92
93 /* ゲーチョキパーの背景画像の設定 */
94 /* html で id="guu" と id 属性を付けている */
95 #guu { background-image: url('player_guu.png'); }
96 #choki { background-image: url('player_choki.png'); }
97 #paa { background-image: url('player_paa.png'); }
98
99 /* ゲーチョキパー各ボタンの大きさを指定 */
100 .player_hand_type button {
101     background-size: 100% 100%; /* 背景画像の大きさは縦横100% */
102     border: none; /* ボタンの枠線は無し */
103     height: 100px; /* ボタンの高さは100px */
104     padding: 0; /* 内部への詰め物は無し */
105     /* カーソルを乗せた時の形状は、pointer(手のマーク) */
106     cursor: pointer;
107 }
108
109 .score_and_replay { /* 得点領域の指定 */
110     /* 内部の要素もグリッドレイアウトで配置します */
111     display: grid;
112     grid-template-columns: 1fr 1fr; /* 二列用意します */
113     grid-template-rows: 1fr; /* 一行用意します */
114     align-self: center; /* 上下方向に中央揃えにします */
115 }
116
117 /* クラス属性.score と id属性#play を付与した要素を対象に */
118 .score,
119 #play {
120     font-size: 24px; /* 書体の大きさは24px */
121     margin: 10px 0; /* 上下に10px 左右に0px の余白 */
122     align-self: center; /* 上下方向で中央揃え */
123 }
124
125 /* <a class="button"> と buttonクラスを付与したa要素を対象に */
126 .button {
127     /* リンクには下線が付くが、飾り付け(decoration)は、無し */
128     text-decoration: none;
```

```
129 color: white; /* 文字の色は白*/
130 /* 枠線は、しっかりとした(solid)線で、1px幅の白で */
131 border: solid 1px white;
132 padding: 10px; /* 詰め物は、上下左右に10px で */
133 border-radius: 10px; /* 角は、半径(radius) 10pxで丸くする */
134 background: var(--amairo); /* 背景色は、天色で */
135 cursor: pointer; /* カーソルを乗せると、pointer(手のマーク) */
136 }
137
138 /* じ ゃんけん紹介記事 */
139 article { /* aritcle 要素を対象に */
140 /* 行の配置は、先ほど命名した article の線の下に */
141 grid-row: article;
142 /* 枠線は5pxの太さで二重線 色は移色 */
143 border: 5px double var(--utsushiro);
144 border-radius: 1rem; /* 角を1文字分丸くする */
145 padding-bottom: 1rem; /* 要素下側に1文字分詰め物をする */
146 margin-bottom: 1rem; /* 要素下側に1文字分余白を設ける */
147
148 /* 内部の要素もグリッドレイアウトで配置します */
149 display: grid;
150 grid-template-columns: 1fr; /* 列は一列 */
151 grid-template-rows: auto auto; /* 行は二列 用意する */
152 margin-top: 20px; /* 上側を20px開ける */
153 justify-self: center; /* 水平方向に中央揃え */
154 }
155
156 article p { /* aritcle 要素 内の p要素を対象に */
157 font-size: 18px; /* 書体の大きさは10px */
158 text-align: left; /* 文字は左寄せ */
159 padding: 1rem; /* 1文字分詰め物をする */
160 text-indent: 1rem; /* 文字を1文字字下げする */
161 }
162
163 /* aritcle内の figure内の figcaption要素を対象に */
164 article figure figcaption {
165 font-size: 24px; /* 書体の大きさは24px */
166 font-weight: bold; /* 文字は太字 */
167 text-align: center; /* 文字は中央揃え */
168 margin-bottom: 1rem; /* 下側に1文字分間隔を取る */
169 }
170
171 article figure img { /* aritcle内の figure内の img要素を対象に */
```

```

172    >/
173    width: 75%;      /* 画像の横幅の指定 */
174    max-width: 400px; /* 画像の最大幅の指定 */
175    height: auto;     /* 高さは、横幅に応じて自動 */
176
177    display: block;   /* 中央揃えのためにブロックタイプに変更 */
178    margin: 0 auto;   /* 左右中央揃えに */
179
180    margin-bottom: 1rem; /* 画像の下を1文字分開ける */
181 }
182
183 article ul {           /* aritcle内の ul内の li要素を対象に */
184   list-style: none;     /* 箇条書きの・(黒丸)は不要 */
185   padding: 0;           /* 詰め物も不要 */
186 }
187
188 article ul li {        /* aritcle内の ul内の li要素を対象に */
189   justify-self: center; /* 水平方向に中央揃え */
190   text-align: center;   /* 文字は中央揃え */
191 }
192
193 /*フッター*/
194 footer { /* footer 要素を対象に */
195   /* 列配置は、左から一番目の線から、
196   右から数えて一番目(-1)の線まで */
197   grid-column: 1 / -1;
198   grid-row: foot; /* 行の配置は、先ほど命名した foot の線の下 */
199   background: var(--sakurairo); /* 背景色は、桜色 */
200   display: grid; /* グリッド(格子)線を使うモードにする */
201 }
202
203 footer div { /* footer 要素の中のdiv要素を対象に */
204   grid-column: 1; /* 1番目のグリッド線の右側に配置 */
205   grid-row: 1;    /* 1番目のグリッド線の下側に配置 */
206   justify-self: center; /* 左右中央揃えで配置する */
207   align-self: center; /* 上下中央揃えで配置する */
208   text-align: center;
209 }
210
211 /* デスクトップ版の追加設定 */
212 /* レスポンシブ対応 幅768px以上の端末用に追加CSSを記述する */
213 @media (min-width: 768px) {
214   body {

```

```
214 display: grid; /* グリッド(格子)線を使うモードにする */
215 /* column(列) の設定 */
216 /* 左右に余白 中央に375pxを二列確保 */
217 grid-template-columns: 1fr 375px 375px 1fr;
218 grid-template-rows: /* row(行)を用意する */
219     [head]           100px /* 一行目の高さは100px */
220                         /* 線名をheadと命名 */
221     [main article] auto /* 二行目の高さは自動 */
222                         /* 線名を main と命名 */
223                         /* 線名を article と命名 */
224                         /* 二つの線を一本に重ねる */
225     [foot]          100px /* 四行目の高さは100px footと命名 */
226 }
227
228 /* main 要素を二列目から三列目までに配置 */
229 main {
230     grid-column: 2 / 3;
231 }
232
233 /* コンピュータのじゃんけんの絵の最大高さを400pxに */
234 main figure img {
235     max-height: 400px;
236 }
237
238 /* プレイヤーのじゃんけんボタンの高さを140pxに */
239 .player_hand_type button {
240     height: 140px;
241 }
242
243 /* aritcle 要素を三列目から四列目までに配置 */
244 article {
245     grid-column: 3 / 4;
246     margin-left: 20px;
247 }
248 }
```

5.4 CSS の基本

紙幅の都合上、全ては説明できないので、要点のみを解説します。

```
/* 基本設定 */
```

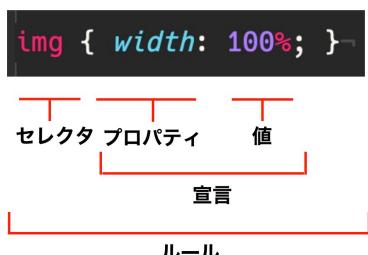
CSS でのコメントです。`/* */` と手で入力もできますが、Comannnd + / で入力できます。たくさんの CSS を書きますので、CSS の見出しや書いた CSS の説明文として、活用しましょう。

```
* {  
  margin: 0;  
}
```

デザインをし易くするため、全ての要素の余白を 0 にします。

```
img {  
  width: 100%;  
  height: auto;  
}
```

写真の幅を、横幅一杯 (100%) に、高さは、縦横比が保たれるよう、自動にします。



CSS ではデザインやレイアウトの設定をどこに適用するかを「セレクタ」で指定します。`` タグのデザインを変更したい場合には、セレクタを「`img`」と指定します。

どのようなデザインにするかは、プロパティと値の組で `\{ \}` で囲んで記述します。例えば横幅を変更したければ `width` プロパティを利用します。プロパティと値とは `@:` (コロン) で区切れます。

プロパティと値の組は、「宣言」と呼ばれます。複数の宣言は、`;` (セミコロン) で区切って記述します。

セレクタと宣言の記述全体は、「ルール」と呼ばれます。同じところに適用したいルールも、必要に応じて複数のルールに分けて記述することができます。

A screenshot of a code editor showing two CSS rules:

```
img {  
  width: 100%;  
}  
  
img {  
  height: auto;  
}
```

複数のルールを記述した例

いずれも等価な CSS です。

```
img {  
  width: 100%;  
  height: auto;  
}
```

```
img {  
  width: 100%;  
  width: 50%;  
}
```

```
img {  
  width: 100%;  
}
```

```
img {  
  width: 50%;  
}
```

同じ適用先に、同じプロパティを複数指定した場合、後から記述した設定が優先され、先に記述した設定を上書きします。

左の場合、横幅 width の値は 50% になります。

5.5 CSS セレクタ

セレクタでは、CSS の設定を「どこ」に適用するかを指定します。基本的なセレクタの書き方は次の通りです。

```
<body>
  <img src="">
  <article><img src=""></article>
  <article><img src=""></article>
</body>
```

全ての要素* { } セレクタを「*」と指定すると、全ての要素が適用先となります。

```
<body>
  <img src="">
  <article><img src=""></article>
  <article><img src=""></article>
</body>
```

全ての img 要素 img { } 要素名を指定すると、その要素のみに適用先を限定できます。

例えばセレクタを「img」と指定すると、img 要素のみが適用先となります。

```
<body>
  <img src="">
  <article><img src=""></article>
  <article><img src=""></article>
</body>
```

特定の要素の中の img 要素 article img { } HTML の階層構造を使って適用先を限定することもできます。

例えばセレクタを「article img」と指定すると、article の中の img 要素が適用先となります。

```
<body>
  <img src="">
  <article><img src=""></article>
  <article><img src=""></article>
</body>
```

一階層下の img body > img { } 特定の要素の一階層下の要素に限定して適用することもできます。

例えばセレクタを「body > img」と指定すると、body の一階層下の img 要素のみが適用先となります。

```
<body>
  <img src="">
    <article><img src=""></article>
    <article><img src=""></article>
</body>
```

特定の要素に隣接する article img

`+ article { }` 特定の要素に隣接した要素に限定して適用することもできます。

例えばセレクタを「img + article」と指定すると、img に続けて記述した同じ階層の article 要素が適用先となります。

```
<body>
  <img src="">
    <article><img src=""></article>
    <article><img src=""></article>
</body>
```

特定の要素の後に記述した全ての article img ~ article { }

`特定の要素の後に記述した要素に限定して適用することもできます。`

例えばセレクタを「img ~ article」と指定すると、img の後に記述した同じ階層の全ての article 要素が適用先となります。

```
<body>
  <img src="">
    <article><img src=""></article>
    <article><img src=""></article>
</body>
```

複数の適用先 body > img, img

`+ article { }` 複数の適用先に同じ設定を適用したい場合、セレクタを「,(カンマ)」で区切って指定します。

例えば「body > img」と「img + article」の二つのセレクタをカンマ区切りで指定すると、body の一階層下の img 要素と、img に続けて記述した同じ階層の article 要素が適用先となります。

```
<body>
  
  <p>みんな知っているじゃんけん</p>
  <p class="score">0勝0敗</p>
</body>
```

ID 属性#computer ページ内に一つしか存在しない要素の場合、ID 属性で指定することも可能です。id 属性を指定する際は、「 #(シャープ) 」を用います。

例えば、セレクタを「 #computer 」と指定すると、id="computer" 属性を付与した要素が適用先となります。

```
<body>
  
  <p>みんな知っているじゃんけん</p>
  <p class="score">0勝0敗</p>
</body>
```

class 属性.score { } クラス属性を付与すると、要素名などでは区別しづらい要素を特定しやすく便利です。class 属性を指定する際は、「.(ピリオド)」を用います。

例えば、セレクタを「 .score 」と指定すると、class="score" 属性を付与した要素が適用先となります。

5.6 CSS グリッドレイアウトとは

Web 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）を用いて、自由自在に要素を配置することができます。

♣ グリッドとは何か？

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。要素をグリッド上の行と列の中に配置することができます。

♣ アイテムの配置

グリッドの線の番号や名前を使ってグリッドのある位置を指定してアイテムを配置することができます。

♣ グリッドコンテナの作成

グリッドコンテナを作成するには、要素に対して `display: grid` を指定します。グリッドコンテナを作成すると、すべての直接の子要素がグリッドアイテムへと変わります。コンテナ=箱、アイテム=要素です。みかん箱に、みかんを入れる、そのような想像をしてください。

▼ janken.css

```

21 body { /* body は全ての要素の親なので */
22   display: grid; /* グリッド(格子)線を使うモードにする */
23   /* column(列) 左右に20px 残りは中央 */
24   grid-template-columns: 20px 1fr 20px;
25   /* row(行)を用意する */
26   grid-template-rows:
27     [head]      100px /* 一行目の高さは100px headと命名 */
28     [main]      auto  /* 二行目の高さは自動 titleと命名 */
29     [article]   auto  /* 三行目の高さは自動 subtitleと命名 */
30   /* 四行目の高さは100px footと命名 */
31 }
```

`body { display: grid }` と指定することで、要素等が納まる箱（コンテナ）に設定しています。`grid-template-columns(列)` と、`grid-template-rows(行)` で、縦横に補助線（グリッド）を引き、その補助線に沿って、要素を配置することができるようになります。

♣ グリッド線を使って要素を配置する

```

40 header { /* header 要素を対象に */
41   grid-row: head; /* 行の配置は先ほど命名したheadの線の下に */
42   justify-self: center; /* 左右中央揃えで配置する */
43   align-self: center; /* 上下中央揃えで配置する */
```

補助線（グリッド）を作成したので、グリッドに沿って、要素を配置します。

例えば、header 要素を、head という線の下に配置するには、`grid-row: head` と書くことで、配置できます。そして上下左右に中央揃えすると、「じゃんけんゲーム」と真ん中に配置され綺麗に表示されます。



▲図 5.1: グリッド線による要素の配置

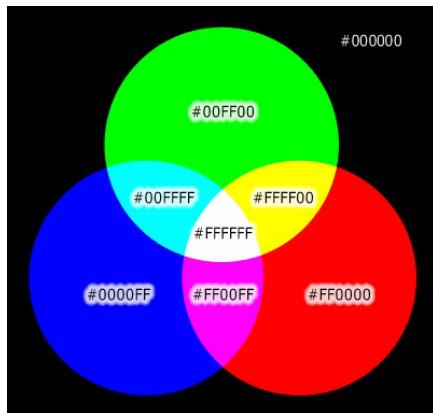
5.7 色について

パソコンの画面などは光の三原色と呼ばれる赤 (red)・緑 (green)・青 (blue) の組み合わせによって全ての色を再現しています。このような色の決め方を RGB カラーと呼んでいます。光を混ぜ合わせていくので、重なれば重ねるほど

ど明るくなっています。3色が重なる部分は白になります。

現在のコンピューターではRGBはそれぞれ256段階の値を持っています。つまり256の3乗で16,777,216色を再現することが可能になります。

ホームページで配色する場合はこの256段階を16進数に変換し、最初に#を付けて使います。^{*2}



赤は#ff0000、緑は#00ff00です。赤い光と緑の光を合わせると、黄#ffff00になります。

また、光の三原色、赤・緑・青と、色の三原色、黄・赤紫・青緑がちょうど相補性を成すのも美しいです。

♣ CSSでの色の指定をする

十六進数や色の名前を使って指定することができます。

```
h1 {
  color: #ff0000; /* 赤 */
}

p {
  color: red; /* 赤 */
}
```

^{*2} 出典：基本的な色指定であるRGB値

また、**CSS カスタムプロパティ** を定義すると、和名での指定もできます。

```
: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) */
  --botanairo: #e7609e; /* 牡丹色(ぼたんいろ) */
  --ayameiro: #674196; /* 菖蒲色(あやめいろ) */
  --otomeiro: #f3cccc; /* 乙女色(おとめいろ) */
  --momijiyo: #a61017; /* 紅葉色(もみじいろ) */
  --nibiro: #9ea1a3; /* 鈍色(にびいろ) */
  --kurohairo: #0d0d0d; /* 黒羽色(くろはいろ) */

  --sakurairo: #fef4f4; /* 桜色(さくらいろ) */
  --harukazeiro: transparent; /* 春風色(はるかぜいろ) */
}
```

▲図 5.2: CSS カスタムプロパティの定義

```
p {
  color: var(--amairo); /* 天色(あまいろ) */
```

5.8 レスポンシブデザイン

パソコンからウェブサイトを閲覧する時代は、一つのhtmlと一つのcssですみました。携帯電話が登場したことで、携帯用のウェブサイトが作られるようになりました。そして、パソコン用、携帯用、二つのウェブサイトを管理するのは大変です。そこで、一つのhtmlとcssで、パソコン、携帯に対応させ

る方法が考案されました。レスポンシブデザインと呼ばれる手法です。

@media は、メディアクエリと呼ばれます。

@media (min-width: 768px) { iPad 用の追加の css } と書くことで、画面幅が、768px 以上の端末 (iPad や Mac など) で閲覧している時に適用される追加の css を記すことができます。

▼ janken.css

```
212 /* レスポンシブ対応 幅768px以上の端末用に追加CSSを記述する */
213 @media (min-width: 768px) {
214   body {
215     (略)
216   }
217 }
```

この章の執筆は巻末の参考文献 「CSS グリッドで作る HTML5 & CSS3 レッスンブック」に多くを負っています。とても有益な書籍ですので、是非ご一読なさってみてください。

第6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。

UI/UXでの検討点は次の事柄でした。

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。
3. グーチョキパーどれを出しているのか、分かるようにし、アニメーション機能も実装します。
4. ○勝○敗と、今までの勝敗を表示できるようにします。
5. 開始ボタンを分かりやすくするために、色を付け、大きくします。
6. じゃんけんを知らない人はいないとは思いますが、簡単な紹介文と

Wikipediaへのリンクを用意します。

7. ジャンケンゲームを見る端末はさまざまです。iPhoneで見る人もいますし、Macで見る人もいます。さまざまな端末で好ましい見た目を提供できるよう「レスポンシブデザイン」を行っていきます。

このうち、5, 6, 7は既に完了していますから、残りの部分を、JavaScriptで実装していきましょう。

6.1 プレイヤーの手を取得する

▼ janken.js

```
// player の手を取得
const jankenInputBox = document.getElementById("player_hand_type");
let player = parseInt(jankenInputBox.value);
```

これまで、数値入力枠の中に利用者が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。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、今後、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ています。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がとれるようにします。

`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` と書くと、html で、`<button id="guu" value="0"></button>` と書いていた、この value 属性の値"0"を取得することができます。

`parseInt` は、文字列としての"0"を解析 (parse) し、整数値としての 0 にする関数です。

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 = parseInt(event.target.value);
```

6.2 コンピュータの手を表示する

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しょう。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の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、配列を使うととても便利です。

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（入れ物はあるけれど、中身は空っぽで容器だけの幕内弁当を想像すると分かりやすいでしょう）。`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（幕の内弁当のおかずに、グー、チョキ、パーの三つが入っている光景を想像してください）。

♣ 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。ぜひ、使えるよう、学習されてください。

.....

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 に標準で用意されている乱数は、0 から 1 の間の浮動小数点数を得ることができますが、整数を返す乱数を自作していますので、利用しましょう。

```
// 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。

6.3 アニメーション機能を実装する

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。せつ

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。(出典:IT用語辞典)

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`*1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。

*1 JavaScriptで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。

```
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。

▼リスト 6.1:

```
const FPS = 60; // 一秒間あたり、60コマ表示する
```

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。

アニメーション実行中か、否かを表す変数として、`isPause` 変数を用いるとことにして、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。

♣ 再帰関数

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;
}
```

`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、とても分かりやすくなります。

少し、寄り道とはなりますが、**プログラミング言語 Ruby** の作者として名高い、**まつもとゆきひろ**さんのエッセイをご紹介いたします。

6.4 名前重要

著者: Matz

ネイティブ・アメリカンの信仰に「すべての人物・事物には眞の名前があり、その名前を知るものはそれを支配することができる」というものがあるのだそ

うです。ですから、彼らは自分の真の名前を秘密にして、家族など本当に信頼できる人にしか打ち明けないのでそうです。そして、対外的にはあだ名を用意してそちらを使うということです。そういうえばアニメ化もされた U・K・ルーグウインの「ゲド戦記」でも同じ設定が用いられていましたね。「ゲド」というのは主人公の真の名前なので物語中にはほとんど登場せず、物語の中では彼は一貫して「ハイタカ」と呼ばれていました。

さて、プログラミングの世界において、この信仰はある程度真実ではないかと感じることがたびたびあります。つまり、事物の名前には、理屈では説明しきれない不思議なパワーがあるような気がするのです。

たとえば、私が開発している Ruby も、名前のパワーを体現しているように思えます。1993 年に Ruby の開発を始めた時、Perl にあやかって宝石の名前を選んで Ruby と命名しました。あまり深刻に考えず、宝石の名前の中から、短く、覚えやすく、美しい名前として Ruby を選んだだけでしたが、後に Ruby が、6 月の誕生石である真珠（パール）に続く、7 月の誕生石であることに気がついた時、まさに適切な名前であると感じました。また、活字もそれぞれの大きさに応じて宝石の名前が付けられているのですが、パールは 5 ポイント、ルビーは 5.5 ポイントで並んでいます。このルビーがふりがなの「ルビ」の語源になったのはまた別の話。

今、振り返って思うのは、もし私が Ruby という名前を選ばなかつたらきっと、現在の Ruby の普及を見ることはなかつただろうということです。この Ruby という名前にパワーがあったからこそ、Ruby の魅力が増加したのではないかと感じるので。ただ単に Ruby がプログラミング言語として優れているだけでなく、この名前の持つパワーによって、愛される存在となっているのではないかと感じるので。この名前があればこそ、これまでの長い間 Ruby を開発し続けるモチベーションが維持できたり、また多くのユーザが Ruby という言語に関心をもってくださったのではないかと感じています。

そんなこともあって、私の設計上の座右の銘は「名前重要」です。あらゆる

機能をデザインする時に、私はその名前にもっともこだわります。プログラマとしてのキャリアの中で、適切な名前をつけることができた機能は成功し、そうでない機能については後で後悔することが多かったように思うからです。

実際、Rubyに対する機能追加の要求に対しても、しばしば「要求は分かつた。あれば便利なのも理解できる。でも、名前が気に入らない。良い名前が決まつたら採用する」として拒否したものも数限りなくあります。しかし、名前が気に入らなかつたもので、取り入れなかつたことを後で後悔したことはほとんどありません。

これはつまりこういうことなのではないかと思います。適切な名前をつけられると言うことは、その機能が正しく理解されて、設計されているということで、逆にふさわしい名前がつけられないということは、その機能が果たすべき役割を設計者自身も十分理解できていないということなのではないでしょうか。個人的には適切な名前をつけることができた機能については、その設計の8割が完成したと考えても言い過ぎでないことが多いように思います。

ソフトウェアの設計のアプローチとして、「まず名前から入る」というのは、あまり語られていない秘訣としてもっと広く知られてもよいように思います。

*2

6.5 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。

*2 引用：プログラマが知るべき 97 のこと

```
// 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` という関数を作つて、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` 内で行つていますが、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？ 勝った数、負けた数は、html 内で、`0` と書いていました。

JavaScript で扱いやすいよう、ID 属性を付与しておいたので、`document.getElementById("win")` と書くことで、この `win` 要素を取得することができるので、`win` 変数に格納しておきましょう。

`win.textContent` と書くことで、`0` と書いていた「0」を取得することができますが、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、`parseInt` 関数を用いて、文字列の "0" を解析 (parse) し、数値としての 0 を得ます。数値としての 0 を得ることができましたから、「+1」と足し算することで、勝った数を一つ増やせます。

`win.innerText = 1`と書くと、今まで、`0`と書かれていたhtmlを、`1`と、JavaScriptで更新できます。

以上をまとめると、`updateScore`関数は、次のようにになります。

```
// 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;
    }
}
```

6.6 じゃんけんプログラム完成

長い道のりでしたが、今ようやく、じゃんけんプログラムが完成しました。完成したソースコードは次の通りです。

▼ janken.js

```
1 // 潜在的な不具合防止のために記述
2 'use strict';
3
4 // 定数宣言
5 const DRAW = 0; // 引き分け
6 const LOSE = 1; // 負け
7 const WIN = 2; // 勝ち
8
9 const GUU = 0; // グー
10 const CHOKI = 1; // チョキ
11 const PAA = 2; // パー
12
13 const FPS = 24; // 一秒間あたり、24コマ表示する
```

```
14 // グローバル変数宣言
15 let isPause = true; // ゲー・チョキ・パーの切替アニメを制御す
16 >るための変数
17 let computer; // コンピュータの手（ゲー:0、チョキ:1、パー:2 のい
18 >ずれか）
19 // メイン処理
20 function main(){
21   if(!isPause){ // 停止中でなければ
22     computer = rand(0, 2);
23     document.getElementById('computer_hand_type').src =
24       ["guu.png", "choki.png", "paa.png"][computer];
25   }
26   setTimeout(main, 1000 / FPS);
27 }
28
29 // ボタン初期化関数
30 function initButton() {
31   const guu_button = document.getElementById("guu");
32   const choki_button = document.getElementById("choki");
33   const paa_button = document.getElementById("paa");
34   guu_button.addEventListener('click', jankenHandler);
35   choki_button.addEventListener('click', jankenHandler);
36   paa_button.addEventListener('click', jankenHandler);
37
38   // playボタンがクリックされた時には、resume関数を実行して、
39   // じゃんけんの切替アニメが再開(resume)されるようにする。
40   const play_button = document.getElementById("play");
41   play_button.addEventListener('click', resume);
42 }
43
44 // じゃんけんの勝敗を取り扱う関数
45 function jankenHandler(event) {
46   // 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
47 >する
48   const play_button = document.getElementById("play");
49   play_button.innerText = "もう一度";
50
51   // アニメーション停止処理実行
52   pause();
53
54   // プレイヤーの手の取得
```

```
54 const player = parseInt(event.target.value);
55 // 勝敗結果の取得
56 const result = judge(player, computer);
57 // 勝敗に応じ、メッセージ表示 & 勝敗更新
58 if (result === DRAW) {
59     alert('引き分けです！');
60 } else if (result === LOSE) {
61     alert('あなたの負けです！');
62     // 敗数を一つ増やす
63     updateScore(LOSE);
64 } else {
65     alert('あなたの勝ちです！');
66     // 勝数を一つ増やす
67     updateScore(WIN);
68 }
69 }
70
71 // じゃんけんの効果的な勝敗判定アルゴリズム
72 function judge(player, computer) {
73     return (player - computer + 3) % 3;
74 }
75
76 // 勝敗更新処理
77 function updateScore(result) {
78     const win = document.getElementById("win");
79     const lose = document.getElementById("lose");
80
81     if (result === WIN) { // 勝ちの場合
82         win.innerText = parseInt(win.textContent) + 1;
83     } else if (result === LOSE) {
84         lose.innerText = parseInt(lose.textContent) + 1;
85     }
86 }
87
88 // 切替アニメ停止処理
89 function pause(){
90     isPause = true;
91 }
92
93 // 切替アニメ再開処理
94 function resume(){
95     isPause = false;
96 }
```

```
97 // 亂数を返す関数
98 // 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
99
100 function rand(min, max){
101   return Math.floor(Math.random() * (max - min + 1)) + min;
102 }
103
104 // 実際の処理の開始
105 initButton(); // ボタンの初期化を行う。
106 pause();      // 切替アニメを停止状態にする。
107 main();       // 切替アニメを実行待ちにし、
108             // 開始ボタンが押されると切替アニメが実行される。
```

6.7 ウェブサイトへの公開

長い道のりをかけて、完成したじゃんけんプログラム。みんなにも使ってもらえるよう、インターネットに公開してみましょう。

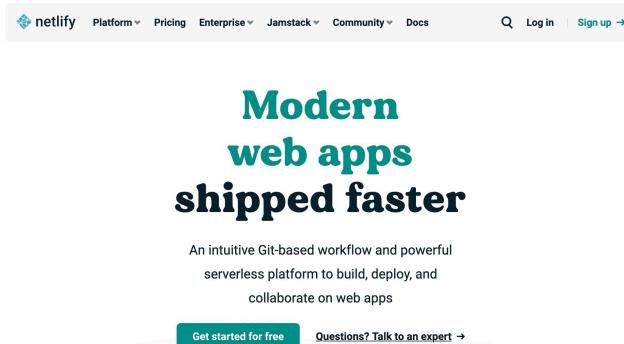
Netlify^{*3} は、静的サイトをホスティングすることができる Web サービスです。今までウェブサイトを閲覧してもらうためには、サーバーと呼ばれるコンピュータに諸々の設定を行って・・・と、とても大変でしたが、Netlify を利用すると、とても簡単にウェブサイトを公開することができます。

小規模利用は無料で可能ですので、今まで作ったウェブサイトを公開してみましょう。

Netlify は、Sign up から、利用者登録を行うより便利に使うことができます。今回は簡単に体験できるよう、<https://app.netlify.com/drop>^{*4}に直接アップロードしましょう。ドラッグ&ドロップで簡単に、作ったウェブサイトを 24 時間公開できます。

^{*3} <https://www.netlify.com>

^{*4} <https://app.netlify.com/drop>



▲図 6.1: ホスティングサービス Netlify



▲図 6.2: Netlify で公開する

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。手にして読み、自らの血肉として頂ければ幸いです。^{*1}

A.1

HTML / CSS / JavaScript を学ぶ

♣ CSS グリッドで作る HTML5 & CSS3 レッスンブック

難易度 ★★☆



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 グリッドを基礎になると、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 Web サイトの構築を学びたい人 ●最新の CSS グリッドに関する知識を得たいと考えている人に最適の一冊です。

^{*1} 紹介文は書籍紹介からの引用・改変です。

♣ JavaScript Primer 迷わないための入門書

難易度 ★★★



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。

♣ 動く Web デザインアイディア帳

難易度 ★★☆



「サイトに動きをつける時、同じ内容を何回も検索をしてソースコードを探している」「サイトで見つけたソースコードをそのままコピペしてみたけど、動かない」「JavaScript の本を購入してみたが、実際のサイトにどう組みこめばいいか具体的なイメージがわからない」「動きの原理を最低限理解して、とにかく早く実務にいかしたい」

本書は Web サイトを動かすことが苦手な右脳系ウェブデザイナーが、サイトを動かす第 1 歩を踏み出すための「動きの逆引き事典」です。近年のウェブサイトで使用されている基本的な動きの原理や仕組みをサンプルコードと共に紹介します。

A.2

コンピュータサイエンスの基礎を学ぶ

基礎編では軽く触れるだけでしたが、ぜひコンピュータサイエンスの世界をより深く学んでください。

♣ みんなのコンピュータサイエンス

難易度 ★☆☆



もはやコンピュータなしには生活が立ち行かなくなるレベルにまで到達しつつある現代社会。その圧倒的なパワーを問題解決に援用するためには小手先の知識だけでは追いつきません。かといって行き当たりばつたりで、全方位に知識を求めるには、その世界は広大にすぎますし、効率が悪すぎます。

本書はコンピュータサイエンスが扱う「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、そのエッセンスと背景となる考え方を紹介します。

どのジャンルであれ、トップクラスのエンジニアを目指すにはコンピュータサイエンスが不可欠ですが、「どこから手を付ければいいのかわからない」「砂を噛むような分厚い理論書は敬遠したい」というステップアップしたいエンジニアやその予備軍、あるいは現役だけれどももう少し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

難易度 ★☆☆



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。

2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。

♣ 教養としてのコンピューターサイエンス講義

難易度 ★☆☆



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・カーニハン氏は、プログラミング言語 C 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

難易度 ★☆☆

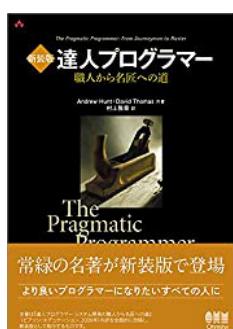


可愛いイラストでとてもわかりやすい解説を行っているため、IT パスポート試験にとって、まず大切な「解説書を一冊読み、用語や計算に慣れる」ことができる書籍です。

A.3 プログラマとして上達したい方のために

♣ 新装版 達人プログラマー 職人から名匠への道

難易度 ★★★



本書は、より生産的になりたいソフトウェア開発者に向け、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、座右の一冊となるでしょう。

♣ コーディングを支える技術

難易度 ★★☆



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。そして、今後生まれてくる新しい概念も、よりいつそう理解しやすくなることでしょう。

A.4 アルゴリズムを学ぶ

優れたプログラムを書くためには、アルゴリズムの理解も欠かせません。

♣ アルゴリズム図鑑 絵で見てわかる 26 のアルゴリズム

難易度 ★☆☆



基本的な 26 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 26 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や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

難易度 ★☆☆



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のみ改訂。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。

A.5 Ruby を学ぶ

分かりやすい記法と優れた書き味で人気のプログラミング言語 Ruby のご

紹介です。

♣ 初めてのプログラミング 第2版

難易度 ★☆☆



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ることができます。

プログラミングを学ぶなら、本書は最初の1冊に最適な入門書です。

♣ たのしい Ruby 第6版

難易度 ★☆☆



本書は、今までプログラミングをしたことがない、という方でも Ruby の使いこなし方の一端がつかめるように、ていねいな解説を行っています。プログラムに必要な変数・定数・メソッド・クラス・制御構造といった文法的な説明から、主なクラスの使い方と簡単な応用まで、できるだけわかりやすく説明することを心がけました。

チュートリアル、基礎、クラス、実践とテーマを切り分けて、平易に解説。Ruby の基礎から応用までがわかる一冊。

♣ プロを目指す人のための Ruby 入門

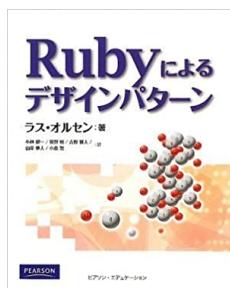
難易度 ★★☆



Ruby の文法をサンプルコードで学び、例題でプログラミングの流れを体験できる解説書です。ほかのプログラミング言語で開発経験のある人が、Ruby を学ぶ際に効率的に学べる内容を詰め込みました。プログラミング未経験者向けの「変数とは」「配列とは」といったプログラミング基礎知識の説明は最小限にし、そのぶん Ruby の特徴（他プログラミング言語との違い）、Ruby におけるリファクタリングの要点、テスト駆動開発やデバッグのやり方など開発現場で必要になる知識を解説しています。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つた不思議な構文」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。

♣ Ruby によるデザインパターン

難易度 ★★★



スクリプト言語 Ruby の持つ力強さ・エレガントさ・シンプルさを、Gof をはじめ従来の代表的なデザインパターンと結合させ、少ないコードでより高度な効果的ソフトウェアを記述する方法を解説します。Metaprogramming や Rails-based Convention Over Configuration patterns などといった Ruby コミュニティから出てきた新しい革新的なパターンも紹介し、Ruby のプログラミング経験をより深く価値のあるものにしてくれる 1 冊です。

付録 B

付録: 参考リンク集

より深く学びたい時に役立つサイトのご紹介です。

B.1 Mozilla 公式チュートリアル

ウェブ開発チュートリアル^{*1}

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があります。これから始める初心者の方や基礎を勉強中の方、そして Web 開発のベテランの方にとっても、ベストプラクティスを学習するのに役に立つ教材が見つかります。

B.2 タイピング

Typing Club^{*2}

「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です。

プログラミング技術習得への第一歩は、タッチタイピング。円滑に

キーボードから文字入力できるよう、毎日こつこつ練習しましょう。

B.3 プログラミング練習

ドットインストール^{*3}

すべてのレッスンは3分以内の動画で提供されています。すきま時間を利用して無理なく気軽に学べます。

Progate^{*4}

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」を採用しました。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。

paiza^{*5}

ラーニング1本3分の動画と練習問題で効率的に学ぶ、オンラインでプログラミングしながらスキルアップできる、プログラミング入門学習コンテンツです。

コードクロニクル^{*6}

RPGを楽しみながら、プログラミングの世界に触れることができます。

B.4 技術用語

IT用語辞典^{*7}

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ることができます。

Wikipedia^{*8}

インターネット百科事典であり、精度・信頼性に疑問はあるものの有益な記事も多く掲載されています。

B.5 写真素材

Web サイトを作る際の写真素材も多くのサイトから提供されています。

ぱくたそ^{*9}

人気の写真素材を無料でダウンロード「ぱくたそ」

使って楽しい、見て楽しい。高解像度で美しい日本の写真が沢山あります。

pro.foto^{*10},BEIZ images^{*11}, 足成（あしなり）^{*12},Pixabay^{*13}, 写真AC^{*14}もお勧めです。

B.6 ウェブデザイン

洗練されたウェブサイトが多数収集されています。学習の参考にしてください。

WebDesignClip^{*15}

Web デザインの参考となるクオリティーの高い国内の Web デザインギャラリー・クリップ集です。Web 制作において、Web デザイナー・開発の方々などのインスピレーションを刺激するような Web サイトや、アイデア・技術に優れたサイトをクリップしています。

B.7 開発マシン

^{*10} <https://pro-foto.jp>

^{*11} <https://www.beiz.jp>

^{*12} <http://www.ashinari.com>

^{*13} <https://pixabay.com/ja/>

^{*14} <https://www.photo-ac.com>

Air に新しい力をのせて MacBook Air^{*16}

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。

B.8 エディタ

Atom^{*17}

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。始めての方へは設定不要で初日から使え、熟練者には深部まで調整可能な、21世紀のプログラマの為のテキストエディタです。

B.9 ブラウザ

Firefox^{*18}

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。

B.10 英語翻訳

DeepL^{*19}

時には海外のサイトを参考にすることもあるかもしれません。

機械学習 (Deep Learning) 技術を用いた高精度な翻訳サイトです。

Weblia 英語翻訳^{*20}

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。翻訳

精度は少し落ちますが、語学習得にも役立ちます。

B.11 ウェブサイトの公開

Netlify^{*21}

ウェブサイトやウェブアプリケーションをインターネットに公開するためには、どこかのウェブサーバーでホスティング（公開）する必要があります。Netlify は無料で利用できるホスティングサービスです。

GitHub や BitBucket のような Git リポジトリサービスと連携していて、リモートリポジトリに push するだけで自動的にデプロイできるのが特徴です。

B.12 情報技術を夢の架け橋に アトリエ未来

アトリエ未来^{*22}

わたくしのウェブサイトです。是非ご覧ください。

終わりに

本書では、HTML / CSS / JavaScript を学び、ウェブアプリを作成、公開いたしました。

全部で、108行のじやんけんプログラム、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つていけそうですね。

「福祉」。「福」「祉^{*23}」どちらも「めぐみ、さいわい」という意味を持ちます。

「熱き心、^{たくま}逞^{かしいな}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、^{もつ}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^{ことわり}理^りを、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^{かがや}耀^か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つて筆を置きます。

いやさか
彌榮

*23 「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。

じゃんけんゲームを創ろう

初めてのウェブプログラミング

令和三年七月七日 ver 1.0.0

著 者 アトリエ未来

発行者 藤谷 真琴

連絡先 contact@atelier-mirai.net

<https://atelier-mirai.net>

© 令和三年 アトリエ未来

(powered by Re:VIEW Starter)