

# 計算機とプログラミング

— 算盤からiPhoneまでの歩み —

アトリエ未来 [著]



計算機の歴史と仕組みを知り  
ウェブアプリ開発の基礎を学びます  
情報技術に触れる中高生から社会人にお薦めの一冊です

# 計算機とプログラミング

— 算盤から iPhone までの歩み —

[著] アトリエ未来

「技術書典 12 新刊」  
令和四年一月二二日 ver 4.0.0

### **■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、<sup>TM</sup>、<sup>®</sup>、<sup>©</sup>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。基礎編では、「計算機の歴史と仕組み - 暮らしに生きるコンピュータ」として、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紹介しています。

実践編では、「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」として、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、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くください。

## ♣ 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。ウェブサイトを作つてみたい方や、プログラミングに興味があつて、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。

計算機の歴史や歩みに興味がある方への最初の一冊として、あるいは、誰かに教えたいと思っている方が生徒にお渡しするテキストとして、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 上 <sup>\*1</sup>で公開しています。

また、ネット上でも公開 <sup>\*2</sup>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/>

また、表紙絵の女の子は、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、ありがとうございます。

背景の桜と青空の絵は、くらうど職人<sup>b</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://hiromu-hayase.tumblr.com>

<sup>b</sup> [https://www.photo-ac.com/profile/590911#a\\_aid=5dd6cce3c0f54&a.cid=b7c162f5](https://www.photo-ac.com/profile/590911#a_aid=5dd6cce3c0f54&a.cid=b7c162f5)

<sup>\*1</sup> <https://github.com/Atelier-Mirai/janken>

<sup>\*2</sup> <https://atelier-mirai-janken.netlify.app>

## ♣ 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。

---

<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。

(正解はこの本のどこかにあるよ。最後まで読んでね。)



7日分の給料の支払いとして金の延べ棒が1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。

6回鋏を入れて切り取れば、金の延べ棒は7等分されるので日払いできますが、金の延べ棒を6回も切り取るのは大変です。

最小限の切り取り回数で済む2回にしたいと思いますが、どことどこを切り取ればよいでしょうか？

# 目次

<b>始めに</b>	i
<b>第1章 暮らしの中のコンピュータ</b>	1
<b>第2章 コンピュータの歴史</b>	3
2.1 さまざまな計算 - 历・税・工事 - . . . . .	3
2.2 算盤から現代のコンピュータまでの歩み . . . . .	6
<b>第3章 コンピュータを創った人々</b>	13
<b>第4章 コンピュータの仕組み</b>	17
4.1 ハードウェア . . . . .	17
4.2 ソフトウェア . . . . .	19
<b>第5章 二進数の話</b>	21
5.1 二進数と十進数、十六進数 . . . . .	21
5.2 コンピュータでのデータ表現 . . . . .	22
5.3 単位の話 . . . . .	26
5.4 計算機理論入門 . . . . .	26
<b>第6章 アルゴリズムとは</b>	29
6.1 アルゴリズムとは . . . . .	29
6.2 アルゴリズムと効率性 . . . . .	31
<b>第7章 プログラムとは</b>	33
7.1 「プログラム」 = 「コンピュータへの指示書」 . . . . .	33
7.2 プログラミング言語の種類 . . . . .	33
7.3 学校教育でのプログラミング . . . . .	35
<b>第8章 ウェブの歴史と技術</b>	37
8.1 ウェブサイトの発祥 . . . . .	37
8.2 HTML とは . . . . .	38
8.3 CSS とは . . . . .	39
8.4 JavaScript とは . . . . .	39
<b>第9章 開発環境の準備</b>	43
9.1 21世紀の高性能エディタ Atom . . . . .	43
9.2 セキュリティ重視のブラウザ Firefox . . . . .	46
9.3 基本的な開発方法 . . . . .	46

---

<b>第 10 章 初めての HTML</b>	<b>51</b>
10.1 HTML の文法 . . . . .	52
10.2 じゃんけんゲームの土台を作る . . . . .	54
<b>第 11 章 初めての JavaScript</b>	<b>57</b>
11.1 初めての JavaScript . . . . .	57
11.2 変数 . . . . .	59
11.3 ウェブブラウザと DOM . . . . .	59
11.4 イベントリスナと関数 . . . . .	60
11.5 繰り返し処理 . . . . .	62
11.6 条件分岐 . . . . .	64
11.7 亂数の利用と入れ子になった if 文 . . . . .	66
11.8 定数の利用とリファクタリング . . . . .	68
11.9 じゃんけんのアルゴリズム . . . . .	70
<b>第 12 章 初めての CSS</b>	<b>73</b>
12.1 ユーザインターフェース . . . . .	73
12.2 CSS の基本 . . . . .	78
12.3 CSS グリッドレイアウトとレスポンシブデザイン . . . . .	85
<b>第 13 章 じゃんけんゲームの完成</b>	<b>87</b>
13.1 プレイヤーの手の取得とコンピュータの手の表示 . . . . .	87
13.2 アニメーション機能と勝敗更新機能 . . . . .	90
13.3 じゃんけんプログラム完成 . . . . .	93
<b>付録 A 珠玉の名著のご紹介</b>	<b>97</b>
<b>付録 B 付録: 参考リンク集</b>	<b>101</b>
<b>付録 C HTML / CSS / JavaScript 簡易まとめ</b>	<b>103</b>
C.1 HTML 簡易まとめ . . . . .	103
C.2 CSS 簡易まとめ . . . . .	107
C.3 JavaScript 簡易まとめ *3 . . . . .	111
<b>終わりに</b>	<b>117</b>

# 第 1 章

## 暮らしの中のコンピュータ

コンピュータが歩んできた歴史を巡る旅の始まりは、身の回りで使われているコンピュータを発見するところから始まります。様々などころで使われているコンピュータ。皆さんも一緒に探求していきましょう。



▲ 図 1.1: 暮らしの中に生きるコンピュータ

### コンピュータ (Mac / iPad / iPhone)

なんといってもコンピュータの代表です。いろいろなウェブサイトを見たり、書類作成などお仕事に活用したり、そして「プログラミング」など。iPad でお絵描きや読書、iPhone で連絡を取り合ったり、写真や音楽、ゲームなどを楽しむことも出来ます。ロケット打ち上げ、宇宙観測や天気予報に活かしたり、都市計画から住宅設計、工場で車や船、飛行機など様々なものを清算したり、音楽や映画、アニメーションなど芸術の分野に至るまで、様々な分野でコンピュータは使われています。

### 衣服

「天衣無縫」 - 「天人や天女の着物には縫い目がないという意から、詩文などが、よけいな修飾がなく、自然でわざとらしくなく完成されていること。また、人柄が純真で素直で、まったく嫌みがないさま。物事が完全無欠であることの形容 (goo 辞書)」

衣食住は、人が生きる上で生活の基盤となる要素です。大麻、木綿、絹糸、古の昔から日本人の身を纏う衣服に用いられてきました。縦糸と横糸を編んで一枚の布にして、布から切り取つて縫いあわせて一着の衣服を仕立てていきます。コンピュータの活用により編み機から直接衣服

いにしえ

を生み出すことが出来るようになりました。縫い目がないから着心地も良く、布の無駄もないとの特徴を持ちます。夢であった「天衣無縫」が顕現した瞬間です。

#### 炊飯器

美味しい御飯を炊き上げてくれる炊飯器。「初めちょろちょろ、中ぱつぱ、赤子泣いてもふた取るな」と、朝早くから起きて竈で御飯を炊くのはなかなか難しいものでした。キャンプで飯盒炊飯した経験をお持ちの方もいらっしゃると思いますが、火加減が難しく焦げになってしまったり芯が残ってしまったこともあるかと思います。火力の調整をしたり、毎朝御飯が炊き上がるためのタイマー機能など、小さなコンピュータ（マイコン）を組み込むことで、美味しい御飯を頂けるようになりました。

#### エアコン

部屋にあるエアコン。暑い夏には涼風を、寒い冬には暖風を送り、快適に過ごせるよう室温を調整しています。部屋の温度を感じるセンサー機能、設定温度を保つように計算する演算機能、実際に風を送り出す送風機能から成り立っています。

#### 給湯器

台所や浴室の給湯器もコンピュータの産物です。昔の人のお風呂として思い浮かぶのは、五右衛門風呂でしょうか。川や井戸から水を汲んできて、薪でお湯を沸かして、湯加減を確かめてと、お風呂は贅沢品でしたので、庶民は銭湯へ通いました。今ではとても簡単に給湯器を設定すると、浴槽一杯に給湯してくれ、温かいシャワーも使うことが出来ます。

#### 信号機

会社や学校へ行く途中に見かける信号機。これにもコンピュータが使われています。赤信号、青信号と、色を変えるのはもちろん、道路の状況に応じて、近くの信号機と連携、青信号の長さを調整することで渋滞緩和を図るなどしています。

#### バス

通勤通学に電車やバスを利用する方も多いでしょう。「さくら高校 行」と電光掲示板で行先表示したり、現金の他、ICカードを繋ぐことで乗車代金を払うことが出来ます。車両自体のガソリンや電気自動車の出力を制御する際や、バス停に「停留所を発車しました」と運行状況表示するなどの用途にも使われています。

こうして見てただけでも身の回りのいろいろなところにコンピュータが使われていますことが分かります。他にはどこに使われているでしょうか。家の中で、お店で、学校や会社でなど、探してみましょう。

# 第2章

## コンピュータの歴史

昔から人は暦や税、土木工事など、様々な計算を行ってきました。この章では、太古から現代に至るまでの計算機の歴史を振り返っていきます。

「コンピュータ」とは何でしょうか。語源を訪ねてみましょう。<sup>\*1</sup>

計算手（けいさんしゅ、英：computer, human computer）とは、電子計算機が実用化される以前の時代において、研究機関や企業などで数学的な計算を担当していた人間のことである。現在では「コンピュータ」と言えば電子計算機を指すが、当時は "computer" という語の成り立ちが表す通り「計算する人間」のことであった。

昔から人は暦や税、土木工事など、様々な計算を行ってきました。そうした計算をより容易にするために、暗算や筆算をしたり、算盤や計算尺を発明しました。そしてたくさんの歯車を組み合わせた機械式計算機が生まれ、電気にに関する理解が深まり、「電子計算機」＝「コンピュータ」が現れました。

現代のコンピュータの歩みについては、[コンピュータ博物館<sup>\\*2</sup>](#)にも豊富な資料がございますので、ご覧ください。[ウィキペディア<sup>\\*3</sup>](#)からの引用を中心のご紹介していきます。

### 2.1 さまざまな計算 - 暦・税・工事 -

#### ♣ 暦について

「こよみ」の語源は、江戸時代の谷川士清の『和訓栢』では「日読み」（かよみ）が定説となっており、一日・二日…と正しく数えることを意味する。ほかに、本居宣長の「一日一日とつぎつぎと来歴（きふ）るを数へゆく由（よし）の名」、新井白石は「古語にコといひしには、詳細の義あり、ヨミとは数をかぞふる事をいひけり」などの定義がある。

古代エジプトにおいて、ナイル川の氾濫の時期に周期性があることに気づいたのが暦の始まりといわれている（シリウス暦）。人類が農耕を行うようになると、適切な農作業の時期を知るために暦は重要なものとなっていました。まず昼夜の周期（地球の自転）が日となり、月の満ち欠けの周期（月の公転）が月に、季節の周期（地球の公転）が年となった。このように暦法は天体運動の周期性に基づいていることから、その観測と周期性の研究が重要であり、これが天文学の基礎となった。

\*1 出典：[ウィキペディア](#)

\*2 <https://museum.ipsj.or.jp>

\*3 <https://ja.wikipedia.org/wiki/>

一方で、石器時代の35000年前に暦を創つたらしいとの意見もある。紀元前3000年頃のシュメール文明では、季節が冬と夏の2つで、1か月29日か30日の12か月の比較的簡単な暦を作り上げたといわれている。

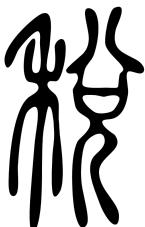
何を基準として一年を定めるか、閏（閏日・閏月）をどのようにして決めるかなどにより、太陽を基準とした太陽暦、月の運行を基準とした太陰暦、両者を折衷した太陰太陽暦など、さまざまな暦法が作られた。

太陰暦は月の運行を意識した暦で、「何日」と月のみかけの形が一致する。したがって月が出てさえいれば、その日が何日であるか暦がなくてもわかる。深夜に月の明かりを頼りとして活動をする場合には、月のみかけの形がわかると都合がよい。また、潮の満ち引きは月の位置と密接な関係があるため、漁業などにも役に立つ。

ただし、完全な太陰暦においては一年が約354日であり、太陽暦に比べ11日短くなるため、3年間で33日（約一ヶ月）ずれてしまい、実際の季節と大きく食い違ってしまう。このため、これを調整する方法として太陽暦を補助的に使用し、閏月の挿入により実際の季節と暦とのズレを修正する方法がとられるようになった。これが太陰太陽暦である。

それに対して、太陽暦は月の形とは関係なく暦が作られている。従って暦だけではその日の月の形は分からぬ。しかし、太陽の運行と暦の月日が一致している為、草花の開花、鳥の渡り等、同じ月日に同じ季節現象を期待でき、その早遅を観察することで、その年の寒暖の傾向を知ることができる。このことは農業や漁業にとって極めて大切である。<sup>\*4</sup>

## ♣ 税について



古代漢字研究の第一人者として知られる白川静さんは、字書三部作『字統』『字訓』『字通』を出版されました。左は、その中から「税」の篆文体です。神に稻穂を捧げる、そういう意味でしょうか。

最近は税の計算をする際にも、コンピュータが用いられ、手で計算することはほとんど無くなりましたが、いかほど納めるべきか、計算するのは大変であったことが想像できます。

国税庁の税の学習コーナー<sup>\*5</sup>より、その税の歴史をご紹介いたします。

### 飛鳥時代

飛鳥時代に行われた大化の改新では、公地公民など、新しい政治の方針が示されました。大宝律令では、租・庸・調という税や労役をかける税のしくみができました。租は男女の農民に課税され、税率は収穫の約3%でした。庸は都での労働、又は布を納める税、調は布や絹などの諸国の特産物を納める税だったようです。

### 奈良・平安・鎌倉・室町時代

奈良時代には、墾田永年私財法が制定され、土地の私有化へと展開していきました。また、平安時代には大きな寺社や貴族の莊園が各地にでき、農民は莊園領主に年貢や公事、夫役などを納めました。鎌倉時代は守護、地頭や莊園領主のもとで経済が発達します。室町時代には、税の中心は年貢でしたが、商業活動の発達により商工業者に対しても税が課せられ、街道に設けられた関所では、関銭などが課せられました。

---

<sup>\*4</sup> 出典: ウィキペディア

<sup>\*5</sup> <https://www.nta.go.jp/taxes/kids/hatten/page16.htm>

### 安土桃山・江戸時代

全国統一を行った豊臣秀吉は、太閤検地を行い、農地の面積や収穫高などを調べて年貢を納めるようにしました。当時の税率は、二公一民といい、収穫の三分の二を年貢として納める厳しいものでした。江戸時代には、田畠に課税される年貢が中心で米などを納めたそうです。また、商工業者に対する税も、運上金・冥加金として納められました。

### 明治時代

明治政府は歳入の安定を図るため、地租改正を実施しました。土地の地価の3%を地租として貨幣で納めさせたそうです。また所得税や法人税が導入されたのもこの頃です。ちなみに所得税は、所得金額300円以上の所得者に課税されるものでした。

### 大正・昭和時代

大正時代から昭和初期にかけては、戦費調達のため、増税が続きました。一方で、現在ある税のしくみができ始めたのもこの頃です。昭和15年に源泉徴収制度が採用されました。昭和21年には日本国憲法が公布され、教育、勤労にならぶ三大義務の一つとして「納税の義務」が定められました。また翌年には、納税者が自主的に自分の所得や税額を計算して申告・納税する申告納税制度が導入され、昭和25年にはシャウプ勧告に基づき税制改革が行われ、今日においても税制度の基盤となっています。

### 平成時代

平成元年に、商品の販売やサービスの提供に対して3%の税金を納める消費税が導入されました。消費税は平成9年には5%、平成26年から8%、令和元年からは10%に増税され、税収の三分の一を占めるまでに成りました。<sup>\*6</sup>

## ♣ 土木工事

古来より様々な土木工事が行われてきました。大林組により、現代の技術を用いてピラミッドを建築するにはと試算した記事がございますので、引用してご紹介いたします。

### クフ王型大ピラミッド建設計画 — 現代に巨大ピラミッドを建設すると



紀元前2500年頃に建てられたエジプトのクフ王の大ピラミッドは、何の為に、そしてどのように創られたのか、今もよく分かっていない。その異形と驚くべき完成度の不思議に近づこうと、建設技術者の立場からアプローチを試みた。

### クフ王型大ピラミッドとは

その底辺は一辺が約230mの正方形で、それぞれの面は正確に東西南北の方位に対面している。また、四つの角は完全な直角だ。方位や角度を正確に得るのは現代でもなかなか困難で、どうやって算出したものかいまだに定説はない。また、一個平均2.5~7tの石灰石の切り石230万個を、正確に四角垂を形成するよう、高さ147mに積み上げており、どうやってこれだけ正確に積み上げることができたのか、専門家ほど驚く正確な完成度である。

<sup>\*6</sup> 大型間接税は導入しないとの公約の下、竹下内閣により導入された消費税は日本の衰退を齎しました。エンタメの力で世の中を盛り上げたい！ 高校生たちがこの国の未来のこと、自分たちのことを考え、少しでも自分たちでできることを見つけ出していく青春社会派物語をご覧ください。[\(https://camp-fire.jp/projects/view/396273\)](https://camp-fire.jp/projects/view/396273)

### 建造の前提条件

立地は、エジプトのギザ市の同じ場所とする。そして、建設技術は現代の最新技術を用い、主要な工事機械は日本から運ぶ。設計・施工の管理はプロジェクトチームが行い、施工作業員は現地に求める。

石材は、古代にエジプト人が用いた石灰岩を当時と同じ近くの採石場から得る。また玄室「王の間」に用いる重量 500t の花崗岩は近辺に無いため、100km 以上上流から別個に運びこむ。

最初の作業が、この石材の量と数を計算することであり、次いでこれを採石、検査、運搬、さらに据え付けという 4 つの工程に分ける。とくに配慮が必要なのが、四角垂の構造体は、工事がすすむにつれて、高度が上るに従い工事量が激減、作業場所も狭くなつていき、クレーンなどは使えない。高さが 60m（全高の 10 分の 4）の段階で、全石数の 80 % を施工することになり、難しい課題となる。そこで、かつてと同様に土盛りをし運搬用の巨大斜路を造成して、石材を搬入することとした。

### 工事の進捗

大ピラミッドの全重量は 580 万 t と試算し、現地の地質は石灰質であり、よく耐えると判断した。基礎地盤はブルドーザで掘削、水平に仕上げ、高さ 50cm の礎石を敷き詰める。その上に本体の石を積み上げていく。高度が上るに従い、土盛り築造した斜路を利用、専用トラックで運びあげる。

### 総工費は 1250 億円

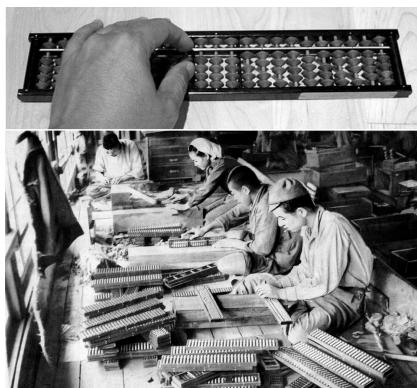
工程の管理とその運営も、精度をきわめたものとなる。そして、工事費見積り総額は 1,250 億円である。工期は 5 年となる。今から 4500 年の昔のエジプト人たちがおこなった同じ工法を、現代の建設費に試算すると、 $20 \text{ 万人} \times 30 \text{ 年} \times 12 \text{ カ月} \times \text{平均 } 6 \text{ 万円/月} = \text{約 } 4 \text{ 兆円}$  となる。現代の世界には、単体でこれほどの金額にのぼるものはまだない。

### すばらしかった人びと

大ナイルは毎年夏から秋にかけて規則正しく広域にわたる氾濫を繰り返した。この間の農閑期に農民の大動員を行い、食糧を支給し富の再分配と救済となった。また、大量の人びとを動員する工事は、明確で精密な計画と組織を有した機構の充実ぶりも物語っている。だからこそ、あの広大なナイル流域の水利・灌漑はじめ膨大で複雑で困難な整備と維持を可能としていた。つまり、このピラミッドの巨大さは、経済力と技術力、数多の労働力が組織化されていた姿も示している。<sup>\*7</sup>

## 2.2

### そろばん 算盤から現代のコンピュータまでの歩み



#### 算盤 (そろばん)

算盤とは、物体に状態で数を記憶させるため、串で刺した珠の位置などで数を表現し、計算の助けとする道具である。ひとつ串（ひと筋の串）が数の「ひと桁」に対応しており、珠を指で上下に移動させることで各数字の表現や変更を行う。加・減・乗・除などの計算が行える。

珠算は整数や小数を扱う場合には桁数が多くても敏速かつ正確に計算できる長所があり、四則計算などは簡易な加減法九九の適用によって計算できる。

<sup>\*7</sup> 現代工法による クフ王型大ピラミッド建設計画 ([https://www.obayashi.co.jp/kikan\\_obayashi/detail/kikan\\_01\\_idea.html](https://www.obayashi.co.jp/kikan_obayashi/detail/kikan_01_idea.html))

起源についてはアステカ起源説、アラブ起源説、バビロニア起源説、中国起源説など諸説ある。メソポタミアなどでは砂の絵に線を引き、そこに石を置いて計算を行っていた「砂算盤」の痕跡がある。同様のものはギリシャなどにも残るが、ギリシャ時代には砂だけでなくテーブルの上などにも置いていた。このテーブルを「アバクス (abacus)」と言う。ローマ時代に持ち運びができるように小さな板に溝を作りその溝に珠を置く溝算盤が発明され、中東を経て中国に伝わり現在の原型となった。現存する最古の算盤はギリシアのサラミス島で発見された「サラミスの算盤」で、紀元前300年頃のものである。

江戸時代には「読み書き算盤」といわれ寺子屋や私塾などで実用的な算術が教えられており、昭和中期までは、事務職や経理職に就くには珠算が必須条件だった。なお、この時代、手動式アナログ計算器としては計算尺があり、理系の人間はそちらも使いこなした。

日本国内では兵庫県小野市と島根県奥出雲町が二大産地であり、小野市の算盤は播州算盤、奥出雲町の算盤は雲州算盤として知られ、正月の「はじき初め」や、八月八日はパチパチと算盤の珠をはじく音に通じるため「算盤の日」となっている。



### アンティキティラ島の機械

アンティキティラ島近海の沈没船から発見された古代ギリシア時代の遺物で、製作時期は紀元前3世紀～紀元前1世紀中頃と推定されている。天体運行を計算するため作られた歯車式機械であると推定されている。

非常に精巧な構造から、古代の著名な科学者が作成にかかわった可能性が取り立たされる。例えば、天文學と数学の中心として知られたロドス島の天文学者ヒッパルコスやストア哲学者ポセイドニオスなどである。

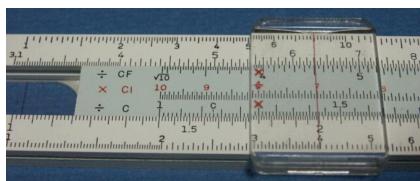
機械の概念は古代コリントスの植民地に起源をたどることができるとし、アルキメデスとの関係も示唆されている。多くの歯車が組み合わさっていることから、最古のアナログ計算機と呼ぶ人もいる。

アンティキティラ島の機械は最古の複雑な科学計算機と考えられており、縮小化と部品の複雑さは、18世紀の時計と比較しても遜色ない程である。30以上の歯車を持ち、クランクを回転させると機構が太陽、月やその他の天体の位置を計算する。

装置には主な表示盤が前面に一つ、背面に二つあり、前面の表示盤には2つの同心円状の目盛が刻まれている。外側のリングはソティス周期に基づく365日のエジプト式カレンダーまたはソティス年を表示する。内側の目盛りにはギリシャの黄道十二星座の記号が刻まれていて角度により区切られている。この暦ダイヤルを4年に1回1日分戻すことにより実際の1太陽年（約365.2422日）との誤差を補正することができる。注目すべきは、最古のうるう年を含んだ暦であるユリウス暦の成立は、この機械が作られた100年後の紀元前46年だということである。

前面の表示盤は少なくとも3つの針を持ち、1つは日付、残りは太陽と月の位置を示していた。月の表示針を動かして月軌道の真近点角が求められる。太陽についても同様の機能があると想像されるが、該当する歯車は発見されておらず定かではない。前面の表示盤には第二の機能として球体模型を使った月相表示機能がある。

機械に刻まれた文字には火星と水星に関する記述があり、製作者には確かにそれらの惑星の位置を示す歯車を盛り込める十分な技術があったように思われる。この機械は当時のギリシャ人が知り得た5惑星全ての位置を表せたとも推測されている。



### 計算尺

対数の原理を利用したアナログ式計算用具で、乗除算および三角関数、対数、平方根、立方根などの計算に用いられる。計算尺は様々な関数の値の対数を計算し、その比率を目盛として固定尺や滑尺に配置したものである。

対数は1614年にスコットランドのジョン・ネイピアが発表した。その6年後にイギリスのエドムント・ガンターが対数尺を考案した。これは数の対数や三角関数  $\sin$ ,  $\tan$  の対数などを幾何的に配置したものであり、コンパスを利用して2つの目盛の長さの加減をしていた。現在の形式の計算尺、つまり複数の尺をずらして計算をするという形の計算尺を発明したのはオートレッドであり1632年のことである。様々な計算尺が考案され、電卓（電子式卓上計算機）が普及する1980年代まで広く使われた。

高性能の関数電卓が普及するまで計算尺は数理系の研究者にとって必須のアイテムであり、マンハッタン計画やアポロ計画の記録映像などにおいても科学者が現場で用いていた。映画の『アポロ13』でも軌道計算を検算する場面に、『風立ちぬ』でも航空機の設計の場面で登場している。

日本製の計算尺は竹製で狂いが少ないと評価され、戦前は世界シェアの80%を占めた頃もあった。



### シッカートの計算機

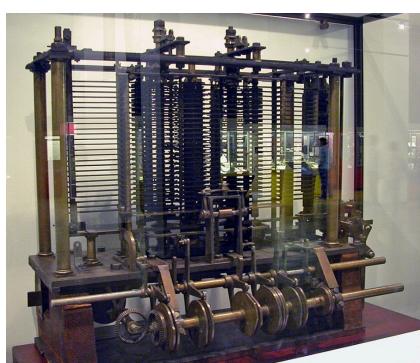
テュービンゲン大学のヘブライ語教授であったヴィルヘルム・シッカートが1623年に発明した機械式計算機である。パスカル、ライプニッツの計算機よりも機能は少ないが、20年先行している。

この計算機は、6桁の加減算およびオーバーフローの検出、複数のネイピアの骨を使った乗算が可能であった。シッカートのヨハネス・ケプラーへの手紙には、天体計算への利用方法が記されている。

「ネイピアの骨」は、対数を利用して「掛け算を足し算だけの計算にする」道具で、ネイピアが科学で扱われる計算の簡略化に尽力した成果として得られた道具である。

ネイピアの死後、ネイピアの骨は様々なに改良されるが、特に1623年のヴィルヘルム・シッカートによる改良が重要である。シッカートは、ネイピアの骨を歯車などを用いて自動化した。シッカートの計算機は、足し算機能も組み込まれており、ダイヤルを回すことにより6桁と1桁の掛け算ができる。

このシッカートの計算機は、世界初の歯車式計算機としても知られ、その後のコンピュータの歴史へ繋がる一步であった。



### バベッジの解析機関 試作品 (1837年)

イギリス人數学者チャールズ・バベッジが設計した、蒸気機関で動くはずだった機械式汎用コンピュータであり、対数や三角関数の数表を作ることに特化した計算機である。資金などの問題があり、この機械は実際には製作されなかったが、1871年の死去直前まで設計が続けられた。論理的に解析機関に匹敵する機能を持つ汎用コンピュータは、時代を下ること百年、1940年代に現実のものとなつた。コンピュータの歴史上、重要なステップを刻んだ一台である。

当時、フランス政府はいくつかの数表を新しい手法で製作していた。数人の数学者が数表の計算方法を決定し、6人ほどでそれを単純な工程に分解して、個々の工程は加算か減算をすればよいだけにする。そして加減算だけを教え込まれた80人の計算手に計算させるのである。これが計算における大量生産的手法の最初の適用例であった。1812年、バベッジは熟練していない計算手を完全に機械に置き換えれば、より素早く正確に数表を作れるというアイデアを思いついた。

解析機関は、制御情報にしたがってオルゴールのようにピンを配置してあって回転、停止、逆回転するドラム群が中心となっている。そして、多くの歯車や力の伝達機構、位置や回転角などで情報を記憶・表示する仕組みなどから構成される、複雑で大きな機械である。蒸気機関を動力として、完成すれば長さ30m、幅10mという、いまの電車1.5両分もの巨大さとなっていたはずである。いわば蒸気機関車ならぬ蒸気機関計算機である。

プログラムとデータの入力は、当時既にジャカード織機のような機械式織機で使われていたパンチカードで供給される予定だった。出力としては印刷原版作成機、曲線プロッターおよびベルを準備していた。演算方式は十進数の固定小数点演算であり、1,000個の50桁の数値を格納できる。演算装置は四則演算が可能で、さらに比較と、オプションで平方根の演算が可能であった。

現代コンピュータのCPUのように命令を持ち、演算装置内部の手続きはパレルと呼ぶ回転するドラムにペグ(釘)を刺することで格納され、それにより複雑な命令を実現した。

バベッジはさらに汎用的な解析機関を構想、1871年に亡くなる直前までその設計を改良し続け、パンチカードでプログラミング可能とした。プログラムをカードで用意することで、最初にプログラムを組めば、それを機械に入れるだけで実行可能である。解析機関はジャカード織機のパンチカードのループで計算機構を制御し、前の計算結果に基づいて次の計算を行なうことができる。

プログラミングは機械語であるが、現在のアセンブリ言語の原型のような記述法が考案されており、逐次制御、条件分岐、繰り返しといった現代のコンピュータのような特徴すら備えていた。チューリング完全を達成していたのではないかと考える者もいる。

エイダ・ラブレスはバベッジのアイデアを完全に理解していた数少ない人物の1人で、単なる計算機に留まらない解析機関の可能性を見出していた。その能力を示すために、ベルヌイ数の数列を計算するプログラムなどを作成したことから、世界初のプログラマと呼ばれている。1979年には、彼女に因んだプログラミング言語Adaが誕生した。



### タイガー計算機(大正13年)

歯車などの機械要素により計算を行う機械式計算器としての代表的存在であり、大正時代に発明されたタイガー計算機は昭和45年まで発売された。

明治45年9月大阪府西成郡豊崎町南浜で営業していた大本鉄鋼所は、好景気の大波に乗って注文に忙殺されていた。見積の為には設計図に沿って原材料費、労務費、その他諸経費等の原価を算定する為「簡単に計算する機械」

を作ろうと、大阪府西成郡鷺洲村海老江に工場を新設、移転すると共に試作品の研究に着手。4年5ヶ月と多額の経費を投じ、大正12年漸く完成した第一号計算器は、発明者大本寅治郎の「寅」をとて「虎印計算器」と命名された。

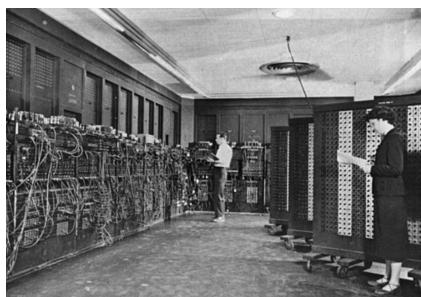
関東大震災後、東京復興の機運がみなぎり大建造物、大工場の建設が始まられた。鉄筋・鉄骨造の建築物や大工事には強度その他の計算を必要とし、しかも算盤や筆算では間に合わず大量の計算器が必要とされることから、広く普及し、電卓の登場と共に、昭和45年、舞台を降りることとなった。<sup>\*8</sup>

<sup>\*8</sup> タイガーハンドクリーク計算器資料館 <https://www.tiger-inc.co.jp/temawashi/temawashi.html> より引用改変

**東京帝国大学 九元連立方程式求解機(昭和19年)**

昭和13年頃、東京帝国大学航空研究所の佐々木達治郎を中心として、機械式計算機の開発が進んでいた。MITのWilburの作った連立方程式の求解機の情報に基づき、志賀亮と三井田純一が製作を担当した。昭和19年頃に完成、9個の変数を持つ連立方程式を解く、我が国初のアナログ計算機となった。

九元連立一次方程式は、一般に9個の未知数と1個の定数を含んだ9個の一次方程式からなっている。本機は鉄のフレームに角度を変えられる真鍮のバーが取り付けられており、その角度が各未知数に対応し、バー上のブーリーにかけられた鉄のテープの長さが各方程式を表している。バーを動かし、テープの長さを読み取ることにより方程式の解を求めることができた。<sup>\*9</sup>

**米国陸軍 ENIAC(昭和21年)**

米国ペンシルベニア大学で開発された黎明期の電子計算機。パッチパネルによるプログラミングは煩雑ではあるが、米陸軍による砲撃射表の計算、マンハッタン計画、円周率計算など、汎用的な計算問題を求解できた。17468本の真空管で作られていて、幅24m、高さ2.5m、奥行き0.9m、総重量30トンと大掛かりな装置であった。消費電力150kW、開発費49万ドルであった。

**FACOM100(昭和29年)**

昭和29年に完成したわが国初の実用リレー式自動計算機。昭和26年に東京大学の山下英男教授の指導により、リレーを主体とした統計分類集計機が完成し、昭和28年にリレー式の株式取引高精算用計算機を試作した。その後、その開発経験を活かし、自由にプログラミングができる本格的なリレー式自動計算機 FACOM100 が完成した。完成後間もなく、日本で初めてノーベル賞を受賞した湯川秀樹博士から極めて複雑な多重積分の計算を依頼された。

FACOM100は人間の手でやったら2年はかかる計算を見事に3日間で答えをだし、湯川秀樹博士は、「これで研究のスピードが飛躍的に進む」と喜んだそうである。計算機の名称は、Fuji Automatic Computerの頭文字をとって「FACOM」と命名されている。<sup>\*10</sup>

**Z80 (昭和51年)**

米国ザイログによって製造された8ビット・マイクロプロセッサ。昭和60年頃までは、パソコンコンピュータのCPUとしてなど、幅広い用途に使用された。現在でも組み込み用途など、目に見えないところで多用されている。

\*9 コンピュータ博物館, <http://museum.ipjs.or.jp/heritage/kyugen.html> より引用改変

\*10 富士通ミュージアム, <https://www.fujitsu.com/jp/about/plus/museum/> より引用改変



### 日本電気 PC-9801(昭和 57 年)

日本電気が発売したパーソナルコンピュータ。キーボードと本体が一体化したデザインで、ハード・ソフトとともに高い機能と完成度を有しており、数多くのソフトウェアや周辺機器が販売された日本のパソコンの代表機種。

PC-9801 は、徐々に広がり始めたビジネス市場に対応するために PC-8001、PC-8801 に代表される NEC の 8 ビットパソコンのソフト・ハード両方の資産を継承しながら、一層の高速処理を実現するために、CPU に NEC 製 16 ビット

マイクロプロセッサ mPD8086 (i8086 互換) 5MHz、画像処理に NEC 開発の LSI であるグラフィック・ディスプレイ・コントローラ mPD7220 を搭載し、日本語処理とカラーグラフィクス表示機能を備え、主記憶容量は最大 640 キロバイト、外部記憶装置のフロッピーディスクは外付けであった。

PC-9801 は、独立系ソフト会社に開発マシンや技術資料を提供することで販売開始初期に各種アプリケーションが揃つたことなどにより、ビジネス市場を中心に広く受け入れられ、ソフトウェア会社、周辺機器会社、出版社、ソフト流通業、販売店、システムハウスなどパソコン産業というべき産業構造の形成に大きく寄与し、98 文化の基礎を築いた。<sup>\*11</sup>

### 【コラム】ムーアの法則

ムーアの法則<sup>\*12</sup>とは、大規模集積回路 (LSI) の製造・生産における長期傾向について論じた指標であり、経験則による将来予測。米インテル社の創業者であるゴードン・ムーアが、集積回路あたりの部品数が毎年 2 倍になると予測した。

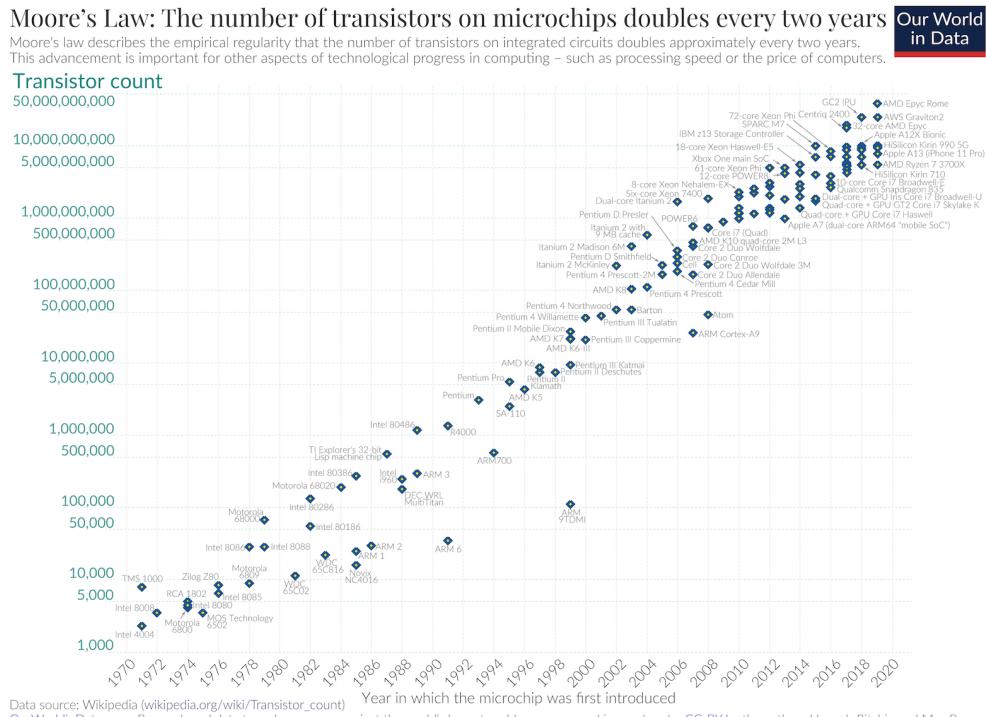
毎年 2 倍はもの凄い増加率で、身近なところに喻えると、赤ちゃんの成長が挙げられるでしょうか。赤ちゃんは、身長 50cm、体重 3kg で産まれますが、一歳を迎えることには、身長 75cm、体重 9kg、三歳では 95cm、14kg になり、身長 115cm、体重 20kg で小学校に入学します。毎年 2 倍とはこれくらいの成長速度です。

対数を使うと簡単に計算できますが、毎日 0.2% ずつ (=月々 6%) の複利で成長すると一年で二倍になります ( $1.06 \times 1.06 \times 1.06 \times \dots$  と、12 回繰り返して下さい)。

『失われた三十年』となった平成はともかく、戦後の『高度成長期』には 18 年に渡り年率約 9% の成長を続けました。<sup>\*13</sup>

コンピュータの成長がどれくらいなのかより実感できるよう、速さに置き換えて感じてみましょう。赤ちゃんのハイハイは、時速 1km と言われています。マラソン選手は、時速 20km で走ります。ジェット機は、時速 900km で空を飛びます。ロケットは、時速 40,000km で、惑星探査に向かいます。数万倍に速くなりました。

<sup>\*11</sup> コンピュータ博物館, <http://museum.ipsj.or.jp/heritage/pc9801.html> より引用改変



▲図2.1: 集積回路のトランジスタ数の増大 \*14

黎明期のコンピュータとして有名な ENIAC(エニアック)は、17468 本の真空管を使い、10 桁の整数の足し算を毎秒 5000 回実行することができました。設置面積は 167 m<sup>2</sup>(約 100 億)、消費電力は 150kW(電気ポット約 150 台分)でした。最新の iPhone は、150 億個のトランジスタを搭載し、一秒間に 15 兆回の計算ができます。

計算を速く行う為に積み重ねられた人の営み。それが今日の豊かな社会を形作りました。

\*9 ムーアの法則: Wikipedia より引用改変

\*10 昭和 30 年の GDP 約 48 兆円、昭和 48 年の GDP 約 229 兆円と 4.8 倍、年率換算 9.1% 成長です。出典: 経済産業省

\*11 トランジスタ数のグラフであり、計算速度のグラフではありませんが、傾向を掴むためのものとして掲載しています。

---

## 第3章

# コンピュータを創った人々

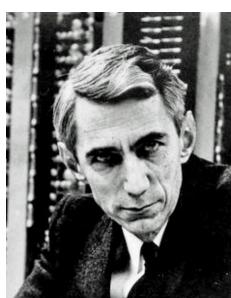
---

今日の情報社会が築かれるにあたっては、多くの方の貢献が欠かせませんでした。数多くの科学者や技術者が関わりましたが、ここでは、ウィキペディアより引用・要約する形で、それぞれの人物の業績を紹介いたします。



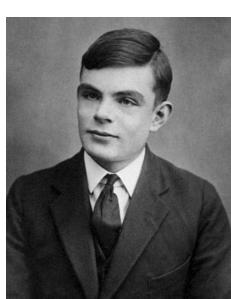
ジョン・フォン・ノイマン

ハンガリー出身の数学学者。ほとんどのコンピュータの動作原理であるプログラム内蔵方式を考案した。原子爆弾（マンハッタン計画）や黎明期の電子計算機 ENIAC（エニアク）の開発でも有名である。ゲーム理論（複数人の意思決定を数学的に研究する学問）の成立に貢献した。企業経営や軍事戦略理論や、将棋やチェスなどの零和ゲームの戦略など、社会に大きな影響を与えた。



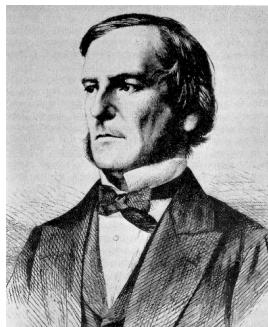
クロード・シャノン

米国の電気工学者、数学学者。情報、通信、暗号、データ圧縮、符号化など今日の情報社会に欠かせない分野を研究し、「情報理論の父」と呼ばれている。マサチューセッツ工科大学で論文を書き、電気回路・電子回路が論理演算に対応することを示した。これにより、デジタル回路・論理回路の概念が確立され、コンピュータの実現に向け、とても大きな一歩となった。情報量の単位「ビット」もシャノンの貢献である。



アラン・チューリング

イギリスの数学者、論理学者、暗号解読者、計算機科学者。「チャーチ=チューリングのテーゼ（提唱）」と計算可能性理論への貢献で広く知られている。アルゴリズム（算法）を実行する機械を形式的に記述した「チューリングマシン」にその名を残す。また「停止性問題の決定不能性」（無限の計算能力を持つコンピュータでも、解けない問題がある）を示した。また、エニグマの暗号解読への貢献や、チューリングテスト、コンピュータチェス、さらに実際面でもコンピュータの誕生に重要な役割を果たし、計算機科学、人工知能の父とも言われている。



### ジョージ・ブール

イギリスの数学者・哲学者。今日のコンピュータ科学の分野の基礎的な理論であるブール代数を確立した。組み合わせ回路（論理回路）はブール代数で表現できる。0と1を電圧の高低に対応させると、デジタル回路の入力と出力をブール論理の式で表現することができる。これにより、ANDゲート、ORゲート、NOTゲートのような基本論理回路や、NANDゲート、NORゲート、XORゲートなどを組み合わせてデジタル回路を構成することができる。



### ニクラウス・ヴィルト

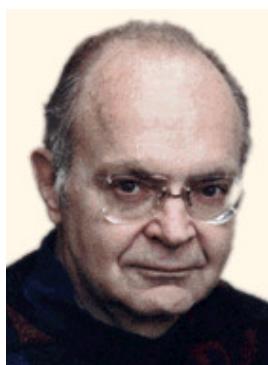
スイスの計算機科学者。プログラミング言語 Pascal、Modula-2などの開発や、ソフトウェア工学分野の開拓的研究で知られる。ヴィルトは、プログラミング言語 ALGOL W、Pascal、Modula、Modula-2 やオペレーティングシステム Oberon の開発などの功績により、ヴィルトは 1984 年にチューリング賞を受賞した。

プログラミングの教育法について書いた記事 Program Development by Stepwise Refinement は、ソフトウェア工学の分野における古典である。1975 年の著作『アルゴリズム+データ構造=プログラム』は広く知られ、今なお価値を失っていない。同書では、コンパイラ設計の説明のために、単純なプログラミング言語 PL/0 を設計。様々な大学のコンパイラ設計の授業で利用された。

### ダナルド・クヌース

アメリカ合衆国の数学者、計算機科学者。スタンフォード大学名誉教授。クヌースによるアルゴリズムに関する著作 The Art of Computer Programming は有名である。アルゴリズム解析と呼ばれる分野を開拓し、計算理論の発展に多大な貢献をしている。その過程で漸近記法 (ランダウ記法, O-記法) で計算量を表すことを一般化させた。

計算機科学への貢献に加え、コンピュータによる組版システム TEX とフォント設計システム METAFONT の開発者でもあり、Computer Modern という書体ファミリも開発した。

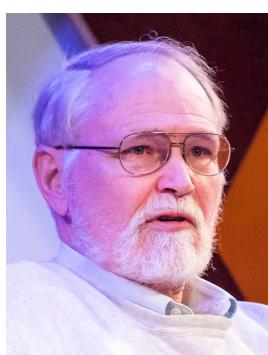


### ブライアン・カーニハン

ブライアン・カーニハンは、ベル研究所に在籍していたカナダ出身の計算機科学者である。C 言語や UNIX の開発者であるデニス・リッチャー、ケン・トンプソンと共に、C 言語および UNIX に対する多くの研究開発結果による貢献で知られている。

デニス・リッチャーと共に著した『プログラミング言語 C』(通称: K&R) は、事実上の規格書として扱われ、現在でも古典的な教科書の一つである。

多くのプログラミング言語入門書で、最初のプログラムとして書かれる Hello world は、彼がベル研究所で書いた B 言語のチュートリアルで初めて使われた。





### デニス・リッチャー

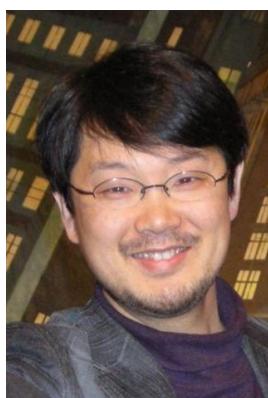
アメリカ合衆国の計算機科学者。1969年頃、同僚のケン・トンプソンと共に、ベル研究所で独自のオペレーティングシステム UNIX を作り始める。この UNIX 上で動作するアプリケーション作成の為に、トンプソンによって B 言語が開発され、リッチャーがこれにデータ型と新しい文法等を追加し C 言語が出来た。1973年にアセンブリ言語で書かれていた UNIX を C 言語で書き換えることに成功した。C 言語の開発は、リッチャーの UNIX への最大の貢献である。

1983年、UNIX 開発の功績により、ケン・トンプソンと共にチューリング賞を受賞している。今日、C 言語は組込システムからスーパーコンピュータまであらゆるプラットフォームで用いられ、彼の業績は偉大である。



### ケン・トンプソン

ケン・トンプソンは、アメリカ合衆国の計算機科学者。長年ベル研究所に勤め、オリジナルの Unix を開発した。また C 言語の前身である B 言語を開発した。2006年から Google で勤務しており、Go を共同開発した。他の主な業績として、正規表現、テキストエディタ QED と ed、UTF-8 コードの定義に加え、チエスの終盤定跡データベースやチェスマシン Belle の開発などコンピュータチエスへの貢献がある。1983年に彼の長年の同僚であるデニス・リッチャーと共にチューリング賞を受賞した。「信用を信頼することについての考察」は、トンプソンハックとして知られる、セキュリティに関する重要な研究成果である。



### まつもとゆきひろ

まつもと ゆきひろは、日本のソフトウェア技術者。株式会社ネットワーク応用通信研究所フェロー、Ruby アソシエーション理事長、松江市名誉市民。通称は Matz。

プログラミング言語「Ruby」の開発者。効率的に記述できるプログラム言語の実現を目指し、平成 5 年から開発を始め、平成 7 年にオブジェクト指向スクリプト言語 Ruby を公開した。Ruby を用いて「Hello, world!」という文字列を出力するために半年を要して苦労したが、「Ruby 言語の開発で飽きたり、辛く感じたりすることはなかった」と語っている。

平成 9 年から松江市に在住し、同市のネットワーク応用通信研究所 (NaCl) にフェローとして勤務している。

Ruby の普及を目的として設立された一般財団法人「Ruby アソシエーション」の理事長も務める。平成 24 年、内閣府から「世界で活躍し『日本』を発信する日本人」の一人に選ばれた。<sup>\*1</sup>

\*1 小さな目標を立て続けたからこそ Ruby はできた まつもとゆきひろ氏が語る、「言語を作りたい」気持ちからの道程 (<http://logmi.jp/tech/articles/325269>)

## ♣ 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・事物には真の名前があり、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だそうです。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。そういえばアニメ化もされた U・K・ル=グウィンの「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、宝石の名前のなかから、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ますが、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あります。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あります。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、私はその名前にもつともこだわります。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つたよう思うからです。

実際、Ruby に対する機能追加の要求に対しても、しばしば「要求は分かった。あれば便利なものも理解できる。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、逆にふざ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。<sup>\*2</sup>

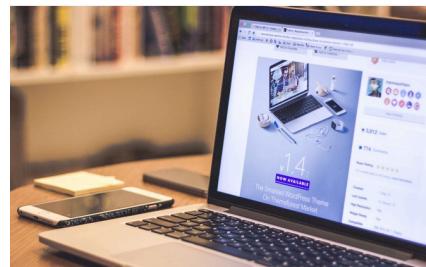
<sup>\*2</sup> 引用：プログラマが知るべき 97 のこと

# 第4章

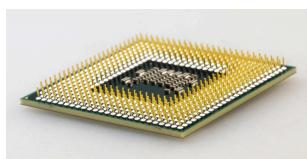
## コンピュータの仕組み

机の上に置かれたコンピュータ。その中身を覗いてみると、演算・制御装置、記憶装置、入力装置、出力装置など、たくさんの中身からできています。IT用語辞典<sup>a</sup>から抜粋しつつ、その詳細を見ていきましょう。

<sup>a</sup> IT用語辞典 <https://e-words.jp/>



### 4.1 ハードウェア



#### CPU

中央処理装置 (Central Processing Unit) とは、コンピュータの主要な構成要素の一つ。他の装置・回路の制御やデータの演算などをを行う装置で、演算装置と制御装置が統合されている。CPUはメインメモリ (RAM) に格納された機械語のプログラムを、バスを通じて

一命令ずつ順番に読み出し (フェッチ)、その内容を解釈して行うべき動作を決定 (デコード) し、内部の回路を駆動して実際に処理を実行する。

一度に4ビットのデータを処理できるCPUを4ビットCPUというように呼び、当初の4ビットから、8, 16, 32ビットと拡張され、現代では64ビットCPUが普及している。

高い周波数信号で動作するCPUほど、多くの処理を行え性能が高い。2GHz(ギガヘルツ: 毎秒10億回)で動作するCPUと1GHzのCPUならば、約2倍の速度差がある。



#### メモリ

メモリとは、記憶、記憶力、回想、追憶、記念などの意味を持つ英単語。IT分野ではコンピュータに内蔵される半導体集積回路 (IC) を利用したデータの記憶装置を指す。コンピュータを構成する装置の一つで、CPUから直接読み書きすることができる記憶装置のことを「主記憶装置」という。

一般に主記憶装置は外部記憶装置よりはるかに高速に動作する装置が用いられるが、単価や装置構成上の制約から少ない搭載容量となっている。このため、コンピュータは起動すると外部記憶から主記憶

に必要なプログラムやデータを読み込んで実行し、不要となったデータは主記憶から消去して新たに必要になったものと入れ替える。永続的な保管が必要なデータは外部記憶へ書き込み保存する。



### ストレージ

ストレージとは、コンピュータの主要な構成要素の一つで、コンピュータが利用するプログラムやデータなどを永続的に記憶する装置。磁気ディスク（ハードディスク HDD）や光学ディスク（CD/DVD/Blu-ray）、フラッシュメモリ（USB メモリ/メモリカード/SSD）、磁気テープなどがある。

同じコンピュータに搭載される装置同士で比較すると、ストレージはメモリに比べて記憶容量が数桁（数十～数千倍）大きく、容量あたりのコストが数桁小さいが、読み書きに要する時間が数桁大きい。

写真は、磁性体を利用した HDD（ハードディスクドライブ）のものであるが、半導体を利用した SSD（ソリッドステートドライブ）がその高速性から普及している。



### マザーボード

マザーボードとは、コンピュータの主要な構成部品の一つで、マイクロプロセッサやメモリなど他の部品を装着し、通電したり相互に通信できるようにする基板のこと。プラスチックなどでできた板状の装置で、表面や内部に各装置を結ぶ配線や制御用の半導体チップ、

電子部品などが高密度に実装されている。プロセッサやメモリモジュール、拡張カードなどを装着するためのスロットやソケットなどの接続部品、電源ユニットからのコードを差し込む電源コネクタ、ストレージ（外部記憶装置）など周辺機器接続用のケーブルを差し込むコネクタなども配置されている。



### 入力装置（キーボード）

正方形や横長の小さなボタンが縦横に整然と並び、文字や記号、コンピュータへの指示などを送信するための入力装置。100 前後のキーが 4～5 段に並び、各キーの上部に入力される文字や機能が記されて

おり、キーを押すと、そのキーが押されたという信号がコンピュータへ送信される。



### 出力装置（ディスプレイ）

ディスプレイとは、表示、展示、陳列、掲示、飾り付け、示すなどの意味を持つ英単語。IT 分野では、コンピュータの出力装置の一つで、画面を発光させて像を映し出す表示装置（display device、ディスプレイ装置）のこと。「モニター」（monitor）とも呼ぶ。

コンピュータの操作画面を映像として電気的に映し出し、処理状況の変化や利用者の操作に即時に反応して表示内容を変化させることができる。

データとして記録された動画像を再生・表示することもできる。ディスプレイ以前に主要な出力装置として利用されていたのは印字装置（プリンタ）であり、状況や操作を表示内容にリアルタイムに反映する特徴は画期的で便利な特性だった。

ディスプレイの画面は格子状に規則正しく並んだ微細な画素（ドット/ピクセル）から成り、その発

光状態を電気的に制御してコンピュータから受信した映像信号を表示する。一つの画素を光の三原色に対応する微細な素子で構成し、カラー表示を行う。



**USB ユニバーサルシリアルバス** USB とは、主にコンピュータと周辺機器を繋ぐ為に用いられるデータ伝送路の標準規格。キーボードやマウス、プリンタ等の接続方式として広く普及している。

初期の規格 (USB 1.1) では 12Mbps(メガビット毎秒)、最新の規格 (USB4) では 40Gbps(ギガビット毎秒) までの通信速度に対応する。当初はキーボードなどの入出力装置から普及が始まったが、

通信速度の向上に伴いネットワークアダプタ (Ethernet アダプタや Wi-Fi アダプタ) や、外部接続の光学ドライブ、ハードディスクなどに利用が広がっていった。手軽なデータの受け渡し手段としてフラッシュメモリを内蔵した USB メモリもよく使われる。

## 4.2 ソフトウェア

コンピュータを構成する電子回路や装置など、物理的実体がある「ハードウェア」に対し、それ自体は形を持たないプログラムや付随するデータなどをソフトウェアという。

コンピュータを動作させる命令の集まりであるコンピュータプログラムを組み合わせ、何らかの機能や目的を果たすようまとめたもので、その役割により、ハードウェアの制御や他のソフトウェアへの基盤的な機能の提供、利用者への基本的な操作手段の提供などを行なう「オペレーティングシステム」(OS : Operating System / 基本ソフト) と、特定の個別的な機能や目的のために作られた「アプリケーションソフト」に大別される。

### ♣ OS (Operating System / 基本ソフト)

機器の基本的な管理や制御のための機能や、多くのソフトウェアが共通して利用する基本的な機能などを実装した、システム全体を管理するソフトウェア。

入出力装置や主記憶装置、外部記憶装置の管理、外部の別の装置やネットワークとのデータ通信の制御などが主な役割で、コンピュータに電源が投入されると最初に起動し、電源が落とされるまで動作し続ける。利用者からの指示に基いて記憶装置内に格納されたソフトウェアを起動したり終了させたりすることができる。

パソコン向けの OS としては、Microsoft 社の Windows や Apple 社の mac OS や、Linux がある。スマートフォンやタブレットでは、Apple 社の iOS や、Google 社の Android OS が普及している。

東京大学名誉教授 坂村健 氏が提唱した TRON<sup>トロン</sup> も、組込機器向けの  $\mu$  ITRON<sup>マイクロアイ stron</sup> として活用されており、任天堂のゲーム機「Nintendo Switch」などに採用されている。<sup>\*1</sup>

### ♣ アプリケーションソフト

OS の機能を利用し、OS の上で動作するソフトウェアである。アプリケーションの開発者は、OS の提供する機能を利用することによって、開発の手間を省くことができ、操作性を統一することができる。また、ハードウェアの仕様の細かな違いは OS が吸収してくれるため、ある OS 向けに開発された

\*1 「Nintendo Switch」が  $\mu$  ITRON4.0 仕様準拠リアルタイム OS を採用 (<https://monoist.itmedia.co.jp/mn/articles/1707/07/news029.html>)

ソフトウェアは、基本的にはそのOSが動作するどんなコンピュータでも利用できる。

用途や目的に応じて多種多様なアプリケーションソフトがある。一例を挙げると、ワープロや表計算、画像閲覧・編集、動画・音楽再生、ゲーム、Webブラウザ、電子メール、カレンダー・スケジュール管理、電卓、カメラ撮影、地図閲覧、プレゼンテーションやデータベース、財務会計、人事管理、在庫管理、プロジェクト管理、文書管理などなどである。

利用者が配布・販売パッケージ入手・購入してOSに組み込む作業「インストール」を行うことで使用可能となる。

### 【コラム】単位の話

$1,000\text{ m} = 1\text{ km}$ ,  $100\text{ a} = 1\text{ ha}$ ,  $1\text{ dL} = 1 / 10\text{ L}$ ,  $1\text{ cm} = 1 / 100\text{ m}$ ,  $1\text{ mm} = 1 / 1,000\text{ m}$ などがお馴染みです。

コンピュータでは大きな数、小さな数を良く使いますので、そのための記号を紹介します。

国際単位系(SI単位系)での接頭辞一覧

接頭辞	接頭辞	記号	乗数
ヨタ	yota	Y	$10\text{ の }24\text{ 乗}=1,000,000,000,000,000,000,000,000,000\text{ 倍}$
ゼタ	zeta	Z	$10\text{ の }21\text{ 乗}=1,000,000,000,000,000,000,000,000\text{ 倍}$
エクサ	exa	E	$10\text{ の }18\text{ 乗}=1,000,000,000,000,000,000\text{ 倍}$
ペタ	peta	P	$10\text{ の }15\text{ 乗}=1,000,000,000,000,000\text{ 倍}$
テラ	tera	T	$10\text{ の }12\text{ 乗}=1,000,000,000,000\text{ 倍}$
ギガ	giga	G	$10\text{ の }9\text{ 乗}=1,000,000,000\text{ 倍}$
メガ	mega	M	$10\text{ の }6\text{ 乗}=1,000,000\text{ 倍}$
キロ	kilo	k	$10\text{ の }3\text{ 乗}=1,000\text{ 倍}$
ヘクト	hecto	h	$10\text{ の }2\text{ 乗}=100\text{ 倍}$
デカ	deca	da	$10\text{ の }1\text{ 乗}=10\text{ 倍}$
			$10\text{ の }0\text{ 乗}=1\text{ 倍}$
デシ	deci	d	$10\text{ の }-1\text{ 乗}=10\text{ 分の }1$
センチ	centi	c	$10\text{ の }-2\text{ 乗}=100\text{ 分の }1$
ミリ	milli	m	$10\text{ の }-3\text{ 乗}=1,000\text{ 分の }1$
マイクロ	micro	$\mu$	$10\text{ の }-6\text{ 乗}=1,000,000\text{ 分の }1$
ナノ	nano	n	$10\text{ の }-9\text{ 乗}=1,000,000,000\text{ 分の }1$
ピコ	pico	p	$10\text{ の }-12\text{ 乗}=1,000,000,000,000\text{ 分の }1$
フェムト	femto	f	$10\text{ の }-15\text{ 乗}=1,000,000,000,000,000\text{ 分の }1$
アト	atto	a	$10\text{ の }-18\text{ 乗}=1,000,000,000,000,000,000\text{ 分の }1$
ゼプト	zepto	z	$10\text{ の }-21\text{ 乗}=1,000,000,000,000,000,000,000\text{ 分の }1$
ヨクト	yocto	y	$10\text{ の }-24\text{ 乗}=1,000,000,000,000,000,000,000,000\text{ 分の }1$

# 第 5 章

## 二進数の話

コンピュータは、「0」と「1」の二つの値を用いる二進数で動いています。黎明期には十進数を用いた計算機もありましたが、実行速度や作成費用などから二進法を用いるものが主流となりました。

今日のコンピュータの基礎となる二進数と、関連して様々な情報表現について紹介します。

また、より深く計算機理論を学びたい方へ向けて、デジタル回路のご紹介と、富山大学幸山教授が書かれた「計算機理論入門」への参照を挙げました。

### 5.1 二進数と十進数、十六進数

十進数	二進数	十六進数
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

十進数は、わたくしたちが日常使っている位取り記法で、とても馴染みがあります。

0、1、2、…、8、9と大きくなっています、「一の位」が「十」集まる上位に「進」みます。

10、11、…、99と大きくなっています、「十の位」が「十」集まる上位に「進」みます。

100、101、…、999と大きくなっています、「百の位」が「十」集まる上位に「進」みます。

「十」集まって一つ上の位に「進」むたびに、その桁の重みが十倍になっていくのが、「十進数」の特徴です。例えば、「9 4 3」という数なら、百を9つと、十を4つ、一を3つ併せた数のことです。

二進数でも同様ですので、見てていきましょう。

0、1と大きくなっています、「一の位」が「二」集まる上位に「進」みます。

10、11と大きくなっています、「二の位」が「二」集まる上位に「進」みます。

100、101～111と大きくなり、「四の位」が「二」集まる上位に「進」みます。

「二」集まって一つ上の位に「進」むたびに、その桁の重みが二倍になっていくのが、「二進数」の特徴です。例えば、「1011」という数なら、八を1つと、二を1つ、一を1つ併せた数のこと、つまり、じゅういちのことです。

二進数では、すぐに桁数が増えていくので、四桁をひとまとめにした、十六進数もよく

使われます。十六進数では、10~15までの数を一桁で表せるよう記号を準備する必要があります。黎明期には様々な記号が提案されましたが、今日ではA~Fまでのアルファベットの使用が定着しました。

\*1

## 5.2 コンピュータでのデータ表現

### ♣ 文字

コンピュータで文字を表現するにはどのようにすれば良いでしょうか？

文 字	10	16	文 字	10	16	文 字	10	16									
進 進			進 進			進 進			進 進			進 進			進 進		
NUL	0	00	DLE	16	10	SP	32	20	0	48	30	@	64	40	P	80	50
SOH	1	01	DC1	17	11	!	33	21	1	49	31	A	65	41	Q	81	51
STX	2	02	DC2	18	12	"	34	22	2	50	32	B	66	42	R	82	52
ETX	3	03	DC3	19	13	#	35	23	3	51	33	C	67	43	S	83	53
EOT	4	04	DC4	20	14	\$	36	24	4	52	34	D	68	44	T	84	54
ENQ	5	05	NAK	21	15	%	37	25	5	53	35	E	69	45	U	85	55
ACK	6	06	SYN	22	16	&	38	26	6	54	36	F	70	46	V	86	56
BEL	7	07	ETB	23	17	'	39	27	7	55	37	G	71	47	W	87	57
BS	8	08	CAN	24	18	(	40	28	8	56	38	H	72	48	X	88	58
HT	9	09	EM	25	19	)	41	29	9	57	39	I	73	49	Y	89	59
LF*	10	0a	SUB	26	1a	*	42	2a	:	58	3a	J	74	4a	Z	90	5a
VT	11	0b	ESC	27	1b	+	43	2b	;	59	3b	K	75	4b	[	91	5b
FF*	12	0c	FS	28	1c	,	44	2c	<	60	3c	L	76	4c	¥	92	5c
CR	13	0d	GS	29	1d	-	45	2d	=	61	3d	M	77	4d	]	93	5d
SO	14	0e	RS	30	1e	.	46	2e	>	62	3e	N	78	4e	^	94	5e
SI	15	0f	US	31	1f	/	47	2f	?	63	3f	O	79	4f	_	95	5f
												o	111	6f	DEL	127	7f

▲図5.1: ASCIIコード表

ASCIIコードと呼ばれる表で、アルファベット「A」には十進数の「65」十六進数の「41」が、「B」には十進数の「66」十六進数の「42」が対応しています。

コンピュータは西洋で生まれましたので、アルファベット(A - Z, a - z)、数字(0 - 9)や、記号(@, #, !など)など約100種類といくつかの制御用のコードを加えて、番号付けした、今日でも現役で使われている表です。

### ♣ 画像

コンピュータで画像を表現するにはどのようにしたら良いのでしょうか？

今日、デジタルカメラで写真を撮影することはとても一般的になりました。その嚆矢となったのが、未来技術遺産にも登録された、カシオ QV-10 \*2です。

\*1 卷末の参考文献「プログラマの数学」にも位取りを始めとして様々な話題がございます。是非ご一読下さい。

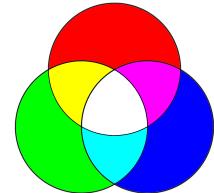
\*2 10年先の写真を見据えて——カシオ「QV-10」(<https://www.itmedia.co.jp/dc/articles/1101/11/news028.h>

QV-10 では、横 320 画素 x 縦 240 画素 (=76,800 画素) の写真を撮ることができました。つまり横 320 枚、縦 240 枚の升目に分割、各升目の色を表現することにより、一枚の画像を表しています。

それでは、色の情報はどのように表せば良いでしょうか。20 年前ものとはなりますが、歴史を感じていただければと、愛知県総合教育センター<sup>\*3</sup>の記事より紹介します。

### 光の三原色

テレビやコンピュータのブラウン管に表示される色は、光の三原色の赤 (R)、緑 (G)、青 (B) を混ぜて、作り出されています（加法混合）。テレビのブラウン管を虫眼鏡などで拡大してみると、赤、緑、青の蛍光体が光っている様子がわかると思います。

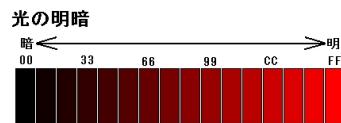


初期のコンピュータ<sup>\*4</sup>は、現在のように多彩な色表示ではなく、白や黒を含めて 8 色表示が可能でした。それは、赤、緑、青の各色について ON/OFF（光らせる／消す）を制御し、その組み合わせで、 $2 \times 2 \times 2 = 8$  通りの色（光の三原色の図参照）を表現しました。

つまり、赤、青、緑が単独で光ることで、(1) 赤、(2) 青、(3) 緑の三色が表現できます。また、赤と緑、緑と青、青と赤の各 2 色が光ると、(4) 黄色、(5) シアン、(6) マゼンタの三色が表現できます。そして、三色が全て光ることで (7) 白を、全て消えることで (8) 黒と、合計 8 色を表現できました。

### 光の明暗

その後、ON / OFF の状態だけではなく、明るさを何段階かに調節した表示が可能になりました。下の図は、赤色を暗い方から明るい方へと 16 段階で表示したものです。



明暗を変化させることにより、赤色で 16 通りの表現ができます。同じように、緑色、青色も 16 段階で表示すると、その各色 16 通りを組み合わせて、 $16 \times 16 \times 16 = 4096$  通りの色が表現できます。

現在では、256 段階 (8 ビット) で明るさが調節できるようになりました。それで、 $256 \times 256 \times 256 = 16,777,216$  通り、約 1677 万色の色が表現できます。人の目で識別できる色数は、数百万～一千万色と言われるので、この約 1677 万色のことを、「フルカラー」と呼ぶこともあります。

### 画像の容量

QV-10 では、横 320 画素 x 縦 240 画素 (=76,800 画素) の写真を撮ることができました。色を各色 256 段階のフルカラーで表現することにすると、一つの画素につき、赤 8 ビット (=1 バイト)、緑 8 ビット (=1 バイト)、青 8 ビット (=1 バイト) と、3 バイト必要ですから、230,400 バイト (=225kB) の容量になります。QV-10 には 2MB のフラッシュメモリが内蔵されており、96 枚までの写真を保存することができました。<sup>\*5</sup>

それから四半世紀経った今日の iPhone には 1200 万画素 ( $4032 \times 3024$ ) のデジカメが搭載されています。一枚の写真の容量は、36,578,304 バイト (=36 メガバイト) になります。画像容量は 158 倍になりました。容量 256GB の iPhone には、 $256\text{GB}/36\text{MB} = 7,282$  枚の写真が保存可能です。画像容量

<sup>\*1</sup> tml) に、愛情溢れる記事が掲載されています。

<sup>\*2</sup> 出典: なぜ、フルカラーで表示できる色は、1677万色なの？ 愛知県総合教育センター (<https://apec.aichi-c.ed.jp/kyouka/jouho/contents/2018/jissyuu/043/gazou.files/iro.htm>)

<sup>\*3</sup> 昭和 54 年に発売された PC-8001 では、テキスト表示 80 行 × 25 行、グラフィック表示 160 × 100 ドット デジタル 8 色 であった。

<sup>\*4</sup> 当時は 24 枚撮りの写真フィルムが発売されていましたが、4 倍もの写真を撮って、その場で確認することができる、とても画期的な製品でした。

が158倍(36MB/225kB)になったにもかかわらず、76倍( $=7282/96$ )もの写真を保存できます。<sup>\*6</sup>

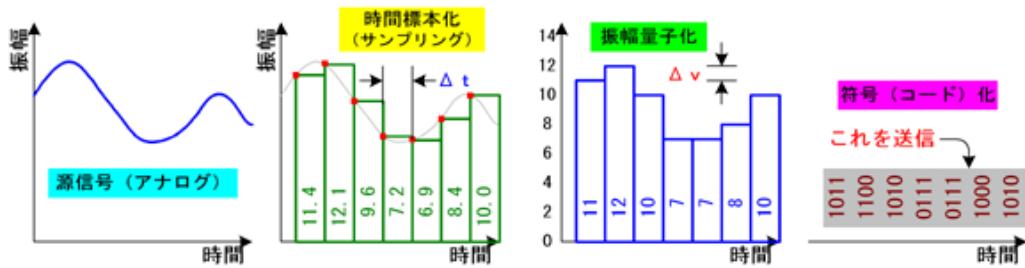
### ♣ 音声

音声をコンピュータで扱うために、様々な方式が考案されています。その中からパルス符号変調(PCM Pulse Code Modulation)を紹介します。

**PCM**とは、音声などのアナログ信号をデジタルデータに変換する方式の一つで、信号の強度を一定周期で標本化(サンプリング)したものです。そのまま保存すれば無圧縮データとなります。

アナログ信号の強度をサンプリング周波数に従って一定間隔で測定し、定められたビット数の範囲で整数値として量子化します。例えば、CDの音声はサンプリング周波数44.1kHz(キロヘルツ)、量子化16ビットで記録されています。これは毎秒44,100回信号を測定し、その強度を65,536(2の16乗)段階の値で表していることを意味しています。

**標本化定理**により、サンプリング周波数の半分の周波数までの信号は再現可能です。人間の可聴音の上限は20kHz程度であることが知られているので、40kHz超のサンプリング周波数を用いれば録音データから概ね自然な音が再生できると言われています。<sup>\*7 \*8</sup>



▲図5.2: パルス符号変調(PCM)方式によるアナログ-デジタル変換

### ♣ 動画

パラパラ漫画を読んだことはありますか。<sup>a</sup>

一枚一枚は人が写っている普通の画像ですが、それらの画像を短い間に切り替えます。すると、「残像効果」<sup>b</sup>「仮現運動」<sup>c</sup>と呼ばれる人間の特性により、あたかも動いているように見えます。

それでは、一秒間に何枚の画像を切り替えると、滑らかな動画に見えるのでしょうか。



<sup>a</sup> 画像出典: IT用語辞典

<sup>b</sup> 出典: Wikipedia 人の視覚で光を見たとき、その光が消えた後も、それまで見ていた光や映像が残って見えるような現象のこと。

<sup>c</sup> 出典: goo国語辞典 実際には運動がないのに、次々と類似の刺激を与えられると、運動があるように感じる現象。映画はこの現象を応用したもの。

\*6 iPhoneのストレージを全て写真に使用できず、データ圧縮等も可能なため、一応の目安です。

\*7 出典: IT用語辞典

\*8 図版出典: パルス変調方式に用いるアナログ信号のデジタル化方式の説明 ([http://www.gxk.jp/elec/musen/lama/H19/html/H1912A09\\_.html](http://www.gxk.jp/elec/musen/lama/H19/html/H1912A09_.html))

一秒間に表示される画像の枚数が少なく、一枚の画像が表示される時間が長いと、動きのカクカクとした不自然で低品質な動画となります。短時間で画像が書き換わると、滑らかで高品質な動画となります。<sup>\*9</sup> 映画では一秒間に 24 コマ、テレビ放送では一秒間に 30 コマの表示が行われています。

以上で、一秒間に 24 コマ、あるいは、30 コマの絵を表示すれば、「動画」になることが分かりました。二時間の映画ではすごくたくさんの画像 ( $172,800$  枚 =  $24 \times 2 \times 60 \times 60$ ) が必要になります。動画の容量が気になりますね。

映画は 24 コマ/秒で、1 枚の標準的な画像は (480 画素 × 720 画素) です。画素一つ一つにつき、色を表現するために 24 ビット (=3 バイト) 必要ですから、一秒間の動画を保存するために約 24MB(メガバイト) (= $480 \times 720 \times 3 \times 24 / 1024 / 1024$ ) が必要になります。DVD 一枚は 4.7GB(ギガバイト)<sup>\*10</sup> の容量がありますから、約 184 秒<sup>\*11</sup> (= $4.25 \times 1024 / 23.73$ ) の動画が記録できます。

### 情報圧縮

DVD 一枚には、184 秒 = 3 分ちょっとの動画が保存できることが分かりました。二時間 120 分の映画の映画を保存するためには、40 枚の DVD が必要となります。しかし、実際には一枚の DVD の中に、二時間 120 分の映画が収録されています。

これを支えているのが、「情報圧縮」技術です。

### 画像・音声圧縮の基本原理と要素技術<sup>\*12</sup>

#### 1. 画像の性質を利用

二値画像: 白黒画素が連続しやすいので、「ランレンジス符号化」<sup>\*13</sup> する。

静止画: 近くの画素は似ているので、「離散コサイン変換 (DCT)」<sup>\*14</sup> する。

動画像: 現画面は前画面に似ているので、「フレーム間予測」<sup>\*15</sup> する。

#### 2. 人間の視聴覚特性を利用

画像: 色信号の劣化には鈍感なので、色情報を「サブ・サンプリング処理」<sup>\*16</sup> により間引く。

音声: 大きな音と同時に存在する小さい音は聞こえにくいという「マスキング効果」<sup>\*17</sup> を利用する。

#### 3. 符号の発生確率の偏りを利用

符号発生確率に差があるので、「可変長符号化」や「算術符号化」<sup>\*18</sup> を行う。

今日では、**Moving Picture Experts Group** が規格した、**MPEG** が広く動画圧縮方式として用いられています。

<sup>\*9</sup> 出典: IT 用語辞典

<sup>\*10</sup> DVD の容量は 4.7GB と表示されて販売されています。このときは  $1\text{GB}=10$  億バイトとして計算しています。コンピュータの内部では、二進数で綺麗に計算できる 2 の 30 乗 =  $1,024 \times 1,024 \times 1,024 = 1,073,741,824$  バイトを、1GB として扱います (1GiB と書くこともあります)。

<sup>\*11</sup> DVD は 4.7GB (=4.37GiB) の容量がありますが、ファイル管理用の領域があるため、利用者が使えるデータ領域は 4.25GiB となります

<sup>\*13</sup> ランレンジス圧縮 (<https://e-words.jp/w/ランレンジス圧縮.html>)

<sup>\*14</sup> 離散コサイン変換 (DCT) (<https://e-words.jp/w/離散コサイン変換.html>)

<sup>\*15</sup> フレーム間予測 (<https://e-words.jp/w/フレーム間予測.html>)

<sup>\*16</sup> サブ・サンプリング (<http://www.hdmi-navi.com/subsampling/>)

<sup>\*17</sup> マスキング効果 (<https://ja.wikipedia.org/wiki/音響心理学#マスキング効果>)

<sup>\*18</sup> 符号割り当て ([https://www.ieice-hbkb.org/files/02/02gun\\_05hen\\_07.pdf](https://www.ieice-hbkb.org/files/02/02gun_05hen_07.pdf))

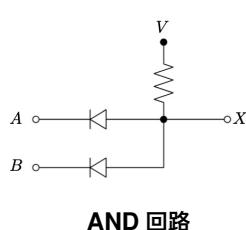
<sup>\*15</sup> 出典: 日本規格協会グループ 画像・映像圧縮 (JPEG/MPEG) マルチメディアの基盤である 画像や映像に関する技術と関連国際規格 ([https://www.jsa.or.jp/datas/media/10000/md\\_2471.pdf](https://www.jsa.or.jp/datas/media/10000/md_2471.pdf))

## 5.3 単位の話

コンピュータでよく使われる単位の話です。既出も含め紹介します。

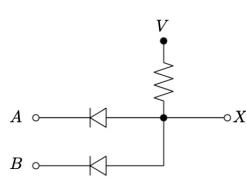
- 1 ビット (bit) コンピュータで扱うことができる最小の情報量です。スイッチが入っている (1) か、切れている (0) かの、二通りの状態を表せます。二進数 1 桁の情報量です。  
2 桁 (2 ビット) では、00, 01, 10, 11 の 4 通りを表せます。  
3 桁 (3 ビット) では、000, 001, 010, 011, 100, 101, 110, 111 の 8 通りを表せます。
- 1 バイト (Byte) 半角文字 1 文字分 (8 ビット 二進法の 8 桁分) の情報量です。256 通りの文字を区別できるため、数字や記号、アルファベットが表せます。
- 2 バイト 全角文字 1 文字分の情報量です。日本語は「ひらがな」や「カタカナ」、「漢字」などから成り立ちます。2 バイトあれば、 $256 \times 256 = 65536$  通りの文字を区別することができます。どの文字にどの番号を割り振るのか、様々な文字体系があります。以前は Shift-JIS(シフトジス)と呼ばれる文字体系が使われていましたが、現在では世界中の文字を 3 バイトで表現する UTF-8(ユーティーエフエイト)が広く使われています。
- 1 キロバイト (kB) = 1,024 バイト。2 進数 10 桁 ( $=2$  の 10 乗) で 1024 通りを表現できます。10 進法で、1,000 倍のことを k (キロ) といいます。コンピュータの世界では、 $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$  を十回掛けると 1024 になるので、1024 でひとまとめにして、1024 バイトのことを 1kB と言い、およそ A4 用紙一枚程度の情報量です。
- 1 メガバイト (MB) = 1,024 キロバイト = 1,048,576 バイトです。  
小さめの写真約 1 枚分、A4 約 1000 枚分の情報量です。昔懐かしいフロッピーディスク 1 枚は 1.44MB でした。
- 1 ギガバイト (GB) = 1,024 メガバイト = 1,048,576 キロバイト = 1,073,741,824 バイト。  
CD 1 枚 74 分で 640MB です。DVD 1 枚 2 時間で、4.7GB ~ 8.5GB です。高画質の映画などブルーレイディスクは、25GB ~ 100GB です。
- 1 テラバイト (TB) = 1,024 ギガバイト = 1,048,576 メガバイト = 1,073,741,824 キロバイト = 1,099,511,627,776 バイト。ハードディスクはとても大容量です。1TB のハードディスクで毎日 2 時間ずつ動画を見るなら、おおよそ 100 日分の容量になります。

## 5.4 計算機理論入門



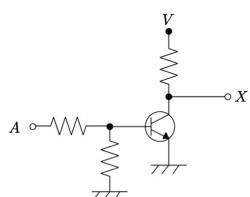
コンピュータは、0 と 1 の二進数で動いていること、その二進数を組み合わせて文字や画像など様々な情報表現が可能なことをご紹介いたしました。

黎明期には、真空管を用いて作っていたコンピュータですが、トランジスタ、IC、LSI と集積化が進み、小さなチップの中に、数百億ものトランジスタが実装されるようになりました。



OR 回路

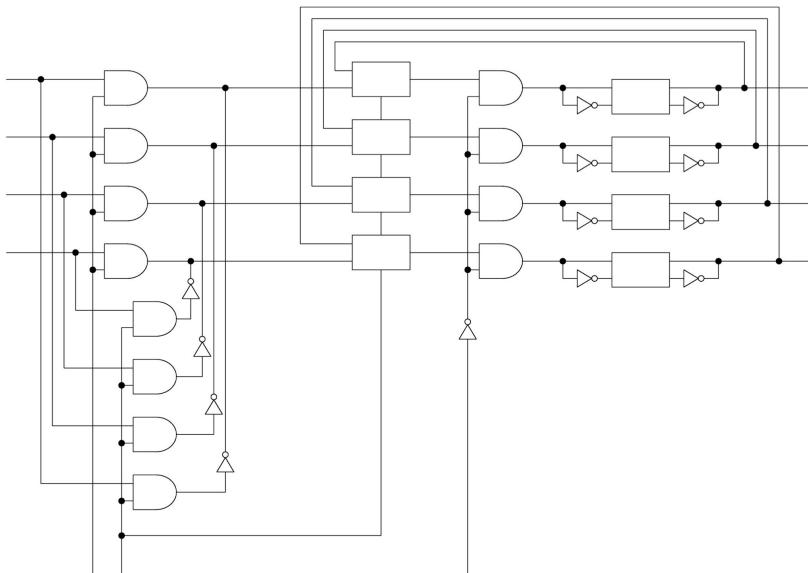
多くのトランジスタが高速に動作することで今日のコンピュータの性能がありますが、デジタル回路を支える三つの論理素子、それは AND, OR, NOT です。この三つの論理素子を組み合わせることで、演算、制御、記憶など、コンピュータを形作ることができます。



NOT 回路

より本格的に計算機理論を学びたい方のために、富山大学の幸山教授が書かれた「計算機理論入門～コンピュータを設計しよう～」がございます。

二進数の基礎から始まり、論理演算や論理回路を学び、最終的には 4 ビットの加減算ができるコンピュータシステムを作り上げていく内容です。半導体やトランジスタの仕組み、電子回路の基礎も学べる内容となっています。



▲図 5.3: 4 ビット計算機

計算機理論入門～コンピュータを設計しよう～<sup>19</sup>より、回路図を引用いたしました。

とても良く纏まっているテキストです。少し高度な内容を扱った高校生向けの内容となります。是非ご一読なさってみてください。

<sup>19</sup> <https://kouyama.sci.u-toyama.ac.jp/main/etc/2003/ssh/sshi.pdf>

## 【コラム】コンピュータ豆知識

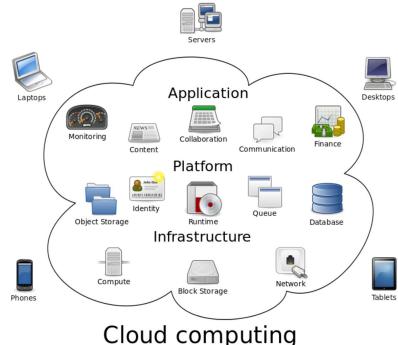
### インターネット

世界中のコンピュータ同士が繋がった巨大なネットワーク(網)。前身は ARPANET(アーパネット、高等研究計画局ネットワーク)。ジョゼフ・カール・ロブネット・リックライダー等がアイディアを創り上げた。日本では、村井純氏等が普及に多大な貢献をしている。

### クラウド \*20

英語で「雲」の意味。従来、自分のコンピュータで行った処理を、インターネット上のコンピュータ上で行う。演算の他、情報伝達、データの保等、様々な業務が提供されている。

例えば、政府の行政システムのクラウド化には、米国アマゾンとグーグル社が使われることとなった。



### 電子署名(デジタル署名)

電子署名とは、文書やメッセージなどのデータの真正性を証明するために付加される、短い暗号データ。作成者を証明し、改竄や取り替えが行われていないことを保証する。欧米で紙の文書に記されるサインに似た働きをするためこのように呼ばれる。 \*21

### 量子コンピュータ

$$\Delta x \Delta p \geq \frac{\hbar}{2}$$

量子コンピュータは、量子力学的な重ね合わせを用いて並列性を実現するコンピュータである。従来のコンピュータ(以下「古典コンピュータ」)の基本素子は、情報量が0か1のいずれの値しか持つ得ない1ビットを扱うものであるのに対して、

量子コンピュータでは量子ビット(qubit; quantum bit、キュービット)により、1キュービットにつき0と1の値を任意の割合で重ね合わせて保持する。n量子ビットあれば、2のn乗の状態を同時に計算できる。もし、数千qubitのハードウェアが実現した場合、この量子ビットを複数利用して、量子コンピュータは古典コンピュータでは実現し得ない規模の並列コンピューティングが実現する。理論上、現在の最速スーパーコンピュータで数千年かかるても解けないような計算でも、例えば数十秒といった短い時間でこなすことができる、とされている。

\*22 出典: IT用語辞典

\*23 画像出典: Wikipedia

# 第 6 章

## アルゴリズム とは

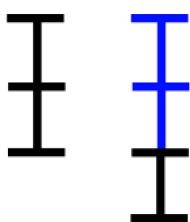
アルゴリズムとは、「計算や作業を遂行するための手順」のことです。良いプログラムを作る上で、アルゴリズムの理解は欠かせません。有名なアルゴリズムの紹介とともに、その表現に良く用いられる「流れ図」や「効率性」の考察も行います。

### 6.1 アルゴリズムとは

語源は、九世紀前半を生きたアラビアの数学者アル・フワーリズミーに因みます。日本語では「算法」と訳されます。

アルゴリズムとは、「計算や作業を遂行するための手順」<sup>\*1</sup> のことです。並び替えや探索など基本的なものから、素数判定、最大公約数、円周率の計算など、数学的なもの、分類分け、データ圧縮などなど、さまざまなアルゴリズムが考案されてきました。

#### ♣ ユークリッドの互除法



紀元前 300 年頃に記されたユークリッドの『原論』に記されている世界最古のアルゴリズムである。2 つの自然数 (1071 と 1029) の最大公約数を求める例を挙げる。1071 を 1029 で割った余りは 42。1029 を 42 で割った余りは 21。42 を 21 で割った余りは 0。よって、最大公約数は 21 である。そして簡単に求めることができる。それぞれの自然数を素因数分解しても最大公約数を求めることができるが、ユークリッドの互除法を用いると格段に速く計算可能である。

(「ラメの定理」により、割って余りを取る操作を、最悪でも小さい十進数の桁数の約 5 倍繰り返せば、最大公約数に達する。)

\*1 出典: アルゴリズム図鑑

## ♣ エラトステネスの 篩

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

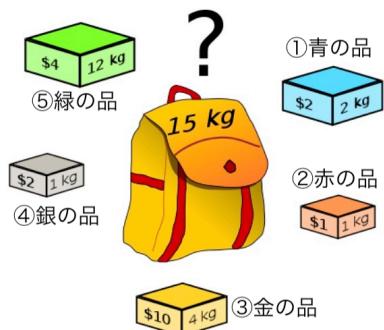
古代ギリシアの科学者エラトステネスが考案した、指定された整数以下の全ての素数を発見するための単純なアルゴリズム。100までの素数を見つける場合、図のように2から100までの数を篩に入れておく。次に篩の中で最小の数2は素数の為、その倍数を篩い落とす、残った数の中で最小の数3は素数の為、その倍数を篩い落とす。これを順に繰り返していく、 $11(\sqrt{100} \text{より大きな最初の素数})$ に達した時点で、篩に残っていた全ての数は素数である。また、地球の大きさを最初に測定したことでも知られている。

## ♣ ハノイの塔



パズルの一種。三本の杭と、大きさの異なる複数の円盤があり、左端の杭に小さいものが上になるように順に積み重ねられている。円盤を一回に一枚ずつどれかの杭に移動させることができると、小さな円盤の上に大きな円盤を乗せることはできない。右端の杭に全ての円盤を移動させよ。

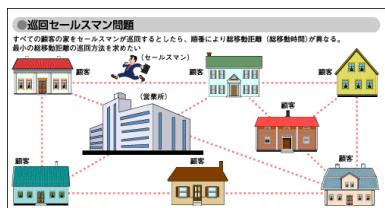
## ♣ ナップサック問題



「袋には15kgまで入れられる。重量や価値の異なる様々な品があるが、袋の価値を最大化するためには、どの品を詰めたら良いか？」という問題で、計算複雑性理論に於て、NP困難と呼ばれる問題のクラスに属する。動的計画法を用いた擬多項式時間アルゴリズムにより、実用的にはほぼ最適な解が得られる。

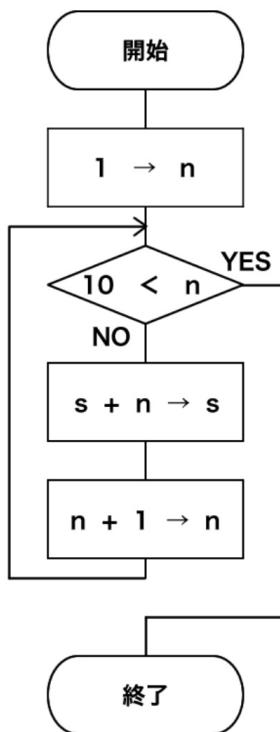
貪欲法と呼ばれるアルゴリズムでの解答例を示す。「貪欲」という名前の通り、各荷物の評価値(=価格÷重量)を算出し、評価値の高いものからナップサックに入していくのが骨子である。①の品は価値が2ドル重量が2kgであるから評価値は $2/2=1$ 、②の品は $\dots$ と順に算出していくと、①～④の品を選んだときに、価値が15ドル重量が8kgであると(最適解ではないにしろ一応の)解が得られる。

### ♣ 巡回セールスマン問題 \*2



ある地域の営業担当のセールスマンが、その地域に住んでいる顧客全員の家を巡回して訪ねることになったとする。巡回の順番によって移動距離が大きく変わるので、移動距離が少なくなるような巡回の方法を求めたい。

## 6.2 アルゴリズムと効率性



ここでは、アルゴリズムの例として、1から10までの合計を求ります。左図は「流れ図(フローチャート)」と呼ばれ、手順を示す際に良く使われます。

1. 「n」という名前の箱を用意し、1を入れます。
2. 10よりnが大きいか、条件判断します。
3. NOなら、「s」という名前の箱に、s + nを計算した答えを入れて、「n」という名前の箱に、n + 1を計算した答えを入れます。そして、2に戻ります。
4. YESなら、「終了」です。1から10までの合計が、「s」という名前の箱に入っています。

計算するための手順が「きちん」と書かれているので、確実に答えを求めることができます。

それではこのアルゴリズムの効率をみていきましょう。1から10までの合計を求める時の計算量(足し算の回数)は10回になります。1から100までの合計を求める時の計算量(足し算の回数)は100回になります。1から1000までの合計を求める時の計算量(足し算の回数)は1000回になります。

このように、nに比例して計算量が増えることを、O(n)と書き、オーダーnと読みます。

より良いアルゴリズムとして、次のように求めることも出来ます。

$$(1 + n) * (n / 2) \rightarrow s$$

計算量はO(1)と、nによらず、一定の時間で計算することができ、最高に効率的です。

解きたい課題に対して複数のアルゴリズムがあります。計算速度や実装の容易さなどを思案し、良いプログラムを書いていきましょう。

\*2 巡回セールスマン問題はimidasより、他はWikipeディアより引用・改変

## 【コラム】カール・フリードリヒ・ガウス

後に数学王と呼ばれたガウス少年の逸話をウィキペディアより紹介します。



ガウスが7歳の時、数学の授業で教師が「1から100までの数字すべてを足せ」という問題を出した。教師は生徒たちが問題を解くには相当な時間がかかるだろうと考えていたが、ガウスはわずか数秒で「5050」という解答を出し、教師を驚かせた。

1から100までの数字を足すと、 $1 + 100 = 101$ 、 $2 + 99 = 101$ 、…、 $50 + 51 = 101$ で、101の集まりが50個できるため、 $101 \times 50 = 5050$ になるとガウスは計算しました。

19歳の時にコンパスと定規のみで正十七角形を作図できることを証明したガウスは、自身の墓碑に正十七角形を刻むよう遺言した逸話も知られています。<sup>\*3</sup>

その後もガウスは、素数定理や最小自乗法など赫々たる業績を挙げ、数学界に巨大な足跡を記しました。

1777年 - ドイツ、ブラウンシュヴァイクに生まれる

1792年 - 素数定理の成立を予想

1795年 - 最小二乗法発見

1796年 - 平方剰余の相互法則の証明。

コンパスと定規のみで正十七角形を作図できることを証明

1799年 - 代数学の基本定理の証明

1801年 - 『整数論の研究』出版 複素数表記、現代整数の表記導入

1801年 - 円周等分多項式の研究

1807年 - ゲッティンゲンの天文台長になり、以後40年同職につく

1809年 - 『天体運行論』出版 最小二乗法を用いたデータ補正、正規分布

1811年 - 複素積分、ガウス平面（複素数平面）ベッセルへの手紙

1827年 - 『曲面の研究』出版、微分幾何学を創始

1855年 - ゲッティンゲンで死去

\*3 作図可能な正多角形 (<http://hooktail.sub.jp/algebra/ConstructablePolygons/>)

# 第 7 章

## プログラムとは

### 7.1 「プログラム」 = 「コンピュータへの指示書」

今日の「コンピュータ」 = 「電子計算機」が登場する以前から「プログラム」という言葉は使われてきました。語源は「前に書かれたもの」で、街頭で大勢に示す文書「公文書」の意味です。

アルゴリズムとは、ある特定の課題への解法、解き方の手順を書き表したものでしたが、このアルゴリズムを、コンピュータが計算できるよう、コンピュータが理解できる言語で書き表したもの、それが「プログラム」です。「プログラム」 = 「コンピュータへの指示書」です。

その昔のコンピュータは、特定の用途の計算のみを行うことができました。（専用計算機）黎明期のコンピュータは、回路と回路の間の配線を繋ぎ代えることで、異なる種類の計算も行うことができるようになりましたが、たくさんの配線を繋ぎ代えることはとても大変です。そのため生まれたアイディアが、「計算機への指令」 = 「プログラム」そのものを計算機に内蔵するというアイディアです。これにより、様々な手順を必要なときに読み込んで書かれた通りの計算を行うようになりました。これが、現在ほとんどのコンピュータで広く採用されている「プログラム内蔵方式」 = 「ノイマン・アーキテクチャ」です。

また、シャノン等によりスイッチのオンとオフの回路（スイッチング回路）から、論理回路・デジタル回路への道が拓かれたことや、チューリングによるチューリングマシンの研究、19世紀を生きた数学者であるジョージ・ブールの仕事、ブール代数も、今日のコンピュータの存在に大きく貢献しています。

コンピュータのプログラムは、3つの要素で成り立ちます。

- 順次処理：上から下へ順番に進みます。
- 条件判断：YES か NO で答えられる質問です。
- 繰り返し：何回でも反復します。

これらを組み合わせて、自らの意図する機能を創り上げていきます。

### 7.2 プログラミング言語の種類

人間がコンピュータに指示するために作られた言語が、プログラミング言語です。用途に応じ様々なプログラミング言語が考案されてきました。そのいくつかをご紹介します。

### ♣ 機械語(左)とアセンブリ言語(右)

```

18      CLC
F8      SED
A9 34 12 LDA #$1234
69 21 43 ADC #$4321
8F 03 7F 01 STA $017F03
D8      CLD
E2 30      SEP #$30
00      BRK

```

機械語は、「0」と「1」の二進数や「0」-「F」までの十六進数を書き連ねたものです。コンピュータはこれを読み取り直接解釈、実行可能な命令データの集まりです。

アセンブリ言語は、コンピュータが直接解釈実行可能な機械語を、人間にわかりやすいよう英略語を用いて書き表した言語です。

左側の機械語は分かりにくいですが、右側のアセンブリ言語は、CLC:比較せよ、LDA:読み込み、ADC:足せ、STA:書き込み、BRK:中断など、人に理解しやすくなっています。

### ♣ C言語

```

int n;
int answer = 0;
for(n = 1; n <= 10; n++){
    answer += n;
}
printf("%d\n", answer);

```

#### 1から10までの合計を求める

略してC(シー)ともいいます。比較的に読みやすい文法を持ちながら、機械に近いところまで書き表すことができるので、小さなコンピュータ(マイコン)が組み込まれた電子機器のプログラミングや、UNIX(ユニックス)などOS(オペレーティングシステム)の開発などに広く用いられています。

### ♣ Ruby(ルビー)

```

answer = 0
(1..10).each do |n|
    answer += n
end
print answer

```

#### 1から10までの合計を求める

まつもとゆきひろさん(通称 Matz)により開発されたオブジェクト指向スクリプト言語です。日本で開発されたプログラミング言語として初めて国際電気標準会議で国際規格に認証されました。

開発者のまつもとゆきひろさんは、「Rubyの言語仕様策定において最も重視しているのはストレスなくプログラミングを楽しむことである(enjoy programming)」と述べています。

### ♣ Scratch(スクラッチ)



#### 1から10までの合計を求める

MITメディアラボが開発した視覚的なプログラミング言語で、初めてプログラミングをする小学生が、遊び心ある実験やアニメ、ゲームなどを作ったりすることができます。

小学生でもRubyを使ってプログラムやロボットを作れるようにした、Smalrubyもお薦めです。

\*1 NPO法人Rubyプログラミング少年団 (<https://smalruby.jp>)

## 7.3 学校教育でのプログラミング

小学校からプログラミング教育が始まりました。概要を文部科学省より引用します。

### ♣ なぜプログラミング教育を導入するのか

今日、コンピュータは人々の生活の様々な場面で活用されています。家電や自動車をはじめ身近なものの中にもコンピュータが内蔵され、人々の生活を便利で豊かにしています。誰にとっても、職業生活をはじめ、学校での学習や生涯学習、家庭生活や余暇生活など、コンピュータなどの情報機器やサービスとそれによって<sup>もたらす</sup>される情報を適切に選択・活用し問題を解決していくことが不可欠な社会が到来しつつあります。

コンピュータをより適切、効果的に活用していくためには、その仕組みを知ることが重要です。コンピュータは人が命令を与えることによって動作します。端的に言えば、この命令が「プログラム」であり、命令を与えることが「プログラミング」です。プログラミングによって、コンピュータ自分が求める動作をさせることができるとともに、コンピュータの仕組みの一端をうかがい知ることできるので、コンピュータが「魔法の箱ではなくなり、より主体的に活用することにつながります。

プログラミング教育は子供たちの可能性を広げることにも繋ります。プログラミング能力を開花させ、創造力を発揮して、起業する若者や特許を取得する子供も現れています。秘めた可能性を発掘し、社会で活躍できるきっかけとなることも期待できるのです。

このように、コンピュータを理解し上手に活用していく力を身に付けることは、あらゆる活動においてコンピュータ等の活用が求められるこれからの社会を生きていく子供たちにとって、将来どのような職業に就くとしても、極めて重要なこととなっています。<sup>\*2</sup>

### ♣ 小学校でのプログラミング教育のねらい

1. 「プログラミング的思考」を育むこと。
2. プログラムの働きや良さ、情報社会がコンピュータ等の情報技術によって支えられていることに気付き、コンピュータを活用して身近な問題の解決や、よりよい社会を築く態度を育むこと。
3. 各教科内容を指導する際に行う場合は、各教科で学びをより確実なものとすること。

小学校においては、文字入力など基本的な操作を習得、プログラミングを体験しながらコンピュータに意図した処理を行わせるために必要な論理的思考力を身に付ける。

### ♣ 中学校でのプログラミング教育のねらい

中学校においては、技術・家庭科においてプログラミング、情報セキュリティに関する内容を充実「計測・制御のプログラミング」に加え、「ネットワークを利用した双方向性のあるコンテンツのプログラミング」等について学ぶ。

### ♣ 高等学校でのプログラミング教育のねらい

高等学校においては、情報科において共通必履修科目「情報 I」を新設し、全ての生徒がプログラミングのほか、ネットワーク（情報セキュリティを含む）やデータベースの基礎等について学習「情報 I」に加え、選択科目「情報 II」を開設。「情報 I」において培った基礎の上に、情報システムや多様なデータを適切かつ効果的に活用し、あるいはコンテンツを創造する力を育成。<sup>\*3</sup>

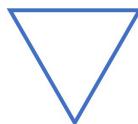
<sup>\*2</sup> 出典：文部科学省：小学校プログラミング教育の手引（第三版）

<sup>\*3</sup> 出典：文部科学省 新学習指導要領のポイント（情報活用能力の育成・ICT活用）

## ♣ プログラミング的思考とは

「自分が意図する一連の活動を実現するために、どのような動きの組合せが必要であり、一つ一つの動きに対応した記号を、どのように組み合わせたらいいか、記号の組合せをどのように改善していくば、より意図した活動に近づくのか」を論理的に考えていく力。

## ♣ 「正多角形を描く」場合について考える



コンピュータで正三角形を描く場合を見てみます。「正三角形を描く」という命令は通常は用意されていないので、そのままでは実行できません。そこで、コンピュータが理解できる(用意されている)命令を組み合わせて命令することを考えます。紙の上に作図する場合、正多角形がもっている「辺の長さが全て等しい」、「角の大きさが全て等しい」、「円に内接する」、「中心角の大きさが全て等しい」のような正多角形の意味や性質を使って作図します。

コンピュータで作図する場合にも同様に考えます。ここでは「辺の長さと、角の大きさが全て等しい」という正多角形の意味を使い作図する例を考えます。

この場合、「長さ 100 進む(線を引く)」、「左に 120 度曲がる」といったコンピュータが理解できる(用意されている)命令を組み合わせることで「正三角形を描け」ます。

より大きな正三角形を描きたければ、「長さ 200 進む(線を引く)」というように修正します。曲がる角度を変えることで、正六角形や正八角形も描くことができます。

紙の上に鉛筆と定規、分度器やコンパス等を用いて正三角形を描く際も、用いる性質や手順は異なるとしても、児童は同じように手順を考えた上で作図しているはずです。

## ♣ 未来の学びコンソーシアムより 実践例の紹介



地域の魅力を発信しよう



スポーツとデータ分析。地域スポーツプログラミングで動くワークツールチームを応援しよう



自動車に搭載された技術と私たちの生活を便利にするプログラム



正多角形をプログラムを使ってかっこくりかえしをつかってリズムをつく



くりかえしをつかってリズムをつく



主語と述語に気を付けながら場面に合ったことばを使おう



家族と食べる朝食を考えよう



ブロックを組み合わせて47都道府県を見つけよう

# 第 8 章

## ウェブの歴史と技術

今日では不可欠となったウェブサイト。その歴史をみていきます。そしてサーバとブラウザとの関係、サイト作成に用いる HTML / CSS / JavaScript の概要を紹介します。

### 8.1 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 年に欧洲原子核研究機構 (CERN) のティム・バーナーズ・リー氏により提案されました。「どのようなコンピュータからでも情報を見覽し、共用できる」ことが目的で、WWW(World Wide Web) と呼ばれます。

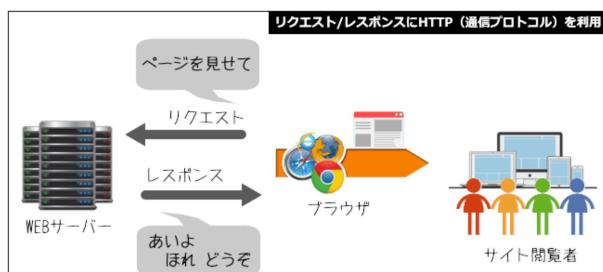
WWW は、研究機関や大学を中心に開発が進み、平成 6 年にオープンな形となりました。同時に米国立スーパーコンピュータ応用研究所 (NCSA) により提供された Mosaic ブラウザにより、多くの研究者が使い始めます。やがて Netscape(Firefox の前身) ブラウザの誕生と Windows 95 により、誰もが使える環境が整い、爆発的に普及しました。<sup>\*1</sup>

#### ♣ サーバとブラウザ

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、「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。<sup>\*2</sup>



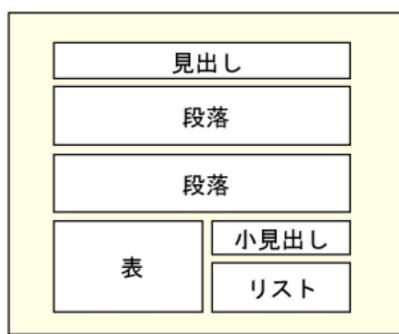
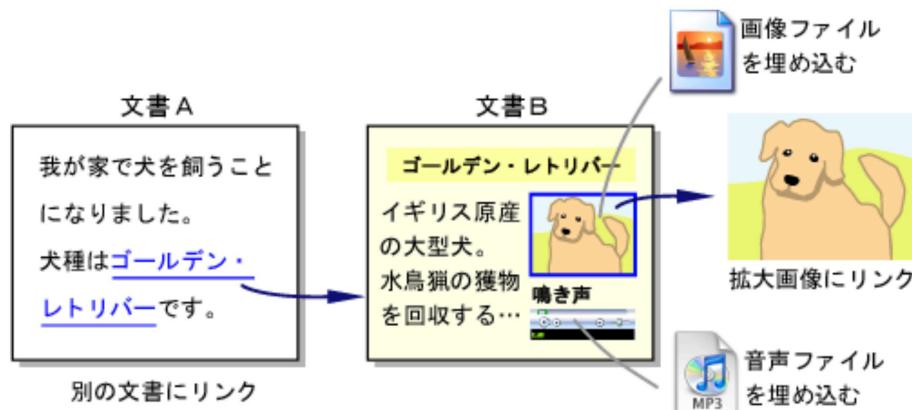
\*1 出典: CSS グリッドで作る HTML5&CSS3 レッスンブック

\*2 画像出典: <https://www.nkshopping.biz/index.php?レスポンスとは。説明文: IT 用語辞典より改変>

## 8.2 HTML とは

HTML(HyperText Markup Language) は、ウェブページを作成するために開発された言語で、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ったり、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。文書内の各部分に目印をつけ、その部分がどんな要素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

\*3 出典：HTML クイックリファレンス

## 8.3 CSS とは

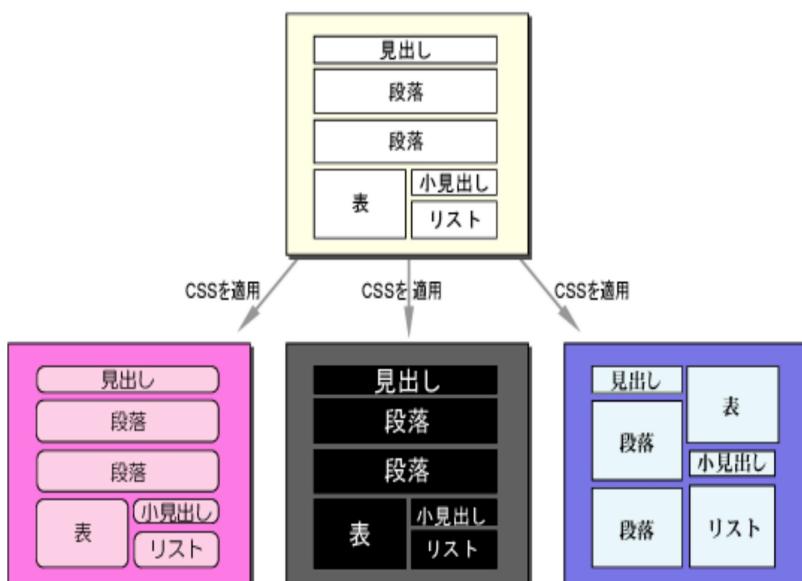
CSS(Cascading Style Sheet) とは、Web ページの要素の配置や見栄えなどを記述するための言語で、指定できる項目は、要素の大きさや配置、要素間の位置関係や空白、要素の境界線や余白、要素間の空白や周囲の余白、文字の大きさや文字や行の間隔、書体（フォント）の種類や変形（太字や斜体など）、箇条書きの表示書式、背景色や背景画像など多岐に渡ります。

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。<sup>\*4</sup>

### ◆ 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）に応じた CSS を適用することで、それぞれに適した表示が行えます。

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。<sup>\*5</sup>



▲ 図 8.1: CSS で見た目を切り替える

## 8.4 JavaScript とは

JavaScript(ジャバスクリプト) は主にブラウザで動くプログラミング言語です。

<sup>\*4</sup> 出典：IT 用語辞典

<sup>\*5</sup> 出典：HTML クイックリファレンス

## ♣ 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。

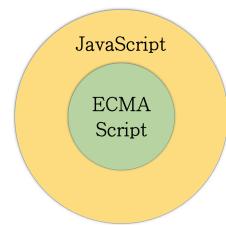
例えば、ウェブサイトで何か操作をしたら表示が書き換わったり、ウェブサイトのサーバと通信してデータを取得したりするなど、現在のウェブサイトには欠かせないプログラミング言語となっています。

## ♣ 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。

## ♣ 標準規格

JavaScript という言語は ECMAScript という仕様によって動作が決められています。ECMAScript 仕様には、基本的な記法や文法、式、宣言、関数、変数、データ型、コンテナ、組み込みオブジェクト、例外処理などの仕様を定めています。近年の拡張ではクラスやモジュール、非同期処理についても盛り込まれました。実行環境に左右されない汎用的な言語のコア部分についての規格です。



これに、Web ブラウザにおける DOM 操作や、特定のプロトコルによるネットワーク通信、ローカル環境でのファイル入出力と言った個別具体的な機能の APIなどを追加したものが、JavaScript です。

## ♣ 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。<sup>\*6</sup>

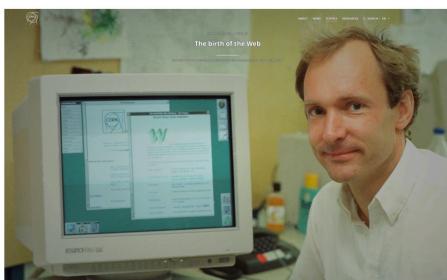
---

<sup>\*6</sup> 出典：IT 用語辞典、JavaScript Primer 迷わないための入門書

## 【コラム】ウェブの誕生

欧州原子核研究機構 (CERN) は、スイスのジュネーヴ郊外に位置する世界最大規模の素粒子物理学の研究所である。地下には 全周 27km の円形加速器・大型ハドロン衝突型加速器が、設置されており、加速器を用いた素粒子物理学および原子核物理学の研究のほか、研究に必要な有用な技術の開発などを行っている。ウェブ発祥の地としても有名である。

### ティム・バーナーズ=リー



ティム・バーナーズ=リーは、同僚のロバート・カイリューとともに World Wide Web (WWW) を考案した。そして、文献の検索および連携のために考案された言語である HTML、インターネット通信のために定めた規則 (HyperText Transfer Protocol 通称 : HTTP)、ハイパーテキストシステムを実装・開発した。<sup>\*8</sup>

右上は、初めて公開されたウェブサイト。文字のみから成るものであったが、今日と同じく、URL とハイパーリンクの仕組みにより情報共用が出来た。

右は、公開に用いられたウェブサーバである NeXTcube。平成二年のクリスマスに世界初のウェブが誕生した。

イギリスの科学者 ティム・バーナーズ=リーは、CERN 在籍していた時に World Wide Web (WWW) を発明した。世界中の大学や研究機関の科学者間で、情報を自動で共用したいという要望に応えるために考案され、開発された。<sup>\*7</sup>

### World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 news](#) , [Frequently Asked Questions](#) .

[What's out there?](#)  
Pointers to the world's online information, [subjects](#) , [W3 servers](#) , etc.  
[Help](#)  
on the browser you are using  
[Software Products](#)  
A list of W3 project components and their current state. (e.g. [Line Mode](#) , X11 [Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#) )  
[Technicals](#)  
Details of protocols, formats, program internals etc  
[Bibliography](#)  
Paper documentation on W3 and references.  
[People](#)  
A list of some people involved in the project.  
[History](#)  
A summary of the history of the project.  
[How can I help ?](#)  
If you would like to support the web..  
[Getting code](#)  
Getting the code by [anonymous FTP](#) , etc.



\*7 <https://home.web.cern.ch/science/computing/birth-web>

\*8 画像と文章はウィキペディアより引用・改変

## 【コラム】人工知能 AI

「人工知能 AI」という言葉を良く聞きます。Artificial Intelligence の略で、人にしかできなかった知的な行為（認識、推論、言語運用、創造など）を、どのような手順（アルゴリズム）とどのようなデータ（事前情報や知識）を準備すれば、それを機械的に実行できるか」を研究する学問です。（情報工学者・通信工学者 佐藤理史 氏）

人の知的能力を計算機上で実現する、様々な技術・ソフトウェア・コンピュータシステムであり、応用例として、自然言語処理（機械翻訳・かな漢字変換・構文解析・文章要約等）、専門家の推論・判断を模倣するエキスパートシステム、画像データを解析し特定のパターンを検出・抽出する画像認識等があります。昭和31年にダートマス会議でジョン・マッカーシーにより命名されました。現在では、記号処理を用いた知能の記述を主体とする情報処理や研究でのアプローチという意味合いでも使われ、また家庭用電気器具の制御システムやゲームの思考ルーチンもこう呼ばれます。

画像処理における深層学習（ディープラーニング）の有用性が競技会で世界的に認知された平成24年頃から急速に研究が活発となり、第三次人工知能ブームが到来しました。平成28年には、深層学習を導入したAIが完全情報ゲームである囲碁や将棋などのトップ棋士を破るなど、最先端技術として注目されています。

音楽分野においては、既存の曲を学習することで、特定の作曲家の作風を真似て作曲する自動作曲ソフトが登場しています。絵画分野においては、コンセプトアート用背景やアニメーションの中割の自動生成、モノクロ漫画の自動彩色など、人間の作業を補助するAIが実現しています。

プログラミングの領域においても、数十億行のコードから学習された GitHub Copilot や、Atom のプラグインとしてご紹介した tabnine もコード補完機能に AI を利用しています。

一方、「深層学習の父」と呼ばれているヨシュア・ベンジオは、中国共産党が市民の監視や政治目的で人工知能を利用していることに警鐘を鳴らしています。ヘルメットや帽子に埋め込んだセンサから、国民の脳波と感情を人工知能で監視するプロジェクトが推し進められ、ネット検閲や、官僚や囚人・歩行者に至るまで、監視カメラと警察のサングラス型スマートグラスやロボットに顔認識システム（天網）を搭載するなど、人工知能による監視社会・管理社会化が行われ、新疆ウイグル自治区では、人工知能により、約1万5千人の人々がテロリストや犯罪者の可能性があるとして、再教育キャンプに収容・拘禁されています。<sup>\*9</sup>

古くからの格言に「大いなる力には、大いなる責任が伴う」があります。技術を人の仕合わせの為に用いる叡智が求められています。<sup>\*10</sup>

\*9 出典： ウィキペディア

\*10 社会への影響について 人工知能とバイアス (<http://www.is.nagoya-u.ac.jp/dep-ss/phill/kukita/works/AI-and-bias-20190717.pdf>) が示唆に富む考察をしています。

# 第9章

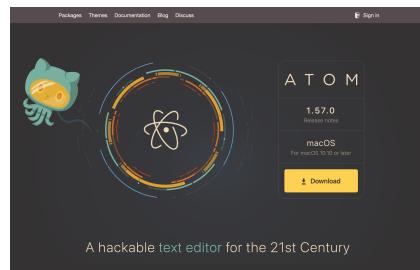
## 開発環境の準備

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。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。ここでは、Firefoxをご紹介いたします。

### 9.1 21世紀の高性能エディタ Atom

Atomは公式サイト<sup>\*1</sup>よりダウンロードできます。

「Download」ボタンを押すとダウンロードされますので、ファイルをダブルクリックして、インストールします。



#### ♣ 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。<sup>\*2</sup>

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. メニューバーや設定画面などを日本語化 [japanese-menu<sup>\\*3</sup>](https://atom.io/packages/japanese-menu)
2. カラーピッカー [color-picker<sup>\\*4</sup>](https://atom.io/packages/color-picker)

\*1 <https://atom.io>

\*2 出典：IT用語辞典

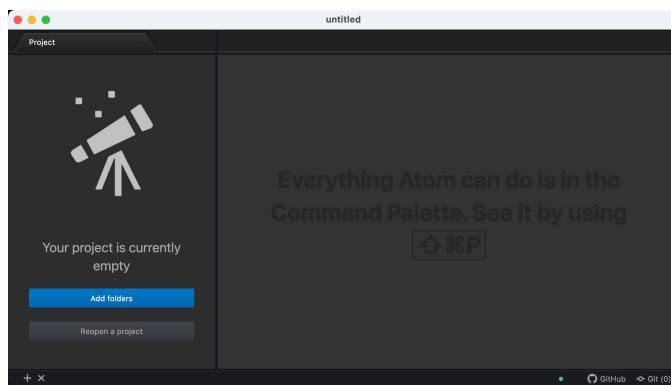
\*3 <https://atom.io/packages/japanese-menu>

\*4 <https://atom.io/packages/color-picker>

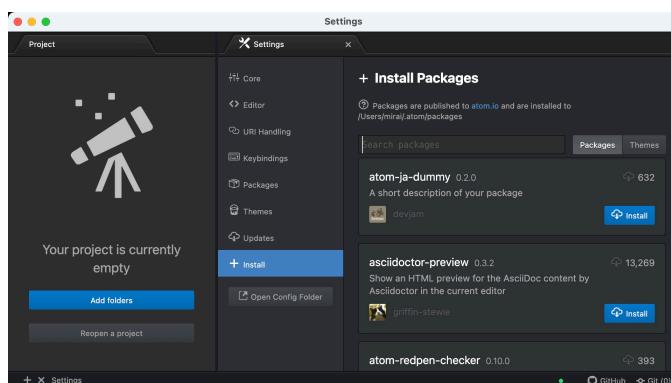
3. CSS の色指定 (#ffffff) を背景色に表示 [pigments<sup>\\*5</sup>](#)
4. ファイルの種類に応じたアイコンを表示 [file-icons<sup>\\*6</sup>](#)
5. カーソルがある列を強調表示 [highlight-column<sup>\\*7</sup>](#)
6. 選択箇所を強調表示 [selection-highlight<sup>\\*8</sup>](#)
7. AI によるコード補完機能 [tabnine<sup>\\*9</sup>](#)

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。



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。

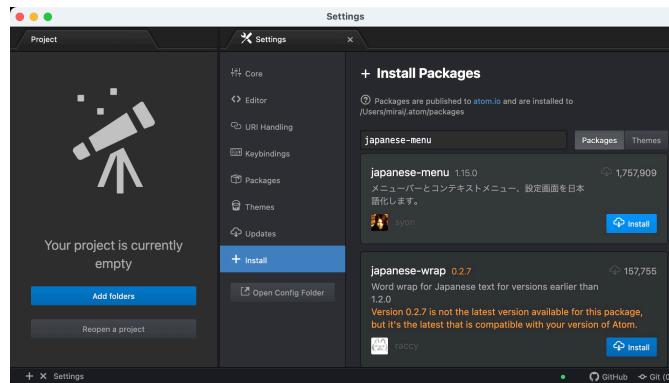
<sup>\*5</sup> <https://atom.io/packages/pigments>

<sup>\*6</sup> <https://atom.io/packages/file-icons>

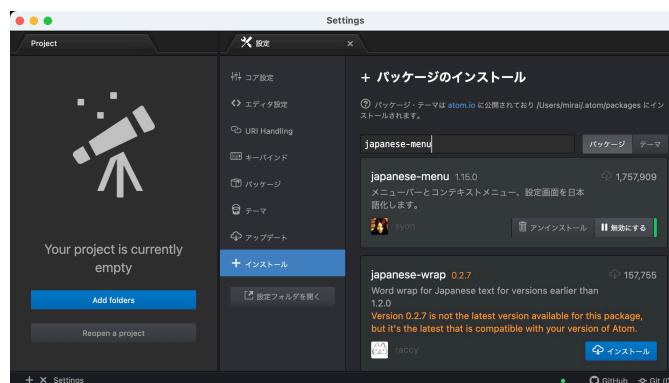
<sup>\*7</sup> <https://atom.io/packages/highlight-column>

<sup>\*8</sup> <https://atom.io/packages/selection-highlight>

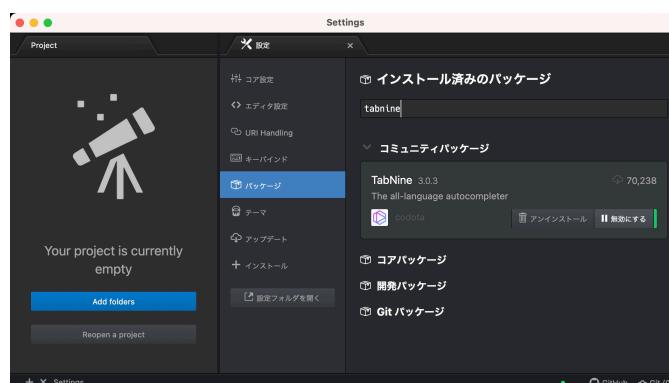
<sup>\*9</sup> <https://atom.io/packages/tabnine>



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。



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<sup>\*10</sup> 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。

## 9.2 セキュリティ重視のブラウザ Firefox

Firefox は、公式サイト<sup>\*11</sup> 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。



## 9.3 基本的な開発方法

Atom と Firefox を導入できたので、プログラムの基本的な作り方をご案内します。

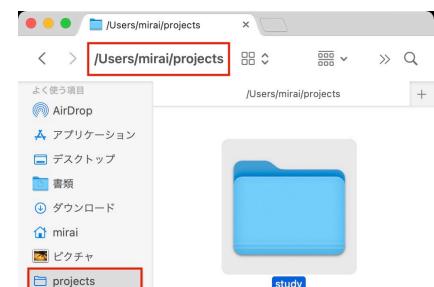
0. 左側に Atom を配置、右側に Firefox を配置します。
1. Atom で、プログラムをいろいろ書いていきます。
2. Firefox で、作ったプログラムの表示や動作を確認します。
3. 出来上がりが良ければ、プログラムを公開して完了です。

もう少し改良したければ、1. に戻ります。

### ♣ 手順のご案内

作業ディレクトリの作成方法や、Atom や Firefox の簡単な使い方の紹介です。<sup>\*12 \*13</sup>

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。

\*10 [https://zenn.dev/atelier\\_mirai/articles/c3ed79af5ba395](https://zenn.dev/atelier_mirai/articles/c3ed79af5ba395)

\*11 <https://www.mozilla.org/ja/firefox/new/>

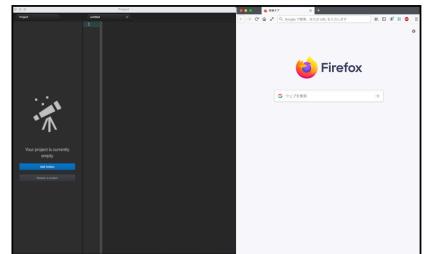
\*12 Mac 利用者向けです。Windows は適宜読み替えてください。

\*13 テキストエディタ Atom 入門 (<https://books.ojai.jp/items/atom>) もお勧めです。

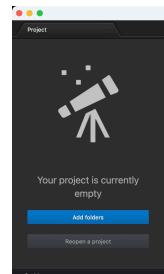
- Atom のアイコンと Firefox のアイコンです。  
良く使うのでドックに登録しておくと便利です。



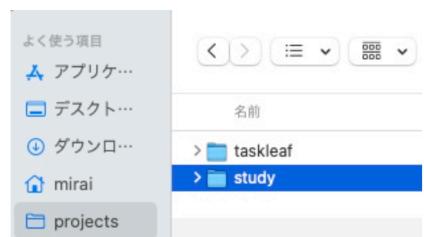
- Atom と Firefox を  
左右に配置します。



- 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、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。

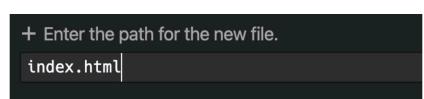


左側の「ツリービュー」に study ディレクトリが表示されました。  
この study ディレクトリ内にいろいろなファイルを納めていき、最終的にウェブアプリを割り上げま  
す。新しいファイルを作成しましょう。

- 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。



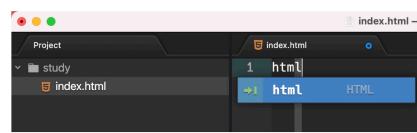
- 「Enter the path for the new file.」(新しいファイル名を入力して下さい) と、表示されるので、index.html とファイル名を入力します。



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」と呼ばれています）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。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。

8. 次のように雛形が入力されます。



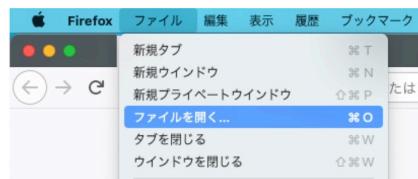
```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>

  </body>
</html>
```

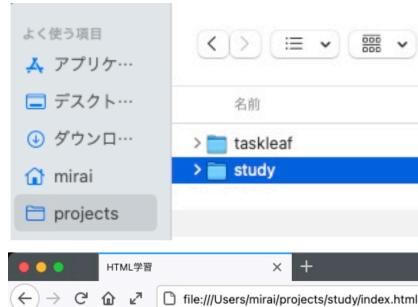
9. 「色」がついていることにも着目してください。「シンタックスハイライト」と呼ばれる、HTMLの文法に分かりやすいよう、着色する機能です。それは今雛形で入力した結果を利用して、次のようにしましょう。`h1` とタイプしエンターキーを押すと、`<h1></h1>` と入力されます。Atom の入力支援機能を活用して編集しましょう。

```
<!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に
      成ります
    </p>
  </body>
</html>
```

10. HTML を入力して、`index.html` ファイルが完成了しました。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)。



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。

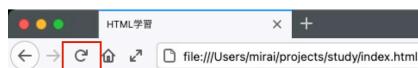


12. Atom で編集した `index.html` を Firefox で閲覧できます。

## 初めてのHTML

HTMLを学習して、素敵なサイトを作れるようになります。

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。



## 【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

### Atom 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 10 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 の文法は簡潔で理解しやすいものです。基礎が大切ですので、きっちり習得しましょう。

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 を作成しました。15 行の短いコードの中に HTML の基本が詰まっています。

HTML に関しては、(準) 公式サイト MDN による、[HTML の基本](#)\*1 が分かりやすく纏まっています。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」もお勧めです。本書でも HTML / CSS を記述の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>
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
12       成ります。
13     </p>
14   </body>
15 </html>
```

### ♣ ファイル名について

ファイル名を、index.html としました。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。

\*1 [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)

## 10.1 HTML の文法

```
<!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。

### ♣ 文書型宣言

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。今日では無意味な記述ですが、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

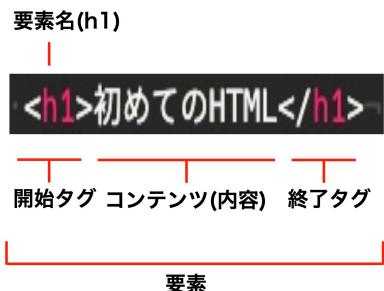
`<body>` は、コンテンツ(内容・出し物)を書く場所で、以下を記述しています。

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

### ♣ 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。



### ♣ 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。

```

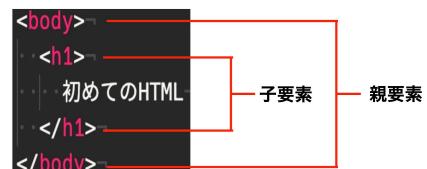
<body>
  <h1>
    初めてのHTML
  </h1>
</body>
<body>
  <h1>
    初めてのHTML
  </body>
</h1>

```

### ♣ 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くことができます。手入力しても良いですが、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。

### ♣ 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ぎ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>
```

## 10.2 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          ゲーは 0, チョキは 1, パー は 2 を入れてね。
17        </p>
```

```

18    <input type="number" id="janken_number" value="0">
19    <button id="play">開始</button>
20  </main>
21
22
23  <footer>
24    <p>
25      情報技術を夢の架け橋に<br>
26      (C) 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 --&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を示すタグです。
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 タグには、
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 タグは、大見出しを表すタグです。
20            他に 中見出しを表す h2 タグや
21            小見出しを表す h3 タグなど h6 タグまであります。 -->
22        <h1>じゃんけんゲーム</h1>
23      </header>
24
25      <!-- main タグには、このページの主題(main)となる
26          コンテンツ（内容・出し物）を記述します。-->
27      <main>
28        <!-- p は 段落(paragraph) を表す際に用いるタグです -->
29        <p>
30          <!-- br タグを入れると、途中で改行できます -->
31          みんな知っているじゃんけん。<br>
32          ゲーは 0, チヨキは 1, パーは 2 を入れてね。
33        </p>
34
35        <!-- input タグは、入力枠を作るためのタグです。

```

```

36    利用者が、0, 1, 2 のいずれかの数字を入力することで、
37    グー、チョキ、パー のどの手を出したいのか、
38    意思表示できるようにします。
39    type="number" と書いて数字用の入力枠にします。
40    後ほど、JavaScript から 扱いやすいよう、
41    この入力枠に、id="player_hand" と ID を付与します。
42    ID を付与することで、多数のHTML要素の中から
43    「一意」に特定の要素を取得することができます。
44    value="0" と書いて、枠の中に「0」を表示します。
45    つまりプレーヤーの手は グー(0) と言うことです。-->
46    <input type="number" id="player_hand_type" value="0">
47
48    <!-- button タグを使って、画面に「開始」ボタンを表示します。
49    このボタンにも、id="play" と ID を付与して置きます。-->
50    <button id="play">開始</button>
51  </main>
52
53  <!-- footer タグは、サイトの付帯情報や著作権表示などに用います。-->
54  <footer>
55    <p>
56      情報技術を夢の架け橋に<br>
57      <!-- © は「こぴーらいと」と入力して変換します -->
58      © アトリエ未来
59    </p>
60  </footer>
61
62  <!-- JavaScriptの読み込み -->
63  <script src="janken.js"></script>
64  </body>
65 </html>

```



## 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

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<sup>a</sup>](https://developer.mozilla.org/ja/docs/Web/HTML/Element)」が、(準)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。作成した index.html をブラウザで開きます。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。

<sup>a</sup> <https://developer.mozilla.org/ja/docs/Web/HTML/Element>

## 第 11 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

```
63 <!-- JavaScript の読み込み -->
64 <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、一つのウェブアプリが出来上がります。`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、進んでいきましょう。

## 11.1 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';
```

\*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、`janken.js` というファイルを追加して、これを編集していきます。

```

3 // ジャンケンの勝ち負けの結果を表示する
4 alert("あなたの勝ちです!");
5

```



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、左のように「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。

## ♣ 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 // 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!");

```

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 に標準で alert 関数が用意されています。 \*3

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。

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。すこしずつコードを追加していくことで、よりジャンケンゲームらしくしていきましょう。

どのようにすれば、より、ジャンケンゲームらしくなるでしょうか？ いきなり「あなたの勝ちで

\*2 /\* コメント \*/ というコメントの書き方もあります。

\*3 標準で備わっている関数を利用する他、プログラマが自分で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。

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。

## 11.2 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なる変数という機能があります。<sup>\*4</sup>

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、プログラム内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と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。

JavaScript には、変数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」<sup>\*5</sup> を宣言する時に使います。一度決めた値は変更できません。
- `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。
- `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、今は使用しません。

## 11.3 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。<sup>\*6</sup>

### ▼ 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;
```

<sup>\*4</sup> 出典：IT 用語辞典

<sup>\*5</sup> 厳密には「再代入できない変数」の宣言です。

<sup>\*6</sup> 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り離せない関係で、動的な Web アプリケーションを作る為には、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、ここで把握して欲しい要点は「JavaScript から HTML を操作できる」ことです。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。

## 11.4 イベントリスナと関数

### ♣ 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。\*7

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。\*8

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」\*9

### ♣ 関数

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` からはじめ

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	- 関数呼出し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1, 引数2)とし、引数が複数ある場合は,(カンマ)で区切れます。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>\*10</sup>

### ♣ 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 // イベントリスナの設定
5 const playButton = document.getElementById("play");
6 playButton.addEventListener('click', jankenHandler);
7
8 // じゃんけん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 }
```

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という仕組みが有り、JavaScriptから、HTMLの要素を操作できたことを思い出して下さい)。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にイベントリスナを追加します(addEventListener)。画面が読み込まれた時、ス

<sup>\*10</sup> 出典：JavaScript Primer 迷わぬための入門書

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>\*11</sup>

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。

## 11.5 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？

```
// じゃんけん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。それでは一万回表示させたいときはどうでしょうか。エディタにはコピー機能も備わっていますが、一万行書くのはとても大変です。

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。基本の `for` 文を使って、次のように書いてみましょう。

```
// じゃんけん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、10000と書き換えると、一万回表示させることもできます。<sup>\*12</sup>

`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) {
```

\*11 `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。

\*12 <https://ja.wikipedia.org/wiki/アラートループ事件>、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た。」「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。

```
    実行する文;
}
```

for 文の実行フロー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 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。<sup>\*13</sup>  
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が表示される
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

コンソール画面に 55 と出力されているはずです。



## 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

情報技術を夢の架け橋に  
© アトリエ未来



<sup>\*13</sup> アルゴリズムやプログラミングの章で登場してきた話題です。

## 11.6 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。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。

#### ▼ 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。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">` と書きましが、この HTML の `input` 要素を JavaScript から扱えます。

そのためには、`janken.js` 内で、`document.getElementById("player_hand_type");` と書きます。すると、このプレイヤーの入力枠を JavaScript から取得できます。取得した入力枠に名前があると扱いやすいので、`const inputBox` と書いて、`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");` と書くことで、`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。入力枠を指すようになりましたので、`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した。そしてコンピュータは「グー」を出すことにして作っていきましょう。

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。べつと、じゃんけんゲームらしくなってきましたね。

## 11.7 亂数の利用と入れ子になった `if` 文

### ♣ 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。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;
}

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);
```

いろいろじゃんけんに使えそうな関数を自作してみて下さい。

## ♣ 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 亂数で設定
14 let computer = rand(0, 2);
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     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
60 function rand(min, max){
61     return Math.floor(Math.random() * (max - min + 1)) + min;
62 }
```

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。

## 11.8 定数の利用とリファクタリング

### ♣ 定数の利用

だいぶプログラムが長くなってきました。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」のことです、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。

```
// 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; // 相子
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("相子です!");
        } 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("相子です!");
        } 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("相子です!");
        }
    }
}
```

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。

## ♣ 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」<sup>\*14</sup>と言います。

定数の利用、関数化、アルゴリズム改善等が、主なリファクタリング手法です。

## 11.9 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>\*15</sup>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。

素直にif文を書くと、プレイヤーがグーの時、チョキの時、パーの時のそれぞれにつき、コン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」になることがあります。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。

勝ち負け相手の三通りの判定をするのに、本当に五通りのif文が必要でしょうか。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、

\*14 出典：Wikipedia

\*15 <https://staku.designbits.jp/check-janken/>

- 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「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、相手が 0, 贠けが 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: 贠け 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。次のようにとても分かりやすくなりました。

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

# 第 12 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。そして利用者が快適に使えるよう、UI/UX も配慮したウェブアプリに仕上げていきます。

前章までで、じゃんけんの大まかな機能の実装は完了いたしました。この章では、せっかく作ったじゃんけんゲームですから、綺麗に意匠も整えていきましょう。

第八章で簡単に触れたように、文書構造を HTML で、利用者の入力に応じた処理を JavaScript で書くことで、じゃんけんゲームが出来ました。CSS は飾り付けや配置などの指定を行うための言語です。じゃんけんアプリを綺麗に飾り付けましょう。

### 12.1 ユーザインターフェース

人とコンピュータの間にある窓口です。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できます。<sup>\*1</sup>

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. コンピュータがどの手を出したか、表示する機能がなかったので、グーチョキパーどれなのかを示して、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。
4. 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。
5. じゃんけんは皆知っていますが、紹介文とウィキペディアへのリンクも用意します。

<sup>\*1</sup> 出典：IT 用語辞典

6. じゃんけんゲームを見る端末は様々です。iPhone で見る人、Mac で見る人など、多様な端末で好ましい見た目を提供する為、「レスポンシブデザイン」を行います。

## じゃんけんゲーム



みんな知っているじゃんけん  
グー・チョキ・パー どれかを押してね

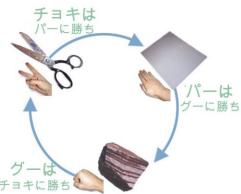


0 勝 0 敗      開始

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。

詳しくは [ウィキペディア](#)を見てね。

### じゃんけん相関図



紙 > 石 : 紙は石を包む  
鉄 > 紙 : 鉄は紙を切る  
石 > 鉄 : 石は鉄に克つ

情報技術を夢の架け橋に  
© アトリエ未来

### ♣ 完成形の HTML

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

#### ▼ 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、サイトを飾り付けします。-->
9          <link rel="stylesheet" href="janken.css">
10     </head>
11
12     <body>
13         <header>
14             <h1>じゃんけんゲーム</h1>
15         </header>
16
17         <main>
```

```

18 <!-- figure タグは 図版(figure)を示すタグです -->
19 <figure>
20   <!-- img タグは、絵(image)を表示するためのタグです -->
21   
22 </figure>
23
24 <p>
25   みんな知っているじゃんけん<br>
26   グー・チョキ・パー どれかを押してね
27 </p>
28
29 <!-- div は汎用タグで、適切なタグがない時に用います
30   クラス名をcontrol_areaとし、配置を整え易くします -->
31 <div class="control_area">
32   <!-- プレイヤーの手を選ぶ為のボタンです。 -->
33   <div class="player_hand_type">
34     <!-- JavaScriptでの操作用に id属性をguu 値を0 にします。 -->
35     <button id="guu" value="0"></button>
36     <button id="choki" value="1"></button>
37     <button id="paa" value="2"></button>
38   </div>
39   <!-- 得点と開始ボタンの領域です -->
40   <div class="score_and_replay">
41     <p class="score">
42       <!-- JavaScriptからの操作用に id属性をwinにします。 -->
43       <span id="win">0</span> 勝
44       <span id="lose">0</span> 敗
45     </p>
46     <button id="play" class="button">開始</button>
47   </div>
48 </div>
49 </main>
50
51 <!-- article は 記事(article) を表す際に用いるタグです -->
52 <article>
53   <div>
54     <p>
55       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。
56     </p>
57     <p>
58       <!-- a はanchor(锚)の意味で、他の文書へのリンクを示す
59       ウェブサイトを特徴づけるタグです。 -->
60       詳しくは<a href="https://ja.wikipedia.org/wiki/%E3%81%98%E3%82%83%E3%82%93%>
      >E3%81%91%E3%82%93">
61       ウィキペディア</a>を見てね。
62     </p>
63   </div>
64
65   <figure>
66     <!-- figcaption で 図版(figure)に標題(caption)を設定します -->
67     <figcaption>じゃんけん相関図</figcaption>
68     
69
70     <!-- ul は箇条書き項目(Unorderd List)に用いるタグです-->
71     <ul>
72       <!-- li は 項目(List)を示す際に用いるタグです -->

```

```

73      <li>紙 > 石 : 紙は石を包む</li>
74      <li>鉄 > 紙 : 鉄は紙を切る</li>
75      <li>石 > 鉄 : 石は鉄に克つ</li>
76    </ul>
77  </figure>
78 </article>
79
80  <footer>
81    <div class="container">
82      <p>情報技術を夢の架け橋に</p>
83      <p> © アトリエ未来 </p>
84    </div>
85  </footer>
86
87  <script src="janken.js"></script>
88 </body>
89 </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、サイトを飾り付けします。-->
9  <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

```

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

```

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

```

21 

```

`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

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

```
32 <!-- プレイヤーの手を選ぶ為のボタンです。 -->
33 <div class="player_hand_type">
34   <!-- JavaScriptでの操作用に id属性をguu 値を0 にします。 -->
35   <button id="guu"  value="0"></button>
36   <button id="choki" value="1"></button>
37   <button id="paa"   value="2"></button>
38 </div>
```

UI 改善のため、ボタン要素を導入しています。CSS でグレ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、今までグレーの時は `0` と入力していましたが、グレ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

```
60 詳しくは<a href="https://ja.wikipedia.org/wiki/%E3%81%98%E3%82%83%E3%82%93%E3%81%91%E>
61 >%E3%82%93">
   ウィキペディア</a>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ウィキペディア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。

## 12.2 CSS の基本

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; /* 要素の幅を制御しやすくする */
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 */
22   display: grid;         /* グリッド(格子)線を使うモードにする */
23   grid-template-columns: /* column(列) の設定を行う */
24   20px 1fr 20px;        /* 左右に20px 残りは中央 */
25   grid-template-rows:    /* row(行)の設定を行う */
26   [head]    100px /* 一行目の高さは100px headと命名 */
27   [main]    auto  /* 二行目の高さは自動 titleと命名 */
28   [article]  auto  /* 三行目の高さは自動 subtitleと命名 */
29   [foot]    100px /* 四行目の高さは100px footと命名 */
30 }
31
32 /* 部品の配置 */
33 body > * { /* body の直下(>) にある全ての要素(*)を対象に */
34   grid-column: 2 / -2; /* 列配置は 左から2番目の線から
35                         右から数えて二番目(-2)の線まで */
36 }
37
38 /* ヘッダー */
39 header {                /* header 要素を対象に */
40   grid-row: head; /* 行の配置は先ほど命名したheadの線の下に */
41   justify-self: center; /* 左右中央揃えで配置する */
42   align-self: center;  /* 上下中央揃えで配置する */
43   font-size: 20px;     /* 書体の大きさは40px */
44   color: var(--kurohairo); /* 文字の色は、黒羽色 #0d0d0d; */
45   /* 右に5px下に5pxずれた所にぼかし幅5pxで
46   鈍色 #9ea1a3 の影を付ける */
47   text-shadow: 5px 5px 5px var(--nibiro);
48 }
49

```

```

50 header h1 {          /* header 内の h1 (大見出しの指定) */
51   text-align: center; /* 文字は中央揃えにする */
52 }
53
54 /* メイン(ウェブサイトの主要機能部)用のスタイル */
55 main {                /* main 要素を対象に */
56   grid-row: main;
57 }
58
59 main figure img {    /* コンピュータのじゃんけんの絵です */
60   max-height: 300px;  /* 絵が歪まぬよう、最大高さを指定します */
61 }
62
63 main p {              /* じゃんけんの説明文です */
64   text-align: center;  /* 真ん中揃えにします */
65   margin-bottom: 20px; /* p 要素の下側に少し余白を設けます */
66 }
67
68 /* class="control_area" とクラス属性を付与した要素を対象に */
69 .control_area {
70   /* control_area内部の要素もグリッドレイアウトで配置します */
71   display: grid;
72   grid-template-columns: 1fr;    /* column(列) は 一列 */
73   grid-template-rows: 1fr auto; /* row(行) は 二行用意します */
74   row-gap: 20px;              /* 行の間隔は20px開けます */
75 }
76
77 /* control_area内でplayer_hand_typeとクラス属性を付与した要素を対象に */
78 .control_area .player_hand_type {
79   display: grid;            /* 内部の要素をグリッドレイアウトで配置します */
80   grid-template-columns: 1fr 1fr 1fr; /* 三列用意します */
81   grid-template-rows: 1fr;     /* 一行用意します */
82 }
83
84 .control_area .player_hand_type button {
85   background-color: transparent; /* ボタンの背景色は透明に */
86 }
87
88 /* html で id="guu" と id属性を付けたグーの背景画像の設定 */
89 #guu { background-image: url('player_guu.png'); }
90 #choki { background-image: url('player_choki.png'); }
91 #paa { background-image: url('player_paa.png'); }
92
93 /* グーチョキパー各ボタンの大きさを指定 */
94 .player_hand_type button {
95   background-size: 100% 100%; /* 背景画像の大きさは縦横100% */
96   border: none;             /* ボタンの枠線は無し */
97   height: 100px;            /* ボタンの高さは100px */
98   padding: 0;               /* 内部への詰め物は無し */
99   cursor: pointer;         /* カーソルの形状は pointer(手のマーク) */
100 }
101
102 /* 得点領域の指定 */
103 .score_and_replay {
104   display: grid;           /* 内部の要素をグリッドで配置 */
105   grid-template-columns: 1fr 1fr; /* 二列用意します */
106   grid-template-rows: 1fr;     /* 一行用意します */
107   align-self: center;       /* 上下方に向中央揃えにします */

```

```
108 }
109
110 /* クラス属性.score と、id属性#playを付与した要素を対象に */
111 .score, #play {
112   font-size: 24px;      /* 書体の大きさは24px */
113   margin: 10px 0;       /* 上下に10px 左右に0px の余白 */
114   align-self: center;  /* 上下方向で中央揃え */
115 }
116
117 /* <a class="button"> と buttonクラスを付与したa要素を対象に */
118 .button {
119   text-decoration: none;    /* 下線による飾り付け(decoration)は無し */
120   color: white;           /* 文字の色は白 */
121   border: solid 1px white; /* 枠線はしっかりした(solid)線で 1px幅の白 */
122   padding: 10px;           /* 詰め物は 上下左右に10px */
123   border-radius: 10px;     /* 角は 半径(radius) 10pxで丸くする */
124   background: var(--amairo); /* 背景色は 天色 */
125   cursor: pointer;        /* カーソルの形状は pointer(手のマーク) */
126 }
127
128 /* じゃんけん紹介記事 */
129 article {                  /* aritcle 要素を対象に */
130   grid-row: article;        /* 行の配置は 先ほど命名した article の線の下に */
131   border: 5px double var(--utsushiiro); /* 枠線は5pxの二重線 色は移色 */
132   border-radius: 1rem;      /* 角を1文字分丸くする */
133   padding-bottom: 1rem;     /* 要素下側に1文字分詰め物をする */
134   margin-bottom: 1rem;      /* 要素下側に1文字分余白を設ける */
135
136   display: grid;            /* 内部要素をグリッドで配置します */
137   grid-template-columns: 1fr; /* 列は一列 */
138   grid-template-rows: auto auto; /* 行は二列 用意する */
139   margin-top: 20px;          /* 上側を20px開ける */
140   justify-self: center;    /* 水平方向に中央揃え */
141 }
142
143 article p {                /* aritcle 要素内の p要素を対象に */
144   font-size: 18px;           /* 書体の大きさは10px */
145   text-align: left;          /* 文字は左寄せ */
146   padding: 1rem;            /* 1文字分詰め物をする */
147   text-indent: 1rem;         /* 文字を1文字字下げする */
148 }
149
150 /* aritcle内の figure内の figcaption要素を対象に */
151 article figure figcaption {
152   font-size: 24px;           /* 書体の大きさは24px */
153   font-weight: bold;         /* 文字は太字 */
154   text-align: center;        /* 文字は中央揃え */
155   margin-bottom: 1rem;       /* 下側に1文字分間隔を取る */
156 }
157
158 /* aritcle内の figure内の img要素を対象に */
159 article figure img {
160   width: 75%;               /* 画像の横幅の指定 */
161   max-width: 400px;          /* 画像の最大幅の指定 */
162   height: auto;              /* 高さは、横幅に応じて自動 */
163   display: block;            /* 中央揃えのためにブロックタイプに変更 */
164   margin: 0 auto;             /* 左右中央揃えに */
165   margin-bottom: 1rem;        /* 画像の下を1文字分開ける */
```

```

166 }
167
168 article ul {           /* aritcle内の ul内の li要素を対象に */
169   list-style: none;    /* 箇条書きの・(黒丸)は不要 */
170   padding: 0;          /* 詰め物も不要 */
171 }
172
173 article ul li {        /* aritcle内の ul内の li要素を対象に */
174   justify-self: center; /* 水平方向に中央揃え */
175   text-align: center;   /* 文字は中央揃え */
176 }
177
178 footer {                /* footer 要素を対象に */
179   grid-column: 1 / -1;   /* 列配置は、左から一番目の線から、
180                         右から数えて一番目(-1)の線まで */
181   grid-row: foot;       /* 行の配置は先ほど命名した foot の線の下 */
182   background: var(--sakurairo); /* 背景色は桜色 */
183   display: grid;         /* グリッド(格子)線を使うモードにする */
184 }
185
186 footer div {            /* footer 要素の中のdiv要素を対象に */
187   grid-column: 1;         /* 1番目のグリッド線の右側に配置 */
188   grid-row: 1;            /* 1番目のグリッド線の下側に配置 */
189   justify-self: center;  /* 左右中央揃えで配置する */
190   align-self: center;    /* 上下中央揃えで配置する */
191   text-align: center;    /* 文字は中央揃え */
192 }
193
194 /* デスクトップ版の追加設定 */
195 /* 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 */
196 @media (min-width: 768px) {
197   body {
198     display: grid;           /* グリッド(格子)線を使うモードにする */
199     grid-template-columns:   /* column(列) の設定 */
200       1fr 375px 375px 1fr; /* 左右に余白 中央に375pxを二列確保 */
201     grid-template-rows:     /* row(行)を用意する */
202       [head] 100px /* 一行目の高さは100px */
203                   /* 線名をheadと命名 */
204       [main article] auto /* 二行目の高さは自動 */
205                   /* 線名を main 及び article と命名 */
206                   /* 二つの線を一本に束ねる */
207       [foot] 100px;        /* 四行目の高さは100px footと命名 */
208   }
209
210   main {
211     grid-column: 2 / 3; /* main 要素を二列目から三列目までに配置 */
212   }
213
214   main figure img {
215     max-height: 400px; /* コンピュータのじゃんけん絵の最大高を400pxに */
216   }
217
218   .player_hand_type button {
219     height: 140px;      /* プレイヤーのじゃんけんボタンの高さを140pxに */
220   }
221
222   article {
223     grid-column: 3 / 4; /* aritcle 要素を三列目から四列目までに配置 */

```

```
224 }
225 }
```

## ♣ 文法

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;
}

img { width: 100%; }
```

The diagram illustrates the structure of a CSS rule. It shows the selector (\*), properties (margin: 0;), values (0), declaration ({}), and the rule itself (img { width: 100%; }). The selector and properties are grouped under 'セレクタ' (Selector). The properties and values are grouped under '宣言' (Declaration). The declaration and the rule itself are grouped under 'ルール' (Rule).

デザインをしやすくするため、全ての要素の余白を 0 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。例えば、`<img>` 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) で区切ります。

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) で区切れます。セレクタと宣言の記述全体は、「ルール」と呼ばれます。同じところに適用するルールを、複数のルールに分けて記述することができます。

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

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

写真の幅を横幅一杯 (100%) に、高さを縦横比が保たれるよう自動調整するための CSS です。

`img { width: 100%; } img { height: auto; }` と複数のルールに分けて記述しても、同じ結果が得られます。

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。

## ♣ 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

`* { }` のように、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 { }` と要素名として `img` を指定すると、`img` 要素のみに適用先を限定できます。

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

### 特定の要素の中の img 要素

HTML の階層構造を使い、適用先を限定可能です。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 要素

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

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

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

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続けて「`body > img, img + article { }`」のように指定すると、`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>
```

```
<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。

### 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。クラス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。

## ♣ 和名での色指定

CSS カスタムプロパティ を定義すると、和名<sup>\*2</sup>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(なのはないいろ) */
  --sanaeiro: #67a70c; /* 早苗色(さなえいろ) */
  --amairo: #2ca9e1; /* 天色(あまいいろ) */
  --utsushiro: #3d6eda; /* 移色(うつしいいろ) */
  --botaniro: #e7609e; /* 牡丹色(ばたんいろ) */
  --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) */
  --nibiyo: #9ea1a3; /* 鈍色(にびいろ) */
  --kurohairo: #0d0d0d; /* 黒羽色(くろはいいろ) */

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

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

## ♣ CSS での色指定

CSS での色指定は、光の三原色（赤、緑、青）の強度を16進数で表したRGBカラーが主流です。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。

16進数での色の指定の他、主な色は、名前で指定することもできます。

\*2 和の色日本の伝統色 (<https://irononamae.web.fc2.com/wa/>)

```

h1 {
  color: #ff0000; /* 赤 */
}

p {
  color: red;      /* 赤 */
}

```

## 12.3 CSS グリッドレイアウトとレスポンシブデザイン

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。

### ♣ CSS グリッドとは

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 */
22   display: grid; /* グリッド(格子)線を使うモードにする */
23   grid-template-columns: /* column(列) の設定を行う */
24     20px 1fr 20px; /* 左右に20px 残りは中央 */
25   grid-template-rows: /* row(行) の設定を行う */
26     [head]    100px /* 一行目の高さは100px headと命名 */
27     [main]    auto  /* 二行目の高さは自動 titleと命名 */
28     [article] auto  /* 三行目の高さは自動 subtitleと命名 */
29     [foot]    100px; /* 四行目の高さは100px footと命名 */
30 }

```

`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、要素の配置が可能となります。

### ♣ グリッド線を使って要素を配置する

```

39 header {           /* header 要素を対象に */
40   grid-row: head; /* 行の配置は先ほど命名したheadの線の下に */
41   justify-self: center; /* 左右中央揃えで配置する */
42   align-self: center; /* 上下中央揃えで配置する */
43   font-size: 20px; /* 書体の大きさは40px */
44   color: var(--kurohairo); /* 文字の色は、黒羽色 #0d0d0d; */
45   /* 右に5px下に5pxずれた所にばかり幅5pxで */
46   /* 鈍色 #9ea1a3 の影を付ける */
47   text-shadow: 5px 5px 5px var(--nibairo);
48 }

```

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。



### ♣ 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CSSを書きました。携帯電話が登場したこと、携帯用のウェブサイトが作られることになりましたが、パソコン用、携帯用、それぞれ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 (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)での閲覧時に適用するCSSを記すことができます。

#### ▼ janken.css

```

195 /* 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 */
196 @media (min-width: 768px) {
197   body {
198     (略)
199   }
200 }

```

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、是非一読なさってください。

# 第 13 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。

この章では次の四点を実装し、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。

### 13.1 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

##### ▼ 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、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。

`parseInt` は、文字列としての"0"を解析(`parse`)し、整数値 0 を返す関数です。

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);
```

## ♣ 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ていました。

`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。`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。<sup>\*1</sup>

### 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);
```

<sup>\*1</sup> 容器だけで中身は空っぽの幕の内弁当と、おかげとしてグーチョキパーの三つが入っている幕の内弁当を想像すると分かりやすいでしょうか。

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。

## 13.2 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>\*2</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ジャンケンプログラムの主となる機能であることから、`main`<sup>\*3</sup> という関数名にします。すると、次のように書けます。

<sup>\*3</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。

```
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。

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>\*4</sup> を用いることにすると、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

---

<sup>\*4</sup> Animation is pause? が変数名の由来です。

```
// 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;
```

```
// 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、`parseInt` 関数を用いて、文字列としての "0" を解析 (parse) し、整数値としての 0 を得ます。整数値としての 0 を得ることができましたから、「+ 1」と足し算することで、勝ち数を一つ増やせます。

`win.innerText = 1` と書くことで、`<span id="win">0</span>` と書かれていた HTML を JavaScript から `<span id="win">1</span>` のように更新できます。

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;
    }
}
```

## 13.3 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 // 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; // 负け
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
15 // グローバル変数宣言
16 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
17 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)
18
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     // ジャンケンの切替アニメが再開(resume)されるようする。
40     const play_button = document.getElementById("play");
41     play_button.addEventListener('click', resume);
42 }
43
44 // ジャンケン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」に更新する
47     const play_button = document.getElementById("play");
48     play_button.innerText = "もう一度";
49
50     // アニメーション停止処理実行
51     pause();
52
53     // 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 // ジャンケン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
98 // 亂数を返す関数
99 // rand(0, 2)と呼び出せば、0, 1, 2 と ゲーチョキバー の乱数を返す
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。

```

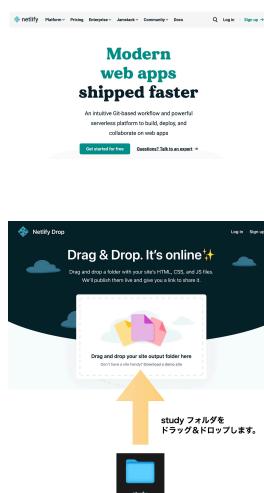
## ♣ ウェブサイトへの公開

遂に完成したじゃんけんプログラムを、インターネットに公開しましょう。Netlify<sup>a</sup> は、静的サイトを公開できるウェブサービスです。小規模利用なら無料です。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。

<sup>a</sup> <https://www.netlify.com>

今回は簡単に体験できるよう、[https://app.netlify.com/drop<sup>a</sup>](https://app.netlify.com/drop)より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。

<sup>a</sup> <https://app.netlify.com/drop>



## 【コラム】日本最初のホームページ<sup>\*5</sup>



日本最初のホームページは、平成4年9月30日に茨城県つくば市にある文部科学省高エネルギー加速器研究機構 計算科学センターの森田洋平博士によって発信されました。左のサーバー世界地図は、ティム博士の論文からのものです。

## KEK Information

Welcome to the KEK WWW server. This server is still in the process of being set up. If you have question on this KEK Information page, send e-mail to morita@kek.jp.

### Help

On this program, or the [World-Wide Web](#).

### HEP

World Wide Web service provided by other High-Energy Physics institutes.

### KIWI

KEK Integrated Workstation environment Initiative.

### Root

WS Manager Support (Root) [EUC].

See also:

[Types of server](#), and [OTHER SUBJECTS](#)

当時、世界にあったサーバーの位置が記されており日本の部分には、KEKと記されています。

ソースコードには、<HTML>や<BODY>といったタグではなく、タグが大文字で書かれているのも時代を感じます。

### <H1>KEK Information</H1>

Welcome to the KEK WWW server. This server is still in the process of being set up. If you have question on this KEK Information page, send e-mail to morita@kek.jp.

<DL>

<DT><a href="LineMode/QuickGuide.html">Help</a>  
<DD>On this program, or the <a href="http://info.cern.ch/hypertext/WWW/TheProject.html">World-Wide Web</a>

<DT><a href="http://info.cern.ch/hypertext/DataSources/bySubject/Physics/HEP.html">H E P</a>  
<DD>World Wide Web service provided by other High-Energy Physics institutes.

<DT><a href="kiwi.html">KIWI</a>  
<DD>KEK Integrated Workstation environment Initiative.

<DT><a href="Root.html">Root</a>  
<DD>WS Manager Support (Root) [EUC].

<DT>See also:

<DD>

<a href="http://info.cern.ch/hypertext/DataSources/ByAccess.html">Types of server</a>, and <a href="http://info.cern.ch/hypertext/DataSources/bySubject/Overview.html">OTHER SUBJECTS</a>

\*5 日本最初のホームページ (<http://www.ibarakiен.gr.jp/www/>) よりのご紹介です。

## 付録 A

# 珠玉の名著のご紹介

### ♣ 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 Web サイトの構築を学ぶ人 ●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。

### ♣ JavaScript Primer 迷わないための入門書



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。

### ♣ 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く、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、しっかりと JavaScript の基礎文法を身につけられます。

### ♣ 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、その精髓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。

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。

### ♣ 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、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 26 のアルゴリズム



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



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。

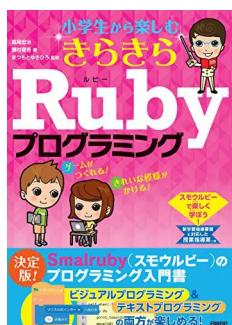
プログラミングを学ぶ最初の一冊に最適な入門書です。

### ♣ 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ります。

### ♣ 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。

#### paiza<sup>\*5</sup>

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。

#### Ruby on Rails チュートリアル<sup>\*6</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>\*7</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>\*8</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>\*9</sup>, pro.photo<sup>\*10</sup>, BEIZ<sup>\*11</sup>, 足成<sup>\*12</sup>, Pixabay<sup>\*13</sup>, 写真 AC<sup>\*14</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>\*15</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>\*16</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。

♣ **21 世紀のプログラマの為のテキストエディタ Atom<sup>\*17</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>\*18</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>\*19</sup>, Weblio 英語翻訳<sup>\*20</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>\*21</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。

---

\*9 <https://www.pakutaso.com/>

\*10 <https://pro-photo.jp>

\*11 <https://www.beiz.jp>

\*12 <http://www.ashinari.com>

\*13 <https://pixabay.com/ja/>

\*14 <https://www.photo-ac.com>

\*15 <https://www.webdesignclip.com>

\*16 <https://www.apple.com/jp/macbook-air/>

\*17 <https://atom.io/>

\*18 <https://www.mozilla.org/>

\*19 <https://www.deepl.com/translator>

\*20 <https://translate.weblio.jp>

\*21 <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。考えているものを見つけやすいうように、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

#### ♣ 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

#### ♣ 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

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

\*2 出典：IT 用語辞典

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。
<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、href 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、small から x-small 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。class または id 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リ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ユーザー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>

<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。

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。

## C.2 CSS 簡易まとめ

CSS に関する簡易なまとめです。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;
}
```

\*4 <https://developer.mozilla.org/ja/docs/Web/CSS>

## ♣ 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。

#### 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 つしかないとします。

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
<code>/* コメント */</code>	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
<code>background</code>	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。
<code>border</code>	要素の境界(枠線)を設定します。これは <code>border-width</code> / <code>border-style</code> / <code>border-color</code> の値を設定します。
<code>border-radius</code>	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楕円の角になります。
<code>box-shadow</code>	要素のフレームの周囲にシャドウ(影)効果を追加します。
<code>color</code>	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。
<code>display</code>	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。
<code>filter</code>	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。
<code>font-family</code>	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。
<code>font-size</code>	フォントの大きさを定義します。
<code>font-style</code>	通常体(normal)、筆記体(italic)、斜体(oblique)のどれでスタイル付けするかを設定します。
<code>font-weight</code>	フォントの太さ(あるいは重み)を指定します。
<code>height</code>	要素の高さを指定します。
<code>line-height</code>	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。

\*5 <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>	インラインボックス、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 \*6

### ♣ 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。 \*7

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。 <code>x</code> に値の再代入はできない
<code>let x</code>	変数宣言。 <code>const</code> と似ているが、 <code>x</code> 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。 \*8

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>	<code>x</code> と <code>y</code> 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 <code>k</code> 、 プロパティの値が <code>v</code> のオブジェクト（連想配列）を作成

\*3 出典：JavaScript Primer 迷わないための入門書 (<https://jsprimer.net>)

## ♣ 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ふ。<sup>\*10</sup>

コード例	説明
"xxx"	ダブルクオートの文字列リテラル。
'xxx'	シングルクオートの文字列リテラル。
'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

## ♣ 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られる、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>\*11</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>\*12</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	( … )	12	小なり		… < …
20	メンバへのアクセス	左から右	… . …		小なりイコール		… <= …
	計算値によるメンバへのアクセス	左から右	… [ … ]		大なり	左から右	… > …
	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 …
	オプショナルチェックイング	左から右	?.		instanceof		… instanceof …
19	new (引数リストなし)	右から左	new …	11	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		… === …
17	論理 NOT	!	…		厳密不等価		… !== …
	ビットごとの NOT		~ …	10	ビット単位 AND	左から右	… & …
	単項 +		+ …		ビット単位 XOR	左から右	… ^ …
	単項 -		- …		ビット単位 OR	左から右	…   …
	前置インクリメント	右から左	++ …		論理 AND	左から右	… && …
	前置デクリメント	-- …	論理 OR		左から右	…    …	
	typeof	typeof …	Null 合体		左から右	… ?? …	
	void	void …	条件		右から左	… ? … : …	
	delete	delete …				… = …	
	await	await …				… += …	
16	べき乗	右から左	… ** …	3	代入	右から左	… -= …
15	乗算		… * …				… *= …
	除算		… / …				… *= …
	剰余		… % …				… /= …
14	加算	+	… + …				… %= …
	減算		… - …				… /= …
	左ビットシフト		… << …				… <<= …
13	右ビットシフト	>>	… >> …				… >>= …
	符号なし		… >>> …				… >>>= …
	右ビットシフト						… &= …

\*12 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こと。<sup>\*13</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"	デフォルトインポート
import { x } from "./x.js"	名前付きインポート
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

## ♣ 制御構造

プログラムの流れを制御するための構文です。

例	説明
<code>while(x){}</code>	<b>while ループ。</b> x が <code>true</code> 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> <code>x &lt; y</code> が <code>true</code> 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 <code>true</code> 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 <code>true</code> 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>

## 【コラム】金の延棒クイズ 【解答】

最後までお読みください、ありがとうございます。金の延棒クイズの解答です。

2回鉄を入れて、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

♣ × モ

♣ × モ

# 終わりに

本書では、算盤から iPhone に至るまでの歴史を俯瞰、計算機科学の基礎知識に触れ、HTML / CSS / JavaScript によるウェブアプリを作成、公開いたしました。

全部で、108行のじんけんプログラム、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<sup>\*16</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つてください。

令和の御世を生きる皆さんのが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

---

\*16 「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。

# 計算機とプログラミング

算盤から iPhone までの歩み

---

令和四年一月二二日 ver 4.0.0

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>

---

© 令和四年 アトリエ未来