プログラミング

講習資料

目次

まえがき

この資料をご覧になってくださり、ありがとうございます。私自身は、学部 2 年の時に授業で C 言語を学んで割と楽しい、と思い、コロナ禍と重なった学部 3 年の時から独学で Web プログラミングを始めました。今では再履バス同好会の Web コンテンツの拡充を会長と協力しながら行っている状態です。稼げるほどの能力はあるわけではありませんが、趣味でやる分には楽しくやっている、という状態です。

稼ぐ、という話が出ましたが、プログラミングで稼ぐことはできます。例えば、クラウドワークスに登録すると、プログラミングの仕事を受注することができます。ただ、そう簡単にはいきません。そもそも、難度の高いアプリ開発になれば現役エンジニアに太刀打ちできないわけですし、また比較的難度の低いものであれば応募人数がそもそも多く、なかなかクライアントから仕事を受注できない、という感じになっています。そもそも仕事を得るにはプログラミングスキルだけでなく交渉スキルが必要です。私はクラウドワークスにお試しで登録しましたが、あまりの仕事のならなさを感じてそっと閉じました。営業力に自信のある方、成長意欲の高い方はハングリーさを持ってぜひ稼ぐことにチャレンジしてはいかがでしょうか。この資料はそのための踏み台として使ってください。

あと、プログラミングができると就職活動も有利になります。やはり今の時代はプログラミングができる人材を多く求めているので、経験者は優遇されます。ただし、「プログラミングできてなおかつ阪大生（じゃない方もいらっしゃるかもしれませんが）だから、IT 企業のいいとこ行けるだろ」と思うと痛い目を見ます。一般的に IT 企業は多重下請け構造になっています。多くの阪大生が目指すのは 1 次請け（N〇T データとか、日〇ソリューションズとか）の比較的待遇がいいところなのですが、そこは大してプログラミングしません。むしろお客様と直接関わって困りごとを聞くのが仕事なので、ゴリゴリの営業です。（実際知り合った人が某日〇ソリューションズの SE のタマゴ養成講座のインターンに参加したところ、ものづくりがほとんどなくてつまらなかった、と言ってました）多重下請けの下の方は薄給で延々とプログラミングをさせられるブラック企業たちです。まとめると、コミュ力に自信がある人か、劣悪な環境の中ひたすらプログラミングをし続けるのをなんとも思わない人以外は

IT 企業はやめとけ

ということです。（安易に IT 企業ええかな、と思って痛い目見た人がここにいます笑）

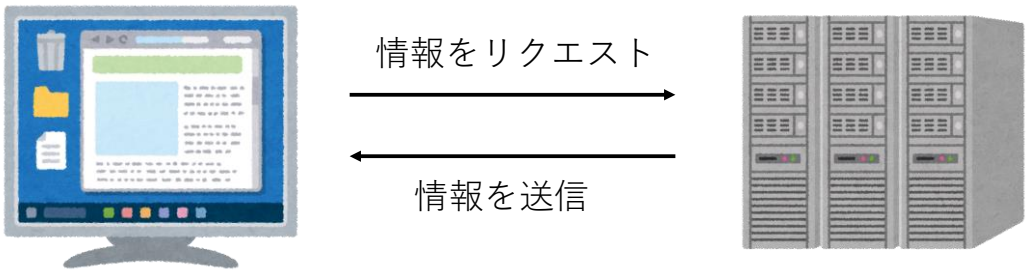
じゃあ何のためにやるんだよ、ということになりますが、私は趣味でやってるので、何のためとかそういうのはありません。卓球するのに理由はありますか。将棋するのに理由はありますか。それと同じです。この資料を通して、あなたをプログラミングの沼に引きずり込みます。

第 1 章 プログラミング・Web 開発の基礎

本章では、プログラミングや Web 開発を行う上で基礎となる部分を概説します。プログラミングのプの字もない章ですが、割と大事なことも書いているので、しっかりついてきてください。

1-1 Web のしくみ

皆さん、Web のコンテンツがどのように表示されるか知っていますか。簡単に言うと以下の図のようになります。



クライアント側
（情報がブラウザに表示され、閲覧できる）

サーバ側
（情報を格納しているファイル類のありか）

私たちが普段目にするのはクライアント側であり、私たちが Web ページにアクセスしたときにはサーバ側に情報をリクエストします。サーバには情報を格納しているファイル類が詰まっているので、そのサーバからクライアントに情報が送信されます。これによって情報を見ることができます。

また、私たちはインターネット経由でブログでコメントを送ったり、EC サイトで決済を行ったりします。その場合はクライアント側の情報をサーバに送り、その情報をサーバが処理します。このイメージを下の図に示します。

情報を送信

クライアント側
（情報をユーザが入力し、送信する）

サーバ側
（ユーザからもらった情報を処理する）

Web システムを作る、ということは、クライアント側で動くプログラムや、サーバ側で動くプログラムを作る、という意味になります。クライアント側で動くプログラムを作ることを「フロントエンド開発」、サーバ側で動くプログラムを作ることを「バックエンド開発」といいます。基本的に執筆者は「フロントエンド開発」の方をよく手がけており、「バックエンド開発」も経験が無いことはないですが、正直あまり身にはついていない、というのが現状です。

※余談ですが、再履バス同好会の Web サイト
http://sairibus.com/
を訪問すると下の方に出てくるブログを見ると、「フロントエンド担当」という人がブログの大部分を書いているのが分かると思いますが、それは執筆者である私です。

1-2 拡張子

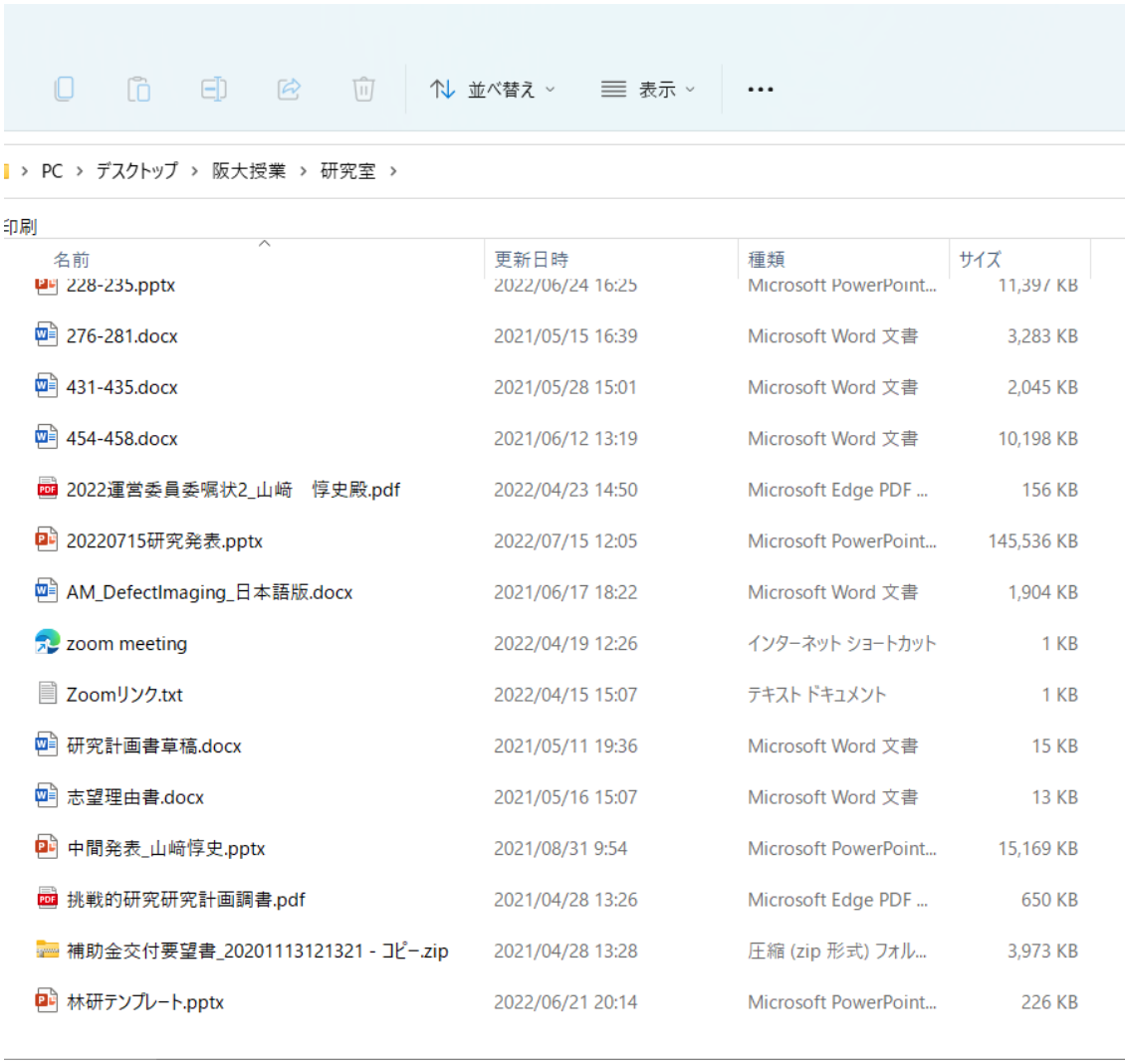
皆さん「拡張子」という用語を聞いたことはありますか。これを知っているとプログラミングだけではなく、これからパソコンを扱う作業をする上でも非常に役に立つので、是非知っておいてください。

拡張子とは、

ファイル名の末尾につく英数字（一般的には 1~4 文字）の記号

となります。とは言っても何のことか分からないと思うので、実際に見てみることにしましょう。

例えば、以下の図に示しているのは、私の PC に入っている、研究室関連のファイルをまとめたフォルダです。



この図を見ると、インターネットショートカットを除くすべてのファイルに対して、拡張子がついています。例えば、ワードファイルであれば「.docx」、パワーポイントファイルであれば「.pptx」、pdf ファイルであれば「.pdf」、テキストファイルであれば「.txt」となっています。

拡張子は、パソコン側がどんな種類のファイルなのかを認識するために使用しています。もし拡張子が無ければ、パワーポイントファイルもワードファイルもテキストファイルも区別がつかないわけです。ワードでパワーポイントファイル開くの嫌でしょう（？）。

次ページによく使用する拡張子を示しています。

拡張子の種類（一般）

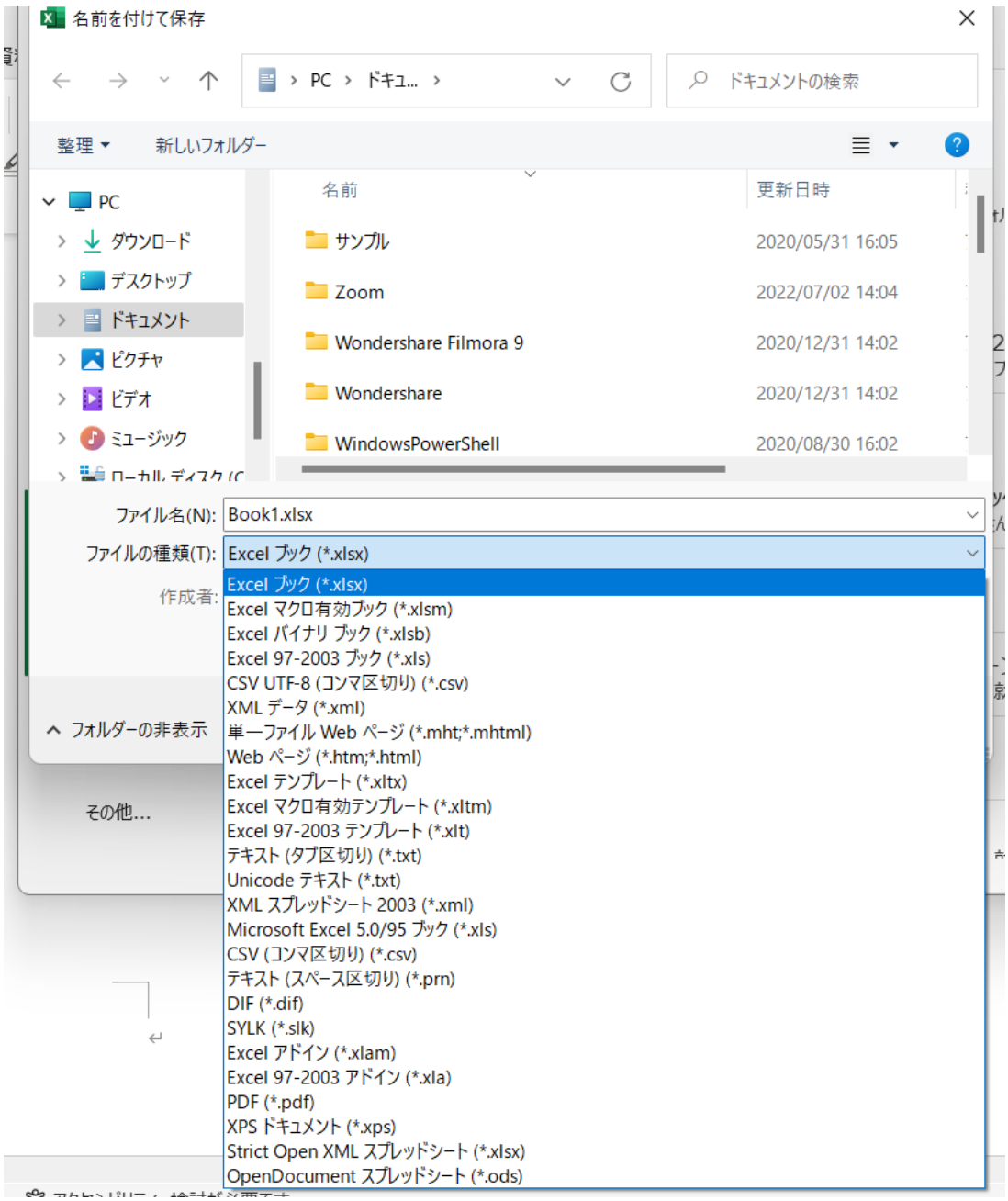
ファイルの種類の説明	拡張子
Word ファイル	.docx
Excel ファイル	.xlsx
PowerPoint ファイル	.pptx
テキストファイル	.txt
CSV ファイル（データ保存によく使われる形式）	.csv
JPEG ファイル（画像ファイルの形式）	.jpg
PNG ファイル（画像ファイルの形式）	.png
MP3 ファイル（音声・音楽ファイルの形式）	.mp3
MP4 ファイル（動画ファイルの形式）	.mp4
ZIP ファイル（複数のファイルを圧縮したもの）	.zip

拡張子の種類（プログラミング関連）

ファイルの種類の説明	拡張子
HTML ファイル	.html
CSS ファイル	.css
JavaScript ファイル	.js
Python ファイル	.py
Python ファイル（Jupyter Notebook で開ける形式）	.ipynb
PHP ファイル	.php
実行ファイル **※メールでこの形式のファイルが送られてきたらウイルスが仕込まれている可能性があり！絶対に実行しないこと！**	.exe

もちろんここで示した拡張子はほんの一部であって、すべてではありません。一般の方は普段の学生生活でもよく使うので、覚えておいた方がよいのではないでしょうか。プログラミング関連の拡張子は、今は覚える必要はありません。この後の章で深く解説しますので、今は「こんなんがあるのだな」という程度でかまいません。

拡張子の知識が生きる場面は、ファイルを保存するときにあります。例えば、エクセルを開いて中身を編集した後、「ファイル→名前を付けて保存→参照」とすれば、以下のような画面が出てきます。



何も考えずにそのまま保存すると、これはエクセルファイルとなり、拡張子が.xlsx となります。しかし、必ずしもエクセルファイルとして保存する必要は全くありません。例えば、上から 5 つめにある「CSV UTF-8(コンマ区切り)」を選択してやると、エクセルで記入したデータがコンマ区切りの形式で保存されます。また、その他の形式でも保存することができます。ただ、拡張子の意味に関する知識なしに保存してしまうと、何ができるかわからないただのゴミファイルが完成してしまうので、お気を付けください。

演習問題 1

Excel ファイルで適当に数字を入れてみましょう。そして、できたファイルを CSV 形式で分かりやすい場所（よく分からなければデスクトップが無難）に保存しましょう。その後、メモ帳を開きます。（検索すれば早いかと）メモ帳の「ファイル→開く」で先ほど保存した CSV ファイルを選択し、メモ帳で開いて、どのようなフォーマットで表示されるか確かめてみましょう。

1-3 ディレクトリと絶対パス、相対パス

この節が本章の最大の山場だと思ってくれて結構です。今節の内容はなかなか理解するのが難しいと思いますが、この後本格的にプログラミングをするのには非常に重要になってくるコンテンツです。

「**ディレクトリ**」という言葉はおそらく情報に触れたことのない人なら知らない言葉だと思いますが、語弊を招くことを恐れずに言うと、ディレクトリとはフォルダのことです。前節で研究室に関連するファイルが研究室フォルダに入っている、という話をしましたが、言い換えると、「研究室に関連するファイルは研究室ディレクトリにある」といえます。

ディレクトリに関連する用語として、「**ルートディレクトリ**」と「**カレントディレクトリ**」という重要な言葉が出てきます。ルートディレクトリとは、

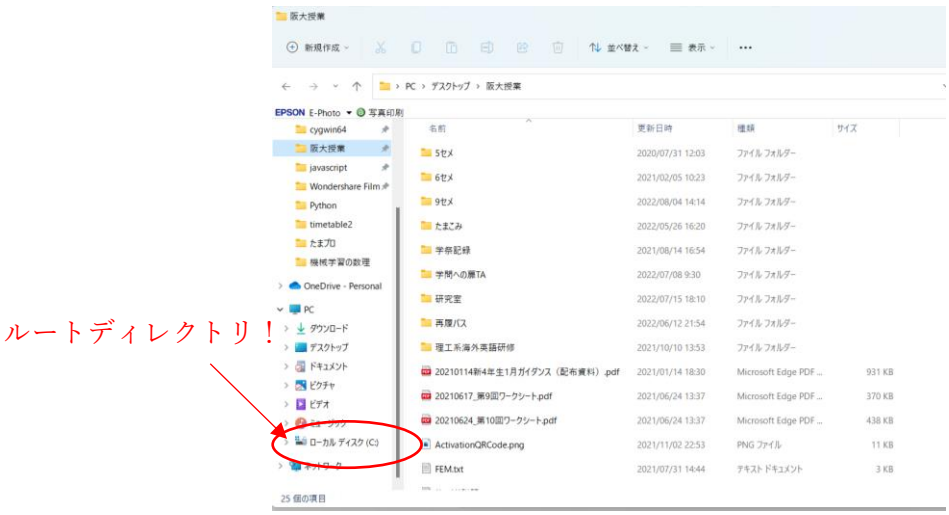
大元となるディレクトリ

です。一方、カレントディレクトリとは、

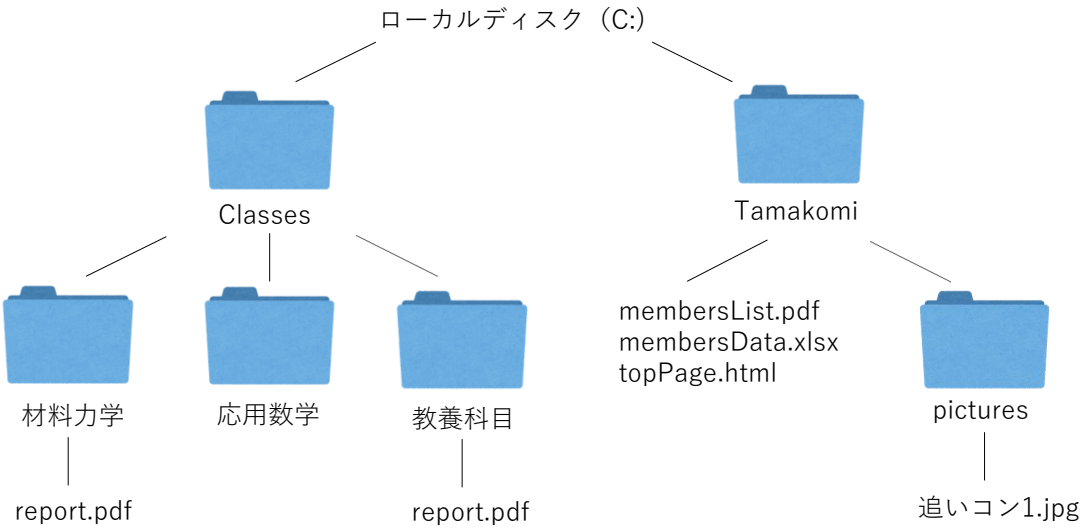
現在開いている（注目している）ファイルがあるディレクトリ

です。

とか言われても「？？？」だと思います。ということで詳しく説明していきます。



あなたが使用している機器がパソコンであれば、エクスプローラー（Windows の場合。フォルダみたいなマークをしているやつです）をクリックすると、上のような画面が出てきます。その左下に、「ローカルディスク(C:)」と書かれたものがあると思います。これこそがルートディレクトリで、パソコン上のすべてのファイルはその配下にあります。パソコン上のファイル構成を極端に簡略化したものを以下に例として示します。



この図に関しては、ルートディレクトリの配下に Classes ディレクトリと Tamakomi ディレクトリの 2 つがあります。そして、Classes ディレクトリの配下には材料力学ディレクトリと応用数学ディレクトリ、教養科目ディレクトリがあります。また、Tamakomi ディレクトリの配下には 3 つのファイルと、pictures ディレクトリが存在します。実際のパソコンのファイル構成はもっと複雑ですが、簡単に説明すると以上のような形でファイルが構成されています。

一方、カレントディレクトリについてはどうなのか、ということになりますが、それは、今開いている（注目している）ファイルによって違います。例えば、上の図で topPage.html というファイルに注目すると、カレントディレクトリは Tamakomi ディレクトリとなります。

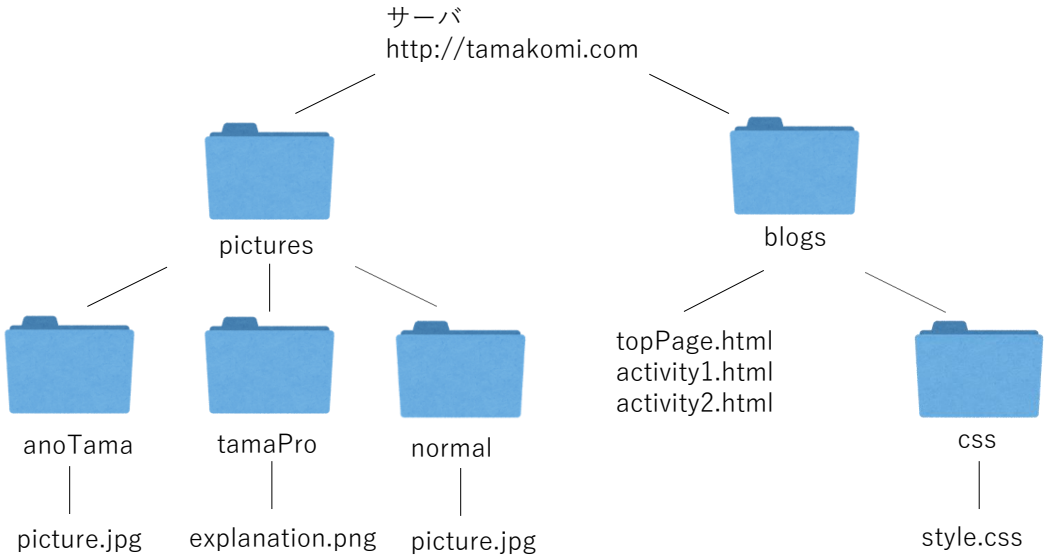
ここまでは、パソコン内の話をしてきましたが、似たような話が Web サイトについてもあてはまります。ここで登場するのが「**ドメイン**」や「**IP アドレス**」といった用語です。ドメインや IP アドレスは、アクセスしたいファイルがあるサーバの場所を指し示すものです。いわばインターネット上の住所を示すもの、といえるでしょう。例えば、再履バス同好会のドメインは”sairibus.com”です。そのドメインで表されるサーバにアクセスするには、“http://sairibus.com“とすればよいです。似たような英数字・記号の並びはインターネットに触れていればおそらく見たことがあるでしょう。また、IP アドレスは、そのドメインを

数字に書き直したもので、例えば”192.168.40.2”みたいな形で表されます。ドメインと IP アドレスは 1 対 1 で対応します。

さらに学ぼう

DNS[Domain Name System]
ドメイン名と IP アドレスを対応づける仕組みのことを指す。

Web サイトの構成の例を以下に示します。



これを見ると、先ほどの図と似ていると思いませんか。つまり、Web サイトの構成においては、ドメインで指し示された場所がルートディレクトリのような役割を果たしている、といえるのです。もちろんカレントディレクトリの概念も健在です。

さて、ここからは絶対パスと相対パスの話をしていきます。まず、**絶対パス**とは、

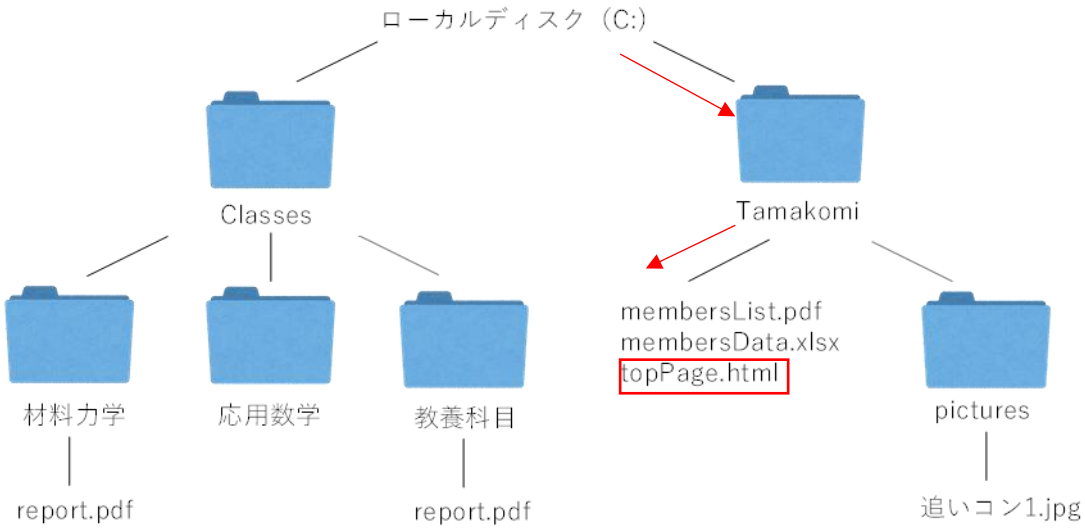
ルートディレクトリを基準として、当該ファイルの存在場所を示すパス

です。一方、**相対パス**とは、

カレントディレクトリを基準として、当該ファイルの存在場所を示すパス

です。

言葉で説明するだけでは分からないので、実際の例を見てみましょう。



先ほどの図をもう一度持ってきました。Tamakomi ディレクトリにある topPage.html の絶対パスを書くと、

C:¥Tamakomi¥topPage.html

となります。

※¥は環境によってはスラッシュ（/）となります。タイピングするときは
C:/Tamakomi/topPage.html
というように入力しますね。

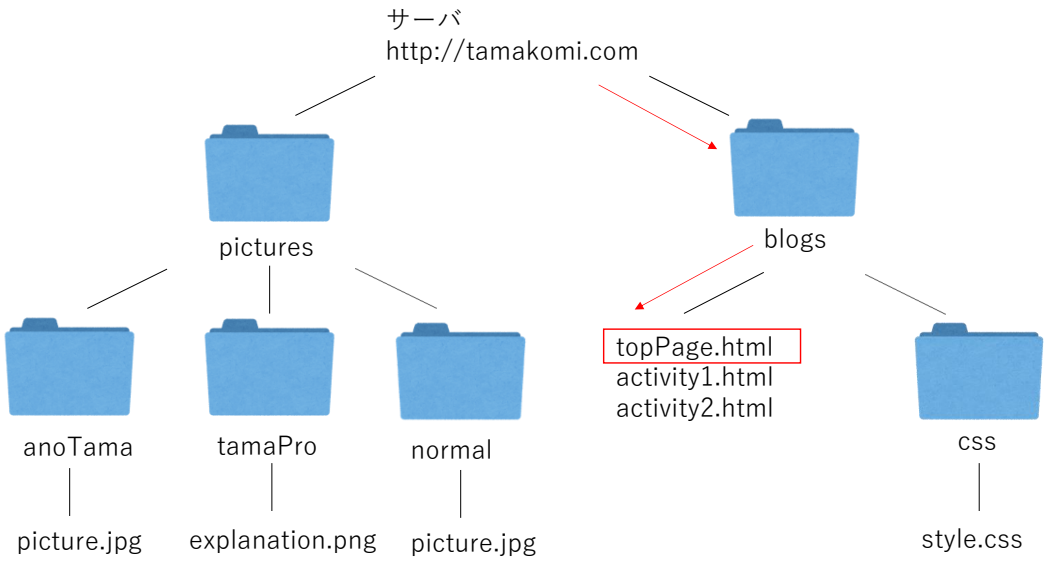
¥(ディレクトリ名)、あるいは/(ディレクトリ名)で今のディレクトリの１層下のディレクトリを指ししめします。これをルートディレクトリから目当てのファイルがあるディレクトリまで並べ続けて、最後は/(ファイル名と拡張子)を付ければ完成です。

全く同じことを Web ページへのアクセスにも適用できます。http://tamakomi.com の blogs ディレクトリにある topPage.html にアクセスしようと思ったなら、

http://tamakomi.com/blogs/topPage.html

とすることで可能になります。

普段私たちが Web ページにアクセスするのに使用しているのは絶対パスということになります。



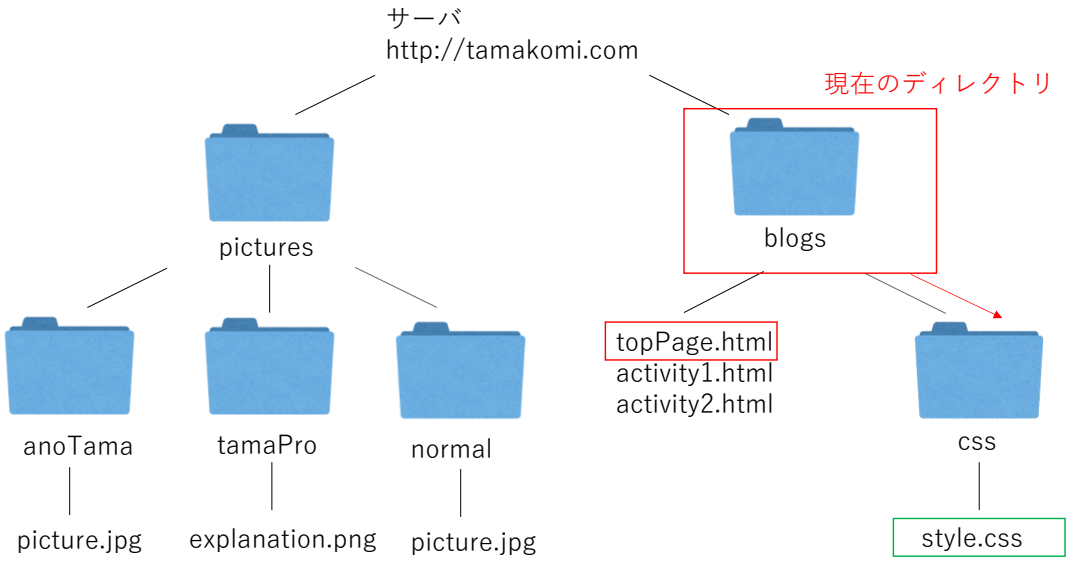
次に、相対パスの例を見てみます。相対パスを用いる場合、2 つ重要な記号があり、それは

./　…　現在のディレクトリを指し示す記号（省略可能）

../　…　1 階層上のディレクトリを指し示す記号

となります。

例えば、Web ページの例を使用して、blogs ディレクトリにある topPage.html が style.css を参照するとします。（今後の章で詳しく説明しますが、HTML ファイルはよく css ファイルを参照します）その際、style.css の場所を相対パスで与えてやらなければなりません。

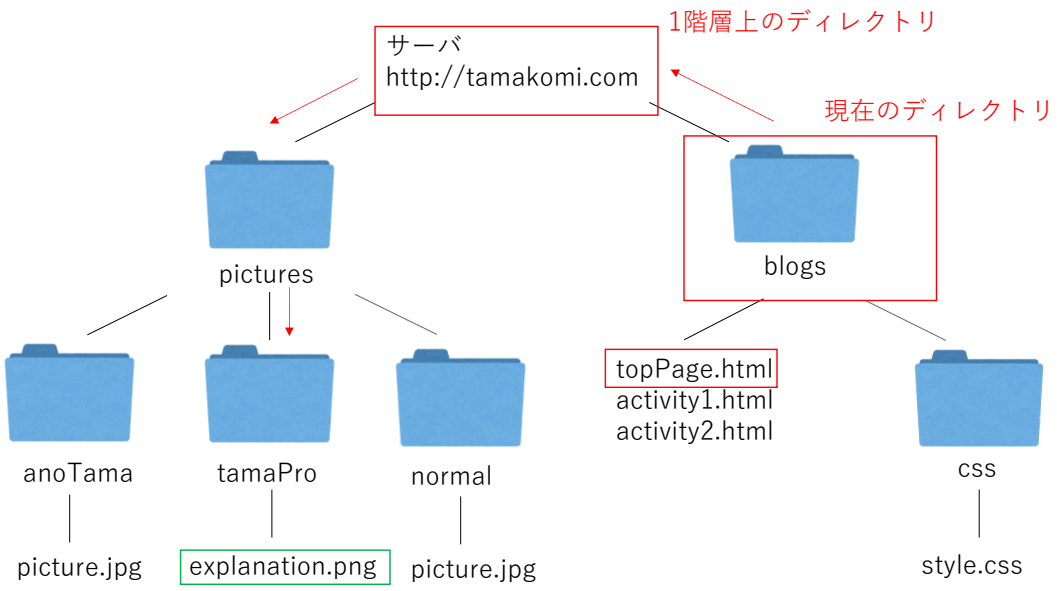


現在注目しているファイルは topPage.html で、それは blogs ディレクトリにあります。参照したいファイルは blogs ディレクトリの直下にある css ディレクトリにある style.css というファイルです。ここから、相対パスを記述すると、

./css/style.css

となります。冒頭の./は省略できます。

それでは、次の例として、同じく topPage.html が tamaPro フォルダにある explanation.png を参照したいとします。



先ほどと同様、カレントディレクトリは blogs ディレクトリです。そこから explanation.png にたどり着こうと思うと、1 階層上のディレクトリに上がる必要があります。そこで登場するのが、先ほど説明した"../"という記号です。1 階層上に上がることができれば、あとはその直下の pictures ディレクトリ、さらにその直下の tamaPro ディレクトリ、というふうにして、最終的に explanation.png というファイルに到達することができます。相対パスで記述すると、

../pictures/tamaPro/explanation.png

となります。

演習問題 2

パソコン中のどのファイルでもよいので、左クリックして「パスのコピー」をクリックしてみましょう。その後、どこでもいいのでそのパスをどこかに貼り付けてみましょう。

演習問題 3

先ほど示した http://tamakomi.com のファイル構成の図を参照してください。カレントディレクトリが”normal”であった場合、blogs ディレクトリにある activity1.html への相対パスを書いてみましょう。

1-4 プログラミング言語

本節では、Web に関連するプログラミング言語の種類について述べていきます。1-2 節で、フロントエンド開発とバックエンド開発の 2 種類があるよ、という話をしました。フロントエンド開発に使われるのが”HTML”, ”CSS”, “JavaScript”という言語で、バックエンド開発に主に使われるのが”Ruby”, “PHP”, “Python”などとなります。それぞれの言語の説明を以下の表に示します。

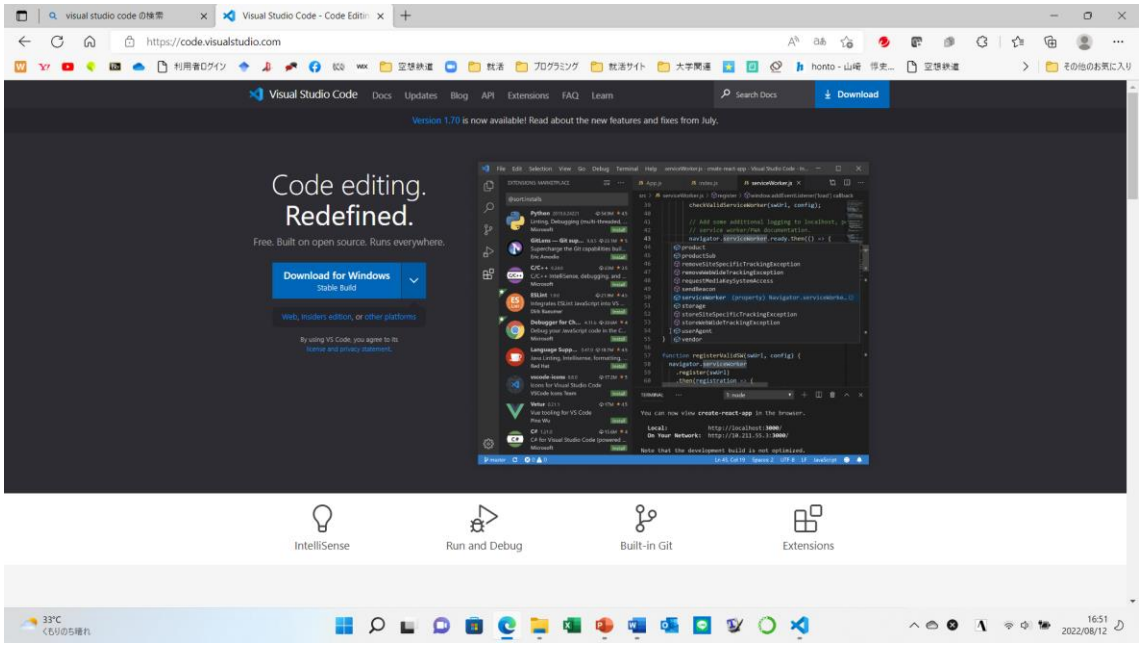
言語	説明
HTML	Web ページの構成を表すための言語。Web 開発したければ必須。マークアップ言語という分類に属し、この言語を用いて計算などは行えないため、厳密にはプログラミング言語ではない。
CSS	Web ページのデザインを形作る言語。スタイルシート言語に属し、これもプログラミング言語ではない。
JavaScript	Web ページに動きを実現できるプログラミング言語。執筆者の作ったサイトやゲームはだいたいこれで動いている。
Ruby	日本発のサーバ側開発用言語。執筆者は触ったことがない。
PHP	サーバ側開発用の言語。Web サイトを作りたい人に人気のサービスとして WordPress があるが、その WordPress は PHP が使われている。再履バス同好会の Web サイトも WordPress でできている。
Python	AI,機械学習ブームの高まりにより人気になっている言語。Web 開発に関して言えば、“Django”, “Flask”といったフレームワークがあり、サーバ側の開発に使用できる。

第 2 章 HTML の基本

本章では、Web のすべての基本となる HTML を触ってみます。この章の内容をマスターするだけで、機能やデザインとしては不十分ながら一応 Web サイトは作れてしまいますので、もうあなたは Web サイト作成できる人材、と胸を張って言うことができます。(胸張って言えるかといわれたら微妙かもしれませんが)

2-1 Visual Studio Code のインストール

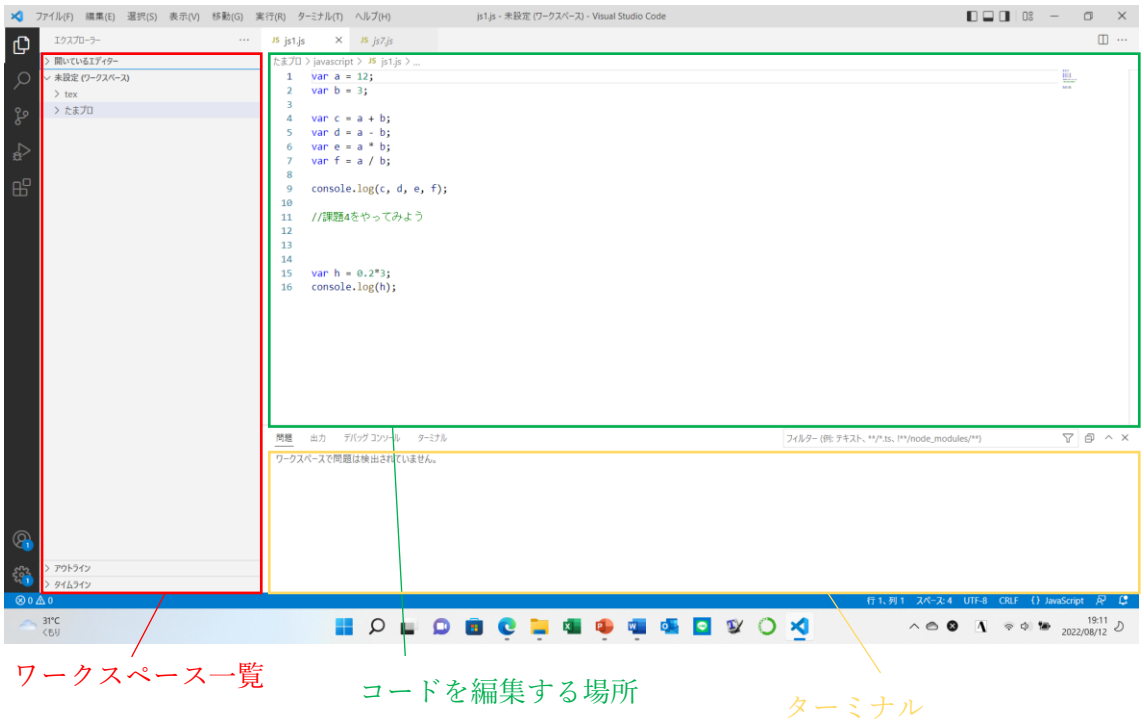
プログラミングを行うには、コードエディタというものが必要になります。そのコードエディタとしておすすめできるのが、この Visual Studio Code です。入力の補助も自動でしてくれる優れもので、プログラミングに適しています。ということでまずはこのソフトをダウンロードしてみましょう。



Let's download it!

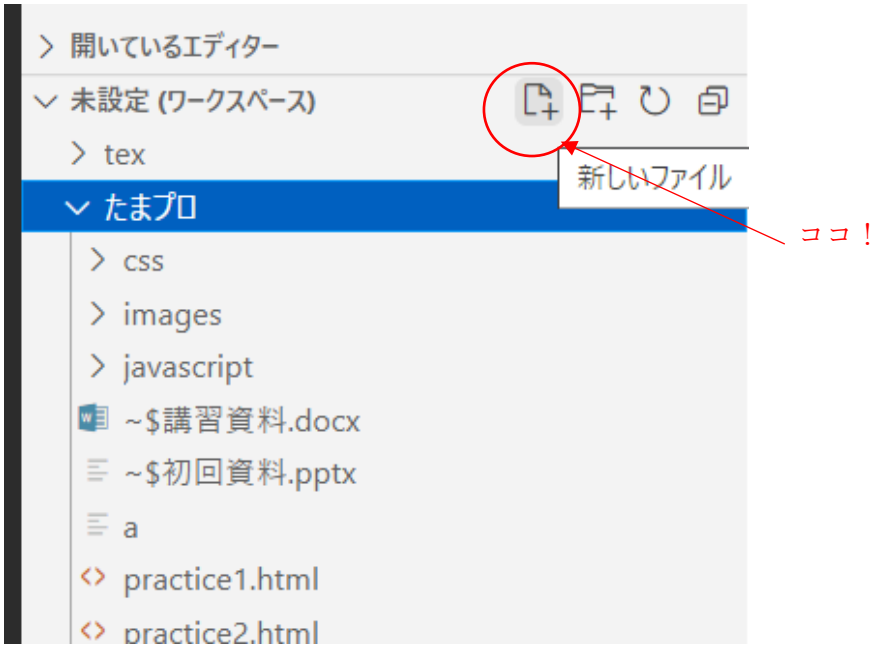
ちなみに、必ずしも Visual Studio Code を使用する必要はありません。お気に入りのエディタがあればそれを使えば良いです。例えば、執筆者は普段 Notepad++を使用しています。もし何も使えそうなのを持っていなければ、Visual Studio Code を入れてみましょう。

インストールして立ち上げると日本語になっていない場合があります。英語でやるんだ、という意志があればそのまま使っても良いのですが、日本語にしたい場合は上のメニューから「View→Command Palette」を選択し、出てきたところに”Display”と入力します。すると、” Configure Display Language”というのが出てくるので、それをクリックします。すると、Select Display Language と表示されるので、候補から”Install additional languages”を選択してください。すると、左側に拡張機能一覧が表示されるので、日本語版のものを選択し、インストールしてください。インストール後、再起動すれば、表示が日本語になっていると思われます。



インストールが完了すれば、まずはワークスペースを追加してみましょう。まずデスクトップ上（必ずしもデスクトップ上でなくてもよいですが、わかりやすいところ）にこの講座のためのフォルダを作りましょう。（ここでは、たまプロという名前を付けます）その後、Visual Studio Code でそのフォルダをワークスペースに登録します。

ただ、今のところ新しいフォルダの中身は空です。そのため、コンテンツを作っていく必要があります。そこで、新しいファイルを作成しましょう。



ココをクリックすれば、新しいファイルの名前を入力するよう要求されるので、template.html と入力してください。ここで気をつけるのは、**拡張子までしっかり入力する**、ということです。Visual Studio Code はいろんな言語を扱えますので、どの言語のファイルになるかを示すよう、拡張子を予め入力する必要があります。そうすると、コードを編集する画面が各言語に合わせた仕様になるので、非常に編集しやすくなります。逆に拡張子を入力しないと、何か分からないただのゴミファイルを生産するだけになります。

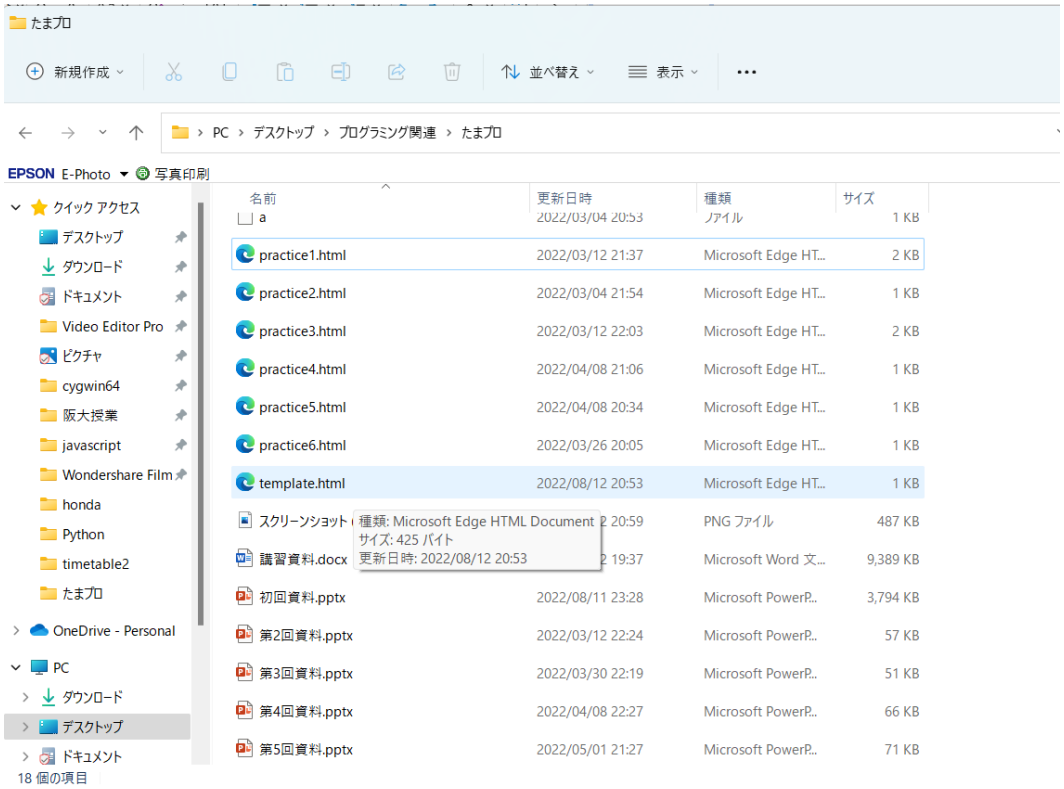
ファイルが作れたら、以下の内容をコピペしてコード編集の画面に貼り付けてみましょう。

template.html

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <header>

        </header>
        <nav>
```

```
        </nav>
        <div id="contents">

        </div>
        <footer>

        </footer>
    </body>
</html>
```

ここまでできたら、ファイルを保存しましょう。ファイルの保存は、「ファイル→保存」でできます。また、Windows のショートカットキーを使うと、「Ctrl+S」でも保存ができます。その後、Visual Studio Code ではなく、ファイル管理をするアプリ（Windows ならエクスプローラー）でファイルのあるディレクトリに到達します。下の図のようになれば OK です。



そして、template.html をクリックすると、デフォルトのブラウザが開きます。そして、template.html の内容が表示されます。

嘘おっしゃい！何も表示されないじゃないか！

と思うかもしれませんが、何も表示されなくて OK です。だって、template.html には内容物を入れる入れ物のようなものは用意はしてありますが、肝心の内容は何も入っていないんですから。

2-2 HTML ファイルのテンプレート

ここからは、先ほどのファイルについて説明していきます。

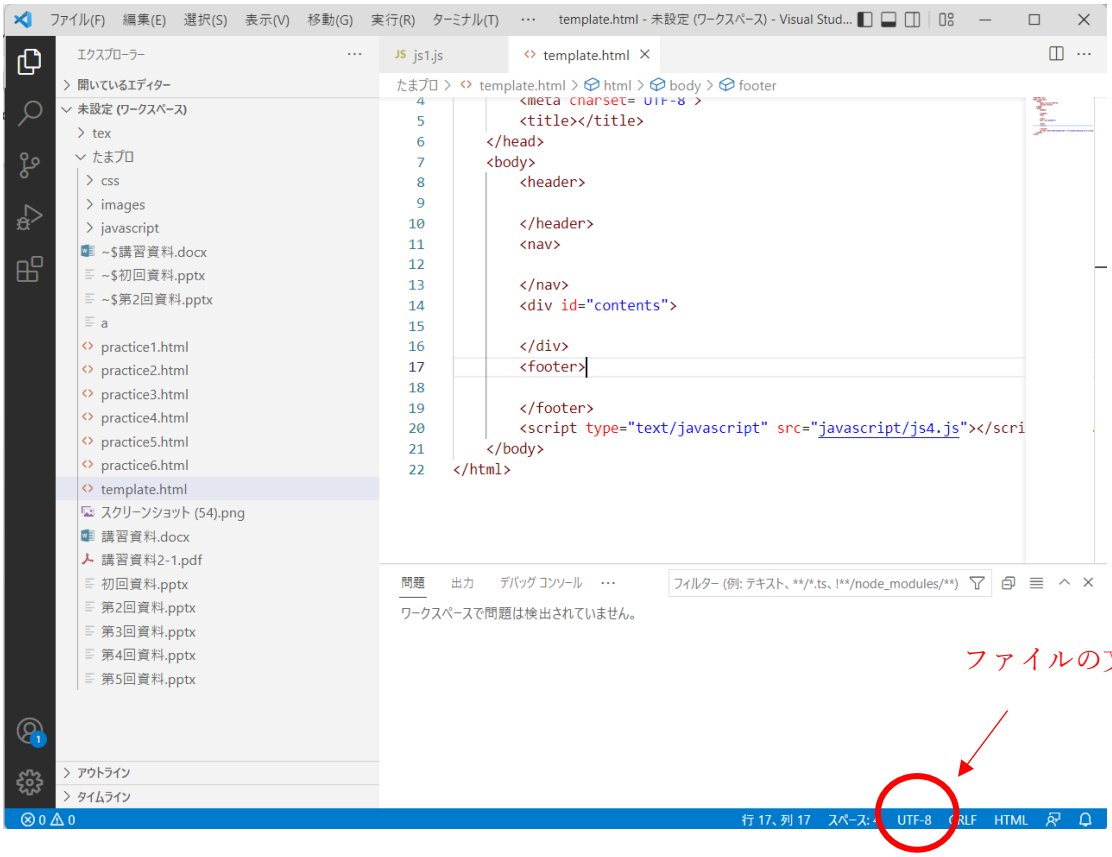
まず、1 行目に出てくる<!DOCTYPE html>ですが、これは **DOCTYPE 宣言**と言って、簡単に言うと、このファイルが HTML で書かれていることを宣言するものとなります。これ以上踏み込みませんが、HTML ファイルを作成するときは最初にこれがいる、ということだけ覚えていれば結構です。

次に、2 行目に注目してください。<html lang="ja">とあると思います。また、最終行には</html>とあると思います。<○○>となっているのは**タグ**と呼ばれ、ページの構造を形作る要素になりますが、タグの多く（全てではない）は<○○>と</○○>で内容を挟み込む構造となっています。html タグは、<html>~</html>に HTML ファイルのコンテンツが入っていることを示すタグで、必ず必要となります。

また、<html lang="ja">の lang="ja"では、タグに対して**属性**を設定しています。属性とは、タグで表された要素の振る舞いを決定づけるものです。ここでの lang 属性は、一応日本語で書かれてあることを宣言するためにつけられていますが、必ずしも付ける必要がないです。

さて、その次に登場するのが head タグです。<head>と</head>で挟まれた領域に入るコンテンツは、Web ページの情報（タイトル等）が入ります。この領域に入るコンテンツは、Web ページに直接表示はされません。

head タグの中身を見ていきましょう。すると、最初にあるのが、<meta charset="UTF-8">です。これは丸暗記・コピペしてもらって構わないのですが、一応説明すると、文字コードの指定になっています。テンプレートの記述は、「このファイルは文字コードが UTF-8 で書かれていますよ」、と宣言しています。文字コードは、UTF-8 の他、Shift-JIS などが存在し、Shift-JIS を使用したい場合は、<meta charset="Shift_JIS">と書けば良いのですが、HTML では非標準となっており、UTF-8 を使用することが推奨されます。



Visual Studio Code の右下で、現在の文字コードが確認できるので、それと meta タグの charset 属性を一致させてください。そうしないと、いわゆる「文字化け」が発生します。

head タグの中身として次に登場するのは title タグです。その名の通り、Web ページのタイトルを指定するタグです。

演習問題 4

template.html をコピーし、同じディレクトリに貼り付け、別名で保存しましょう。その後、新しいファイルに対して<title>と</title>の間に何かしらのテキストを書き込んでみましょう。その後、そのファイルをブラウザで開きましょう。そして、どこが変化したかを見てみましょう。

head タグの終わりの直後に、body タグが現れますが、この body タグの部分に、Web ページとしてブラウザに表示される内容を入れていきます。template.html の中には、header, nav, div, footer の 4 つのタグがあります。div タグについては少なくとも第 2 章では全く意味をなさないタグ（次章で真価を発揮）なので、ここでの説明は省略するとして、残りの 3 つのタグについての説明を簡単に行うと以下のようになります。

header タグ
・・・ヘッダ。Web ページのタイトル等を記載。
nav タグ
・・・ナビゲーション。クリックすることで利用者が目当てのコンテンツのページに飛ぶことができる。
footer タグ
・・・フッタ。Web ページの一番下にある部分。

「Web ページのタイトル等を記載、って先ほど説明があった head タグ配下の title タグと何が違うんだよ?」と思われるかもしれませんが、body タグ配下にある内容は Web ページのコンテンツとして表示される内容なので、Web ページに出てきて利用者の目につくタイトル、と考えると良いでしょう。とはいってもこれらの説明では漠然としていてなかなか理解がしにくいと思うので、演習問題 5 に取り組みましょう。

演習問題 5
スイッチサイエンス社のホームページ（https://www.switch-science.com/）
を見て、ヘッダ、ナビゲーション、フッタがそれぞれどこにあたるか考えてみましょう。

説明した 3 タグは、必ずしも使用する必要はありませんが、コンテンツの構成を明確にするために、よく使用されます。
ここまでで、ひな形となる部分はできました。次節では、コンテンツの制作を行っていきます。

2-3 Web ページのコンテンツ作成

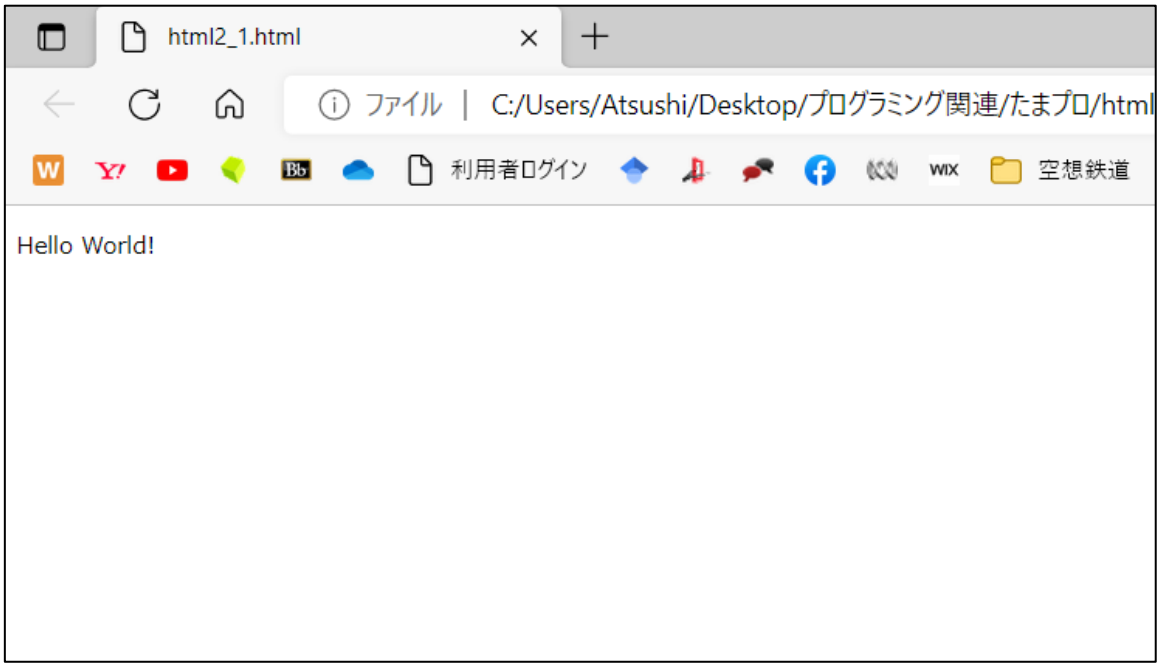
この節では、Web ページの中身について扱っていきます。まずは p タグという、基本のタグを扱います。template.html をコピーして html2_1.html というファイルを作り、div タグの中に p タグを以下のように書き込んでみましょう。
（※ファイルは Atsushi-Yamasaki8249/tamaPro (github.com)からコピペして利用できるようにしてあるはずです。）

html2_1.html

```
~省略~
    <div id="contents">
```

```
        <p>Hello World!</p>
    </div>
                                    ~省略~
```

完成したら、そのページをブラウザから見てみましょう。



“Hello World!”と表示されれば OK です。おめでとうございます！あなたは Web ページを作れるようになりました。あとはこのファイルを借りたサーバに入れて、インターネット経由でアクセスすれば、自分の作ったこのページが誰でも見られるようになります。（需要無いわ）

お次は h1 タグについて学習します。同じく template.html をコピーして新しいファイル html2_2.html を作ります。そして、以下のように書き込んでみましょう。

html2_2.html

```
                                    ~省略~
    <div id="contents">
        <h1>HTML の基礎</h1>
        <p>本章では、HTML の基礎を学びます。</p>
    </div>
                                    ~省略~
```

このように、h1 タグは、見出しを形作る要素です。同じく、見出しを作る要素として、h2, h3, h4, h5, h6 タグがあり、これらは見出しで表されたコンテンツの中の小見出し、という用途で使用します。

次は br タグについて学習します。br タグは、改行をするタグです。使用例を見てみましょう。

html2_3.html

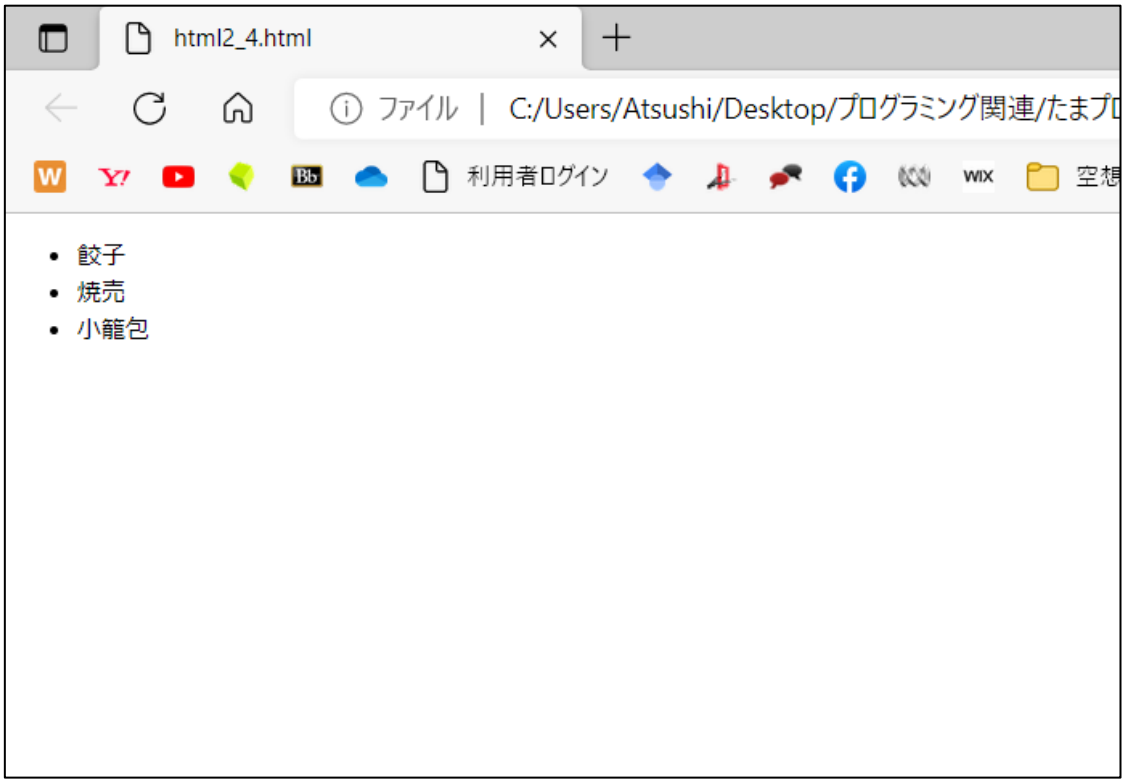
```
                                    ~省略~
        <div id="contents">
            <h1>HTML の基礎</h1>
            <p>本章では、HTML の基礎を学びます。<br>
            Web ページを作れるようになりましょう。</p>
        </div>
                                    ~省略~
```

br タグが無いと、スペースの許す限り永遠に改行せずに文章が続いてしまいます。改行を行って見栄えを整えることができる便利なタグになっております。あと、お気づきだと思いますが、</br>は不要です。

　次に使用するのは、ul, li, ol タグです。実例を見てみましょう。

html2_4.html

```
                                        ~省略~
        <div id="contents">
            <ul>
                <li>餃子</li>
                <li>焼売</li>
                <li>小籠包</li>
            </ul>
        </div>
                                        ~省略~
```



このように、箇条書きリストができました。

演習問題 6

html2_4.html について、ul タグを ol タグに置換して保存しましょう。そして、ブラウザからページを見てみましょう。どこが先ほどと異なるでしょうか。

次に登場するのが、table タグ。これも使い方を見てみましょう。

html2_5.html

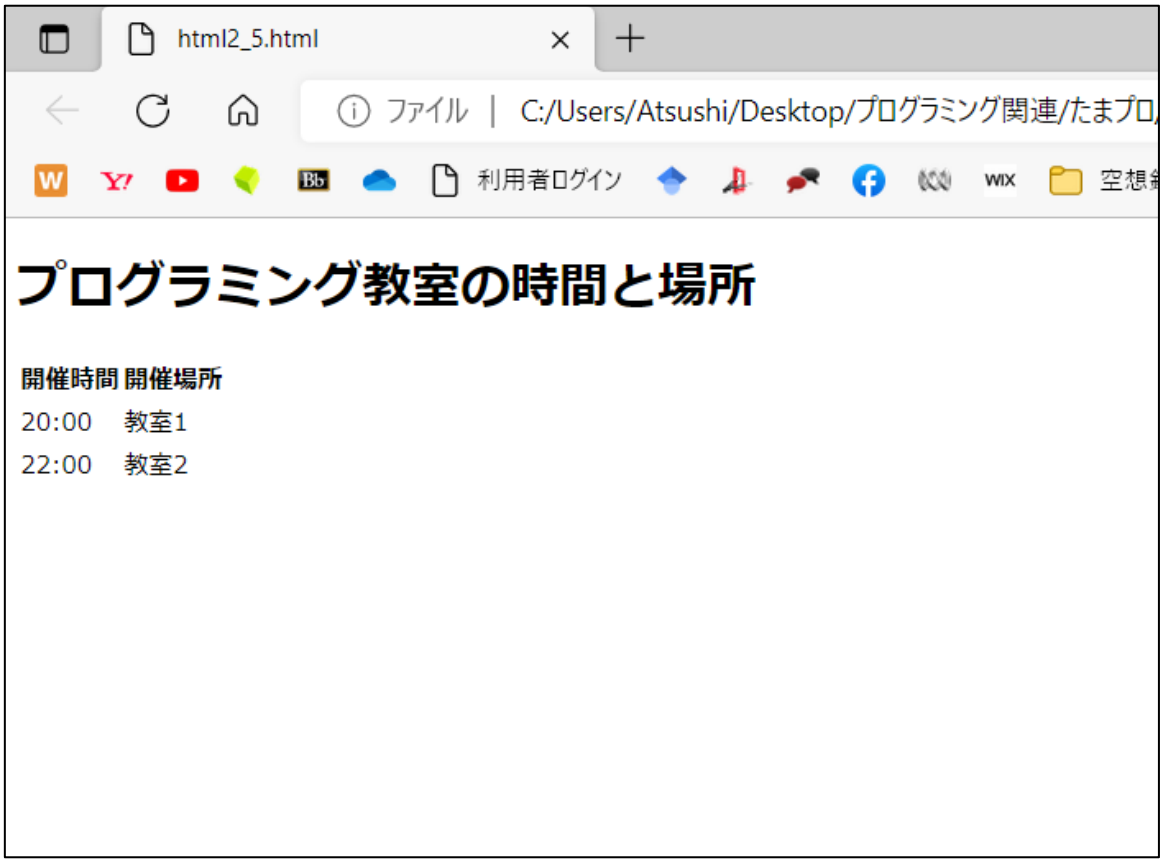
```
~省略~
<div id="contents">
    <h1>プログラミング教室の時間と場所</h1>
    <table>
        <tr>
            <th>開催時間</th>
            <th>開催場所</th>
        </tr>
```

1 行目

```
            <tr>
                <td>20:00</td>
                <td>教室 1</td>
            </tr>
            <tr>
                <td>22:00</td>
                <td>教室 2</td>
            </tr>
        </table>
    </div>
                                ~省略~
```

2 行目

3 行目



このように、表が完成しました。とはいっても罫線が入っておらず表っぽくはないですが、table, tr, th, td タグを用いて表を作ることができます。table タグの中に tr タグを行数だけ入れて、さらに tr タグの中に th タグまたは td タグを入れます。ここで、tr タグの中に入る th タグと td タグの合計（列数）は、全ての行で同じである必要があります。th タグと td タグの違いは、th タグが見出しに使用され、td タグがコンテンツに使用される、というも

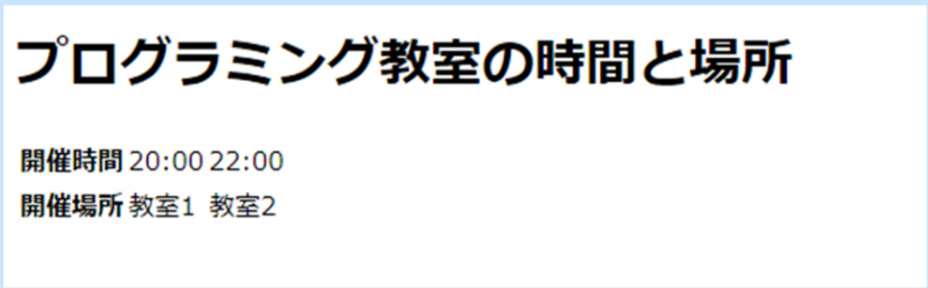
のです。

まとめると以下のようになります。

table タグ
・・・表を作りたいときに置く。配下に tr,th,td タグを置く。
tr タグ
・・・表の 1 行分のコンテンツであることを示すタグ。
th タグ
・・・表の 1 要素を示すタグ。見出しに使用される・
td タグ
・・・表の 1 要素を示すタグ。コンテンツに使用される。

演習問題 7

html2_5.html を書き換えて、以下のように 1 行目に開催時間、2 行目に開催場所が表示される表を作ってみましょう。



2-4 リンクと画像の挿入

次に、他ページへのリンクを作っていきます。ここで、1 章でやったディレクトリと絶対パス・相対パスの内容が活きてきます。

template.html を元にして作った html2_6.html を、html2_5.html と同じディレクトリ（ここ重要！）に置きます。

html2_6.html

```
~省略~
<div id="contents">
    <p>我々は、社会貢献のためにプログラミング教室を開きます。</p>
    <p>開催場所は<a href="html2_5.html">こちら</a></p>
</div>
```

```
~省略~
```

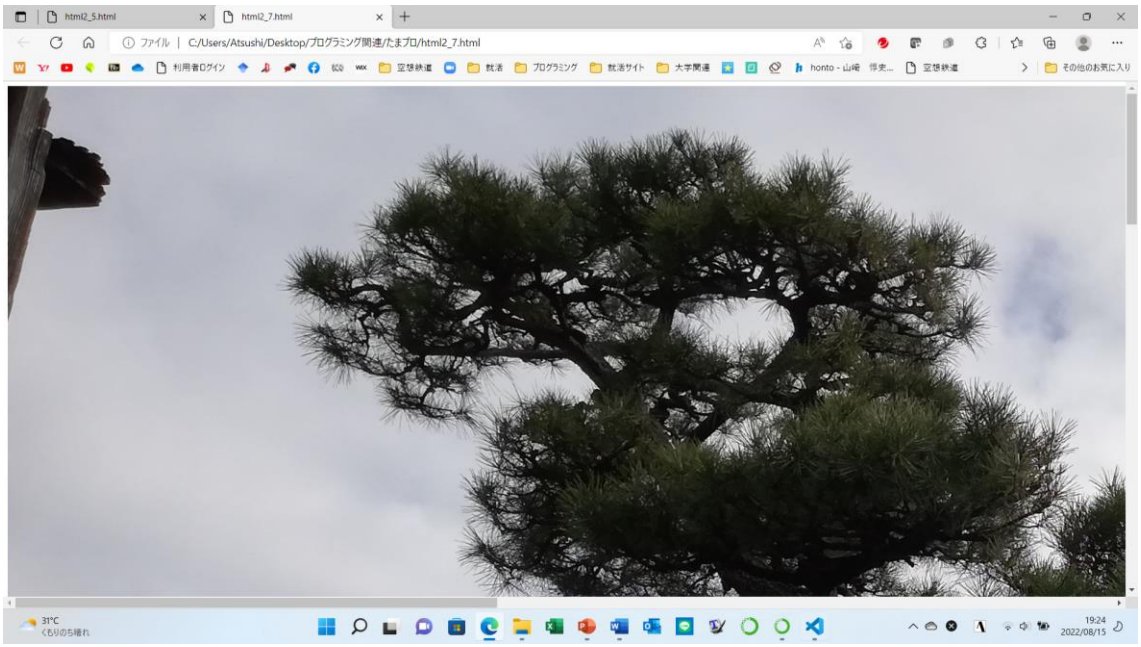
こちら、というところをクリックしたくなると思います。実際クリックすると、html2_5.html の内容が表示されます。

リンクを作ることができるのが、a タグです。a タグ中の内容をクリックすると、a タグの href 属性で指定された場所に移動します。今回は、リンクをクリックすると、html2_5.html の内容を表示させたいです。このとき、今開いている html2_6.html と html2_5.html は同じディレクトリにあるので、相対パスは、“./html2_5.html“となります。./は省略可能です。この相対パスを href 属性に指定します。その際、””で囲ってやることを忘れずに。

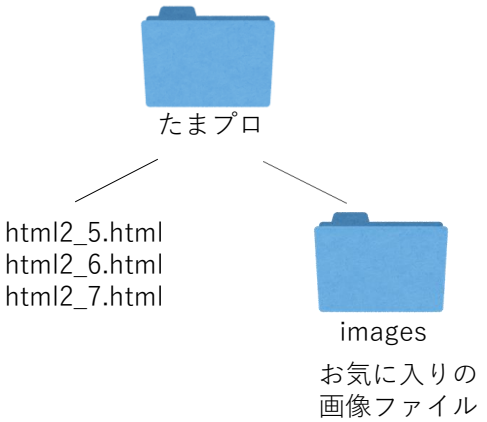
次に、Web サイトに画像を挿入する方法を学びます。これも実例を見てみましょう。

html2_7.html

```
~省略~
    <div id="contents">
        <img src="images/IMG_20220115_094346.jpg">
    </div>
~省略~
```



今この演習を行っているディレクトリ上で新しく images フォルダを作ってみましょう。パソコン上のお気に入りの写真をコピーしてその中に入れてみましょう。そして、html2_7.html に img タグを設置します。そして、src 属性としてその画像へのパスを指定します。ここで、この画像は、html2_7.html があるカレントディレクトリの 1 階層下にある images ディレクトリ中にあるので、相対パスは”images/(画像ファイルの名前と拡張子)となります。



「画像でかっ！」と思うかもしれませんが、お気に入りの画像ファイルが大きければこう

なります。もちろん、画像の表示サイズの調整もできるのですが、これは本章のレベルを超えるので、ここでは触れません。とりあえずはこういうものだと思って進みましょう。

演習問題 8

html2_7.html を書き換えて、画像をクリックすると html2_5.html のコンテンツが表示される Web ページを作りましょう。

2-5 インデント

とりあえず本章の本質的な内容はもう終わりました。この節は補足的な内容になりますが、これからプログラミングを極めるには大事な内容になりますので、しっかり学んでください。

以下に html2_7.html の編集画面を示します。

この編集画面を見て気づいて欲しいのは、head タグが左端ではなく、少し離れた部分にあることです。他のタグでも似たようなことをしています。このように書くことで、要素の始めと終わりが明確になります。そのため、プログラムミスにも気づきやすいです。今回はHTML の場合ですが、今後 JavaScript や Python などを学んでいく際にも、インデントを設けることでプログラムの構造を見やすくすることが大切になります。(Python を学んだことがある人ならご存じかと思いますが、Python はインデントの有無によってプログラムの構造や意味が変わってきます)

肝心のインデントの作り方ですが、左端にカーソルを置いた状態で Tab キーを置くことでインデントが設けられます。一方、インデントを削除したい場合は 1 行の内容が書かれた部分の左端にカーソルを持って行った上で、Shift+Tab キーを押せばよいです。複数行同時にやりたければ複数行の内容を同時に選択した上で Tab キー、または Shift+Tab キーを押すことで、複数行同時のインデントの入力・削除ができます。

第 3 章 CSS による Web ページの装飾

本章では、スタイルシート言語である CSS を用いて、Web ページに装飾を施していきます。これにより、質素だった Web ページに彩りが生まれたり、HTML 単独ではできない機能を実現できたりします。

3-1 HTML への組み込み

CSS の基本的な書き方は、

“プロパティ名： 値;”

となります。プロパティ名とは、例えば文字の色とか要素の配置とかを表す英語で書かれた記号であり、値とはプロパティに対してどんなものを指定するか（具体的な色や具体的な配置）を表します。

HTML ファイルの特定の要素に対して CSS を適用するには、style 属性を使用します。具体的には以下のように書きます。

<(タグ名) style=”プロパティ名: プロパティの値;”>

実際に例を見てみましょう。

html3_1.html

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <header>

        </header>
        <nav>

        </nav>
        <div id="contents">
            <p style="color: #ff0000;">Hello World!</p>
        </div>
```

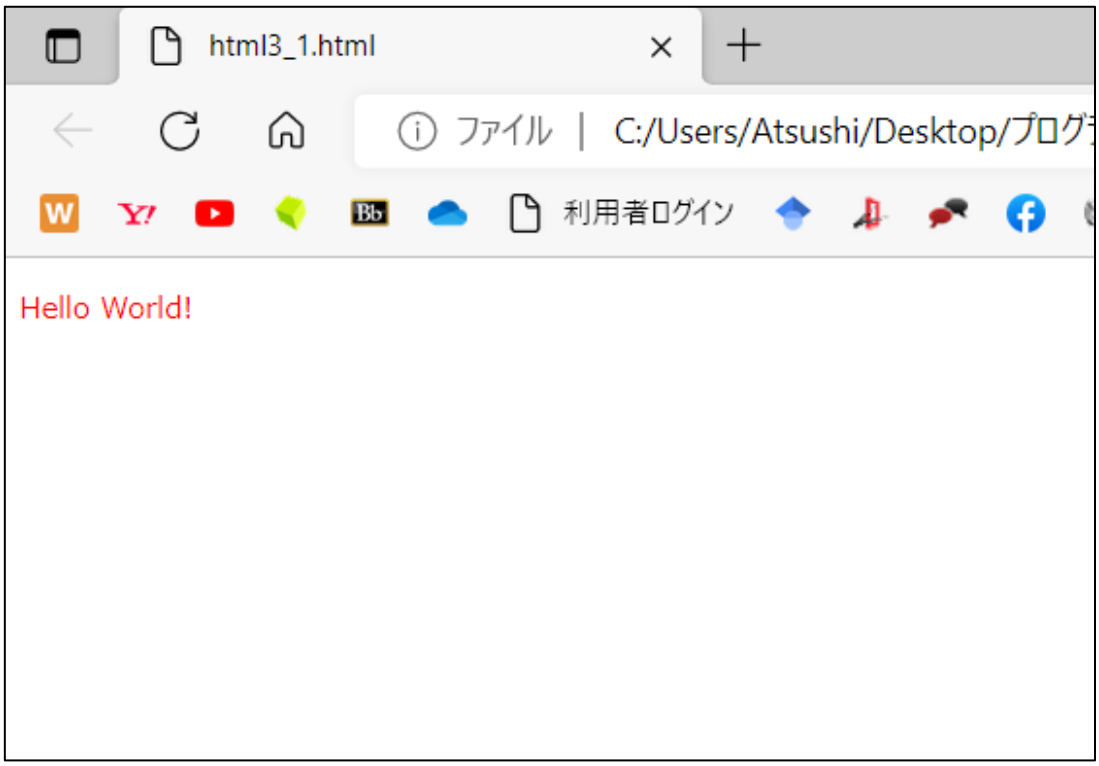
```
        <footer>

        </footer>
    </body>
</html>
```

html2_1.html の p タグに対して、style 属性を追加しただけです。

これを書いていると、Visual Studio Code では、style 属性の指定の時、#ff0000 の左に赤い何かが出てきます。特別な操作を行わなくても登場しますので、なんじゃこりゃ、と思うかもしれませんが、故障ではありません。なぜ赤い四角が出てくるかは本節のこの後で分かりますし、次節でより詳しく解説します。

```
13          </nav>
14          <div id="contents">
15              <p style="color: #ff0000;">Hello World!</p>
16          </div>
17          <footer>
```

ブラウザで html3_1.html を開くと以下のようになります。



文字が赤くなっていることがおわかりだと思います。実は、html3_1.html では、文字を赤色にするよう指定していたのです。赤色の指定をしていたから、Visual Studio Code で赤い

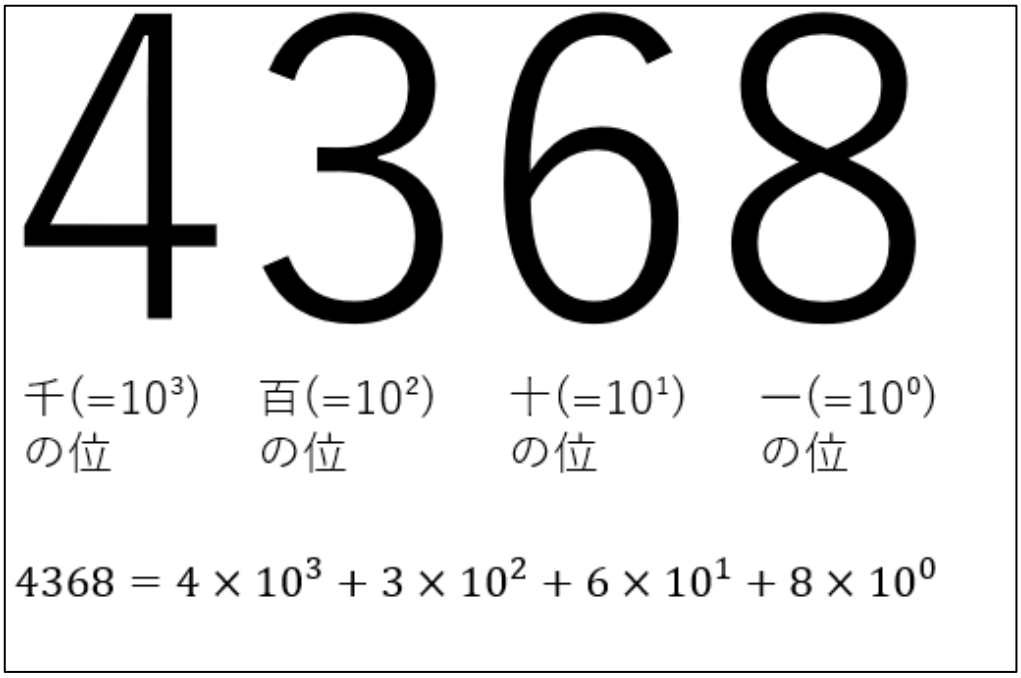
四角が出てきたのです。ちなみに、

```
<p style="color: red;">Hello World!</p>
```

としても同じ結果になります。じゃあなんでただ赤を表現したかったのに#ff0000 とかいう訳の分からないコードで指定するのか、そもそも#ff0000 とは何なのか・・・。それは次節で説明します。

3-2 16 進法とカラーコード

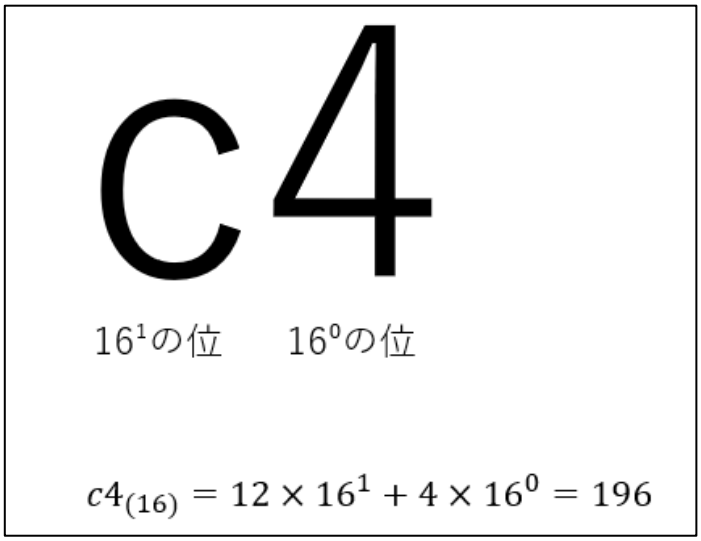
まずは 16 進法について解説していきます。普段私たちが使っている数は 10 進法です。0,1,2,…9 と順に増えていき、10 になると位があがります。



上のように、全ての数は（10 の n 乗）×(0～9 のうちのいずれかの整数)の和で表されます。

同様のことが 16 進数にも当てはまります。10 進数は 0~9 の 10 種類の文字を用いて数を表しているのに対し、16 進数では 0~9,A~F の 16 種類の文字を用います。0~9 は 10 進数の 0~9 と同じで、A は 10、B は 11、C は 12、D は 13、E は 14、F は 15 にそれぞれ対応します。10 進数で 16 を 16 進数で表すと、”10”となり、位が上がることが分かります。読み方は「いちぜろ」であって、決して「じゅう」ではありません。(10 なんてどこにもない！)アルファベットに関しては、少なくともプログラミングに使う場合においては大文字でも小文字でもどちらでもよいです。

16 進数で表された数の例を見てみましょう。



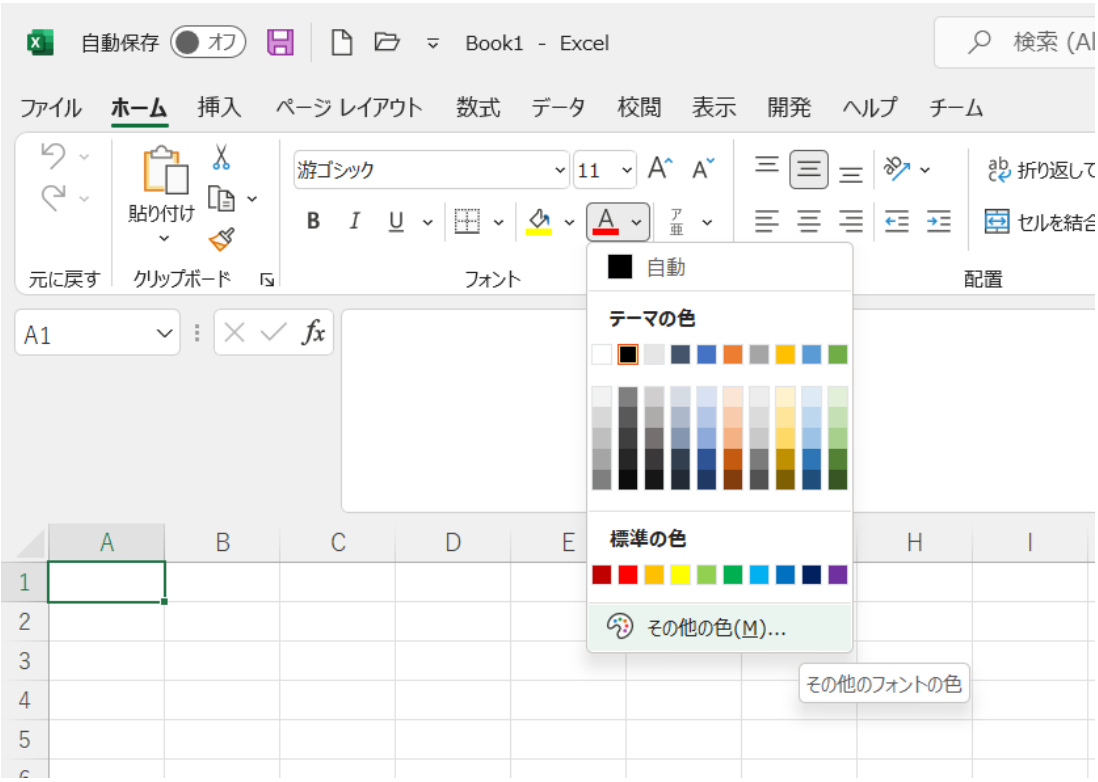
このように、基本的な構造は 10 進法と何ら変わりがありません。ちなみに、下についた括弧は、16 進法であることを強調した表記です。（そうしないと 10 進法の数と間違えることがある）

演習問題 9
16 進数で表された数”CC”は、10 進数で表すとどう表されるか。
演習問題 10
10 進数で表された数”128”を 16 進数で表すとどう表されるか。

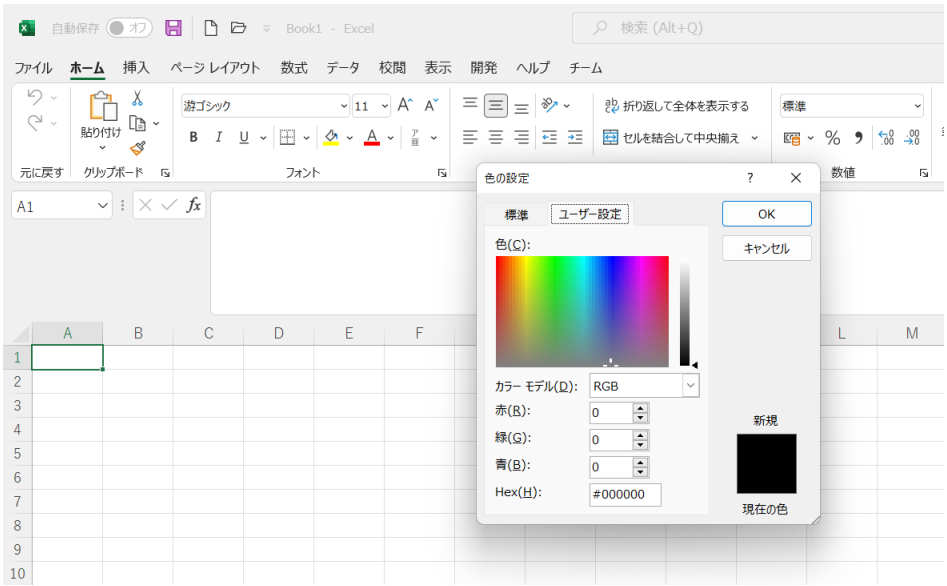
準備ができたところでカラーコードの話に移ります。パソコンで表示される色は、すべて赤、緑、青の 3 色の重ね合わせでできています。

今、ちょっとヤバいことしれっと言ったでしょ。

本当です。パソコンで表示される色はすべて赤と緑と青の重ね合わせです。ただ、赤を例を出すといわゆる「赤の程度」が 0~255 の 256 段階で表されるので、赤かそうでないかの 2 択ではありません。赤と青を混ぜると紫になるとはよく言われますが、パソコンの色はそんな単純な混ぜ合わせだけで構成されるものではありません。各色 256 段階の中から任意に選ぶことで、全ての色を作成することができます。

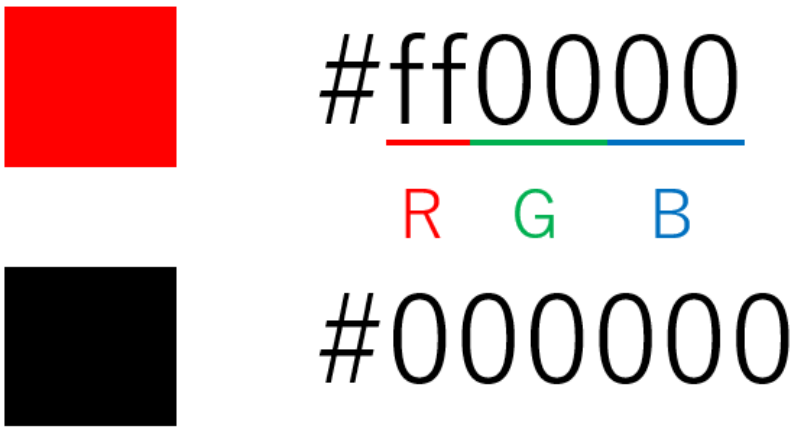


上の図はエクセルの画面を表していますが、文字の色を決める部分をクリックすると、エクセル側がデフォルトで用意している色が表示されますが、その他の色を作ることもできます。「その他の色」をクリックしてみましょう。



最初は「標準」というところが表示されますが、「ユーザー設定」というところを押してみましょう。すると、カラーモデルが RGB となっていることが分かると思います。R が Red、G が Green、B が Blue です。このカラーモデルこそが、先ほど説明した話です。他にもカラーモデルはありますが、基本的に RGB が用いられます。カラーモデルの下に、赤、緑、青と書いていると思います。ここにそれぞれ 0~255 のうちのいずれかの整数を入力することで、色を作ることができます。画面ではすべて赤、緑、青がすべて 0 ですが、このときの色は黒です。意外かもしれませんが、すべての要素を 0 にすれば黒になります。逆にすべての要素を 255 にすれば、白となります。

そして、肝心のカラーコードの表し方ですが、#の後に赤、緑、青の成分を 16 進数で表したものを並べます。例を見てみましょう。



例の 1 つ目では、赤の成分を 255（16 進法で FF)、緑と青の成分を 0 としています。このとき、カラーコードが”#ff0000”で表されます。おわかりの通り、このカラーコードで表される色は純粋な赤です。また、すべての成分を 0 にしてしまえば黒になると説明しましたが、その場合、カラーコードは”#000000”となります。

「色つけたいな～」と思ったとき、執筆者はいつも以下の Web サイトにアクセスし、適切な色を探しています。

http://www.netyasun.com/home/color.html

皆さんもここからカラーコードを選んで、Web サイトに彩りを持たせてみましょう。

演習問題 11

R 成分、G 成分、B 成分の大きさがすべて同じ色の集合は、どんな色の集合であるか、先ほどリンクを載せた Web サイトを見ながら考えてみましょう。また、この集合に含まれる色のみを使用することで生じるメリットについても考えてみましょう。

3-3 CSS ファイルの作成

1 章で CSS ファイル（拡張子が.css のファイル）がちらっと出てきました。しかし、ここまでの説明では、CSS は HTML に書いてしまうもの、ということになってしまっています。じゃあ CSS ファイルは何なんだ、ということになりますが、よく考えてみてください。1 節で紹介した書き方では、複数の要素に同じスタイルを適用しようとすると非常に大変です。また、同じ要素に多数のプロパティを指定しようとすると HTML ファイルに書かれてある内容が煩雑になります。そこで、スタイルに関する内容は別のファイルに書いて HTML 本体とは分離してしまおう、というのがこの節の目的です。この考え方は何も HTML と CSS のみに当てはまるものでは無く、全体としては大きなプログラムを複数に分割して見通しをよくするなんてことは実際によく行われます。

ということで、サンプルプログラムを見てみましょう。

html3_2.html

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <title></title>
        <link rel = "stylesheet" href="css/css3_2.css">
    </head>
    <body>
        <header>

        </header>
        <nav>

        </nav>
        <div id="contents">
```

```
            <p>Hello World!</p>
        </div>
        <footer>

        </footer>
    </body>
</html>
```

css3_2.css

```
p {
    color: red;
}
```

CSS ファイルに関しては、作業中のディレクトリ内に新たに css ディレクトリ（フォルダ）を作成して、その中に入れましょう。

まずは HTML ファイルからみていきましょう。head 要素の子要素に、

<link rel = "stylesheet" href="css/css3_2.css">

があることがわかります。これは、HTML ファイルによってできるページがどの CSS ファイルの指定するスタイルに沿うか、ということを示しています。CSS ファイルを導入する際は、常にこう書けば良いのですが、href 属性だけは気をつけましょう。href 属性として、**使用したい css ファイルに対する相対パス**を書く必要があります。ここでは、html3_2.html の属するディレクトリ（カレントディレクトリ）の 1 つ下のディレクトリである css ディレクトリ中に css3_2.css が存在するので、相対パスは

css/css3_2.css

となります。パスの指定方法忘れたなら 1 章 3 節を復習！

次に、css3_2.css を見てみましょう。やっていることは p 要素の文字を赤色にする、ということです。この書き方を知ってください。css ファイルの書き方は

```
セレクタ {

    プロパティ名: 値;

}
```

です。プロパティ名: 値;の部分は 1 節と同じですが、セレクタという要素が新たに加わります。このセレクタというものは、中に書かれている「プロパティ名: 値;」をどの要素に設定するかを示すものです。例えば p 要素に設定したければセレクタのところには p と書け

ば良いわけです。

しかし、タグの名称だけで全てのことをまかなおうとしても限界があります。例えばcss3_2.css を適用した場合、全ての p 要素の文字色が赤になってしまいます。これはさすがによろしくないと思います。じゃあ特定の要素だけに装飾を施したければどうしたらいいんだ、と思われるかもしれません。そこで登場するのが class と id という概念です。サンプルプログラムを見てみましょう。

html3_3.html

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <link rel = "stylesheet" href="css/css3_3.css"/>
        <title>html3_3</title>
    </head>
    <body>
        <header>

        </header>
        <nav>

        </nav>
        <div id="contents">
            <h1>たまこみいちょう祭企画</h1>
            <p class="blue">英会話サークルたまこみは、なんといちょう祭に出展します！
</p>
            <p  id="green">なんと、様々な学部の先輩が皆さんとお話ししてくれます！
</p>
            <p>話をしたい先輩のいる時間を狙って行ってみよう！</p>
        </div>
        <footer>

        </footer>
    </body>
</html>
```

css3_3.css

```
p {
	color: red;
}

.blue {
	color: #0000ff;
}

#green {
	color: #00ff00;
}
```

html3_3.html 中のポイントは、

```
<p class="blue">
```

と

```
<p id="green">
```

です。特定の p 要素に対してクラス、または id 属性をつけてしまうのがポイントです。そして、css3_3.css では、特定のクラス属性、または id 属性をもつ要素に装飾を施します。クラスを指定するとき、セレクタの書き方は".blue"というように、

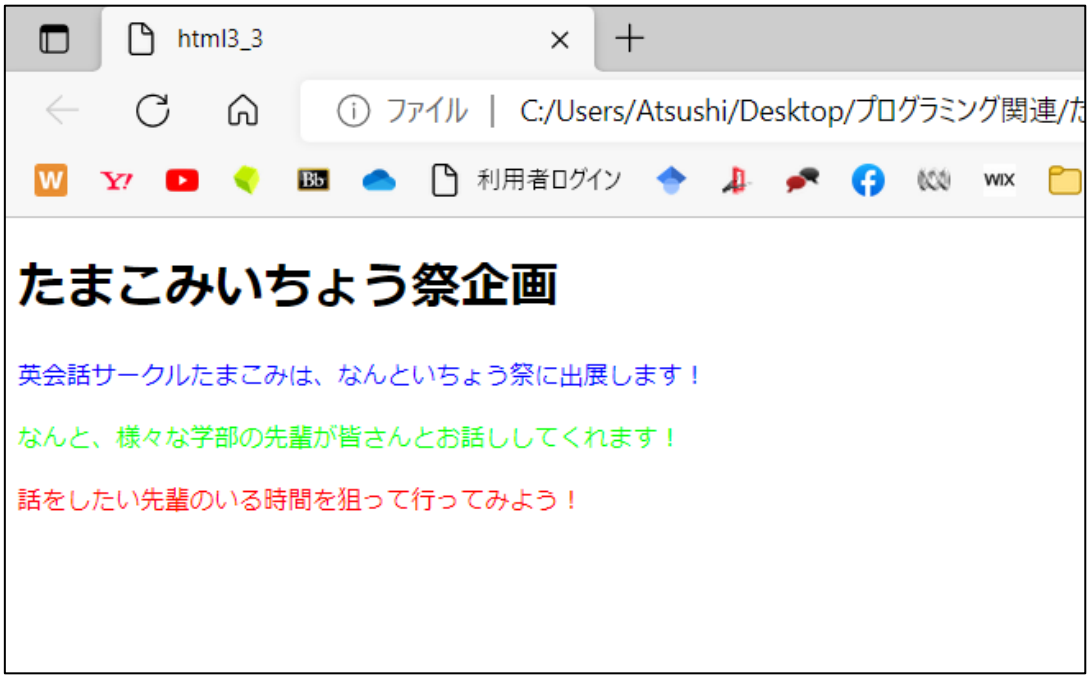
.(クラス名)

です。また、id を指定するとき、セレクタの書き方は"#green"というように、

#(id 名)

です。こうすることで特定の要素に対してのみ装飾を施す、ということが可能になります。

この結果得られる表示は以下のようになります。



いかがでしょうか。同じ p 要素でも、色が違うと思います。

ただ、ここで疑問がいくつか生じます。

Q1:css3_3.css では全ての p 要素に対して文字を赤色にすると指定している。これは.blue に対する指定や#green に対する指定と競合するのでは？
Q2:クラスと id、css ファイル上でセレクタの書き方が異なる以外は全く同じであるような気がするのだが、違いは何か。

Q1 に対しては特に問題がありません。もしこのように競合する指定を施した場合、「より細かい指定」の方が優先される、という決まりになっています。今回、文字を赤色にする、というのは全ての p 要素に対する指定でありますが、文字を青色にする、というのは blue クラスの p 要素のみに適用されます。そのため、文字を青色にする、という指定の方がいわゆる「範囲が細かい」指定であるので、そちらが優先され、文字が青くなる、というわけです。このサンプルだとまだ分かりやすいからいいのですが、複雑なスタイルの指定をしようとすると競合する場合どのようなアルゴリズムで優先されるスタイルが決まるのか、というのを考えるのは難しく、執筆者でもよく分かっていません。しかし、それは複雑なスタイルを適用したい時が来たときに調べたり試行錯誤したりしたらよいだけで、とりあえずざっくりと「より範囲が細かい指定」が優先される、と思ってもらって結構です。

Q2 に関して言えば、クラスと id には大きな違いがあり、クラスは同じ HTML ファイル内で複数の要素に同じ属性を与えることができるのに対し、id は同じ HTML ファイル内では 1 つの id は 1 つの要素にしか与えることができません。

学校でも特定の複数人が同じクラスであることはあっても、特定の

複数人に同じ学籍番号(=id)が与えられていたら嫌でしょう！

id とはそういうことです。重複を許さず、id とそれが与えられた要素は 1 対 1 で対応する、ということです。

サンプルプログラムであるため、書き方を覚えてもらうために id=”green”としましたけど、HTML ファイル内のコンテンツが充実してくるに伴い、緑文字にしたい要素は複数出てくる可能性が高いので、この書き方はあまりよろしくありません。本来ならばこちらも id ではなくクラスで指定すべきです。このような理由で、CSS を適用したいときは、クラス属性が最も広く使用されます。

CSS ファイルはもっと複雑な書き方も許容しています。以下のような例を見てみましょう。

例 1：class=”blue1”と class=”blue2”の両方の文字を青色にしたい場合

```
.blue1, .blue2{
        color: blue;
}
```

例 2：id=”table1”の table 要素の子要素 td の文字を紫色にしたい場合

```
#table1 td{
        color: #800080;
}
```

例 1 のように、並列にする場合はコンマで区切り、例 2 のように、「～の親要素の中にある特定の子要素」を指定したい場合は、コンマを付けずに並べます。

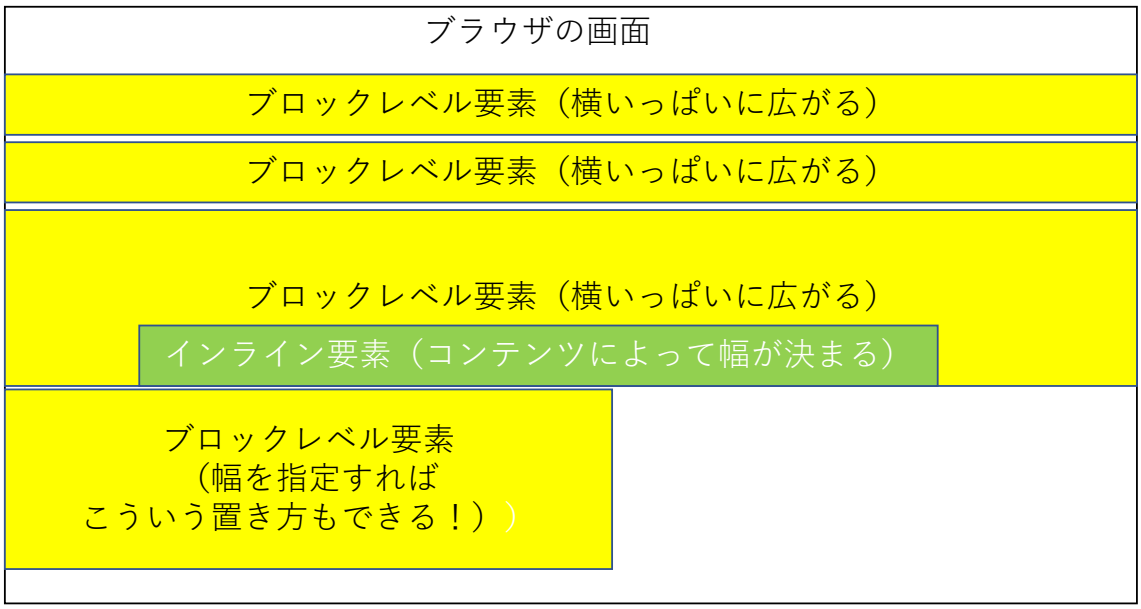
クラスや id の概念によって、細かくスタイルが指定できるようになりましたが、今後もっと細かくスタイルを指定しようと思うと、これでは不十分です。そこで、「強力な武器」が登場するのですが、その「強力な武器」を使用するためには、その前段階として少し込み入った話をしなければなりません。次節はその内容について話します。

3-4 ブロックレベル要素・インライン要素・div タグ・span タグ

新しい用語が目白押しの章です。内容としては本章では一番難しいところなので、気合いを入れて乗り越えましょう。

ブロックレベル要素とは、新しい行から始まって、（特に指定が無い限り）利用可能であるだけ横幅をとる要素のことです。一方、インライン要素とは、行の途中からでも始められて、横幅がコンテンツの中身に応じて決まる要素です。インライン要素は、ブロック要素の

子要素となることが多いです。
　ざっくり図にしてみるとこんな感じでしょうかね。



※最新の HTML5 では、各タグについて、ブロックレベル要素・インライン要素という区別が撤廃され、カテゴリーごとにタグが分類されています。しかし、タグの分類という観点では、ブロックレベル要素・インライン要素という区別がわかりやすいので、便宜上これからもこの分類法を使っていきます。

　ブロックレベル要素・インライン要素の代表的なもの（主にコンテンツの中身として使われるもの）を表にまとめます。

ブロックレベル要素の代表的なタグ

タグ名	簡単な説明
h1, h2, h3, …	見出しを表すタグ
p	テキストを表すタグ
ul	番号無しリストを作るタグ
ol	番号付きリストを作るタグ
li	リストの項目を作るタグ
table	表を作るタグ
form	フォーム要素を作るタグ（今後登場？）
div	特に意味は無い
header, footer, nav	それぞれヘッダ、フッタ、ナビゲーションを表すタグ

インライン要素の代表的なタグ

タグ名	簡単な説明
a	アンカー要素　リンク先を示す
img	画像
canvas	キャンバス要素　執筆者がゲームや再履バス同好会の学祭コンテンツを作る際に酷使
button	ボタン　フォームを扱うときよく使用される
sub	下付き文字
sup	上付き文字
strong	強調　内容の重要性・重大性を示す
span	特に意味は無い

これら 2 つの種類の要素には、様々な違いがあります。執筆者も最初はこのあたりあんまりわからなくて、意味の無いスタイルの指定をして反映されないなど、多々苦しみました。ということで、皆さんには同じ苦しみを味わうことのないよう、重要な違いを述べていきます。

①横幅と高さの指定について

ブロックレベル要素・・・横幅と高さが指定できる

インライン要素・・・横幅と高さは指定できない

先ほども少し言いましたが、ブロックレベル要素は幅を定めることができます。そのことによって、Web ページ上において好きなように置くことができます。一方、インライン要素は、中身のコンテンツによって高さや幅が異なってきます。こちらから指定することはできません。

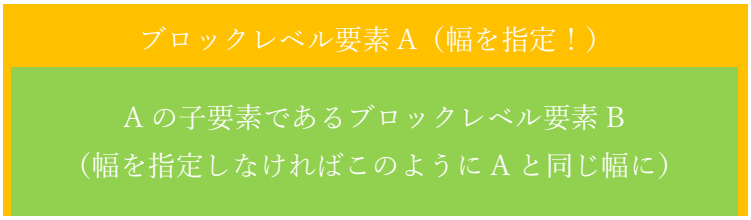
②横幅の初期値について

ブロックレベル要素・・・親要素の横幅

インライン要素・・・内容で決まる

ブロックレベル要素は、特に指定しなければ横幅の初期値は親要素の横幅になります。

body タグの直下にあるブロックレベル要素は、ブラウザの横幅と同じだけ幅をとります。（横いっぱい広がるとは、こういうことです！）一方、ブロックレベル要素の子要素としてさらにブロックレベル要素が入る、ということもありますが、この際は、特に指定しなければ、親要素と同じ幅だけ子要素も幅を取ります。



※分かりやすいように図では微妙に横幅を変えていますが、実際には同じ横幅です。

インライン要素は、何度も繰り返しているとおり、内容で幅が決まります。

③他の要素との並び方

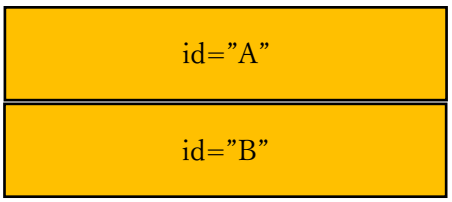
ブロックレベル要素・・・縦並び

インライン要素・・・横並び

もしHTMLファイル上にブロックレベル要素やインライン要素を連続で書いたらどうなるでしょうか。

```
<div id=”A”></div>
<div id=”B></div>
```

というように、ブロックレベル要素を 2 つコード上で並べると、以下のように並びます。



一方、

```
<span id=”A”></span>
<span id=”B></span>
```

というように、インライン要素を 2 つコード上で並べると、以下のように並びます。

以上がブロックレベル要素・インライン要素の説明となります。

ここで、先ほど「何も意味は無い」といった div タグ、span タグについて説明していきます。これらは、それぞれブロックレベル要素、インライン要素となる「かたまり」をつくるタグなのです。このタグに class 属性、id 属性をつけて、その属性に対してスタイルを指定してやることで、div タグや span タグの子要素全てに指定されたスタイルが適用されます。複数のタグやテキストをひとくくりにでき、一括でスタイルを適用できる、非常に便利なタグなのです。

演習問題 12

css ディレクトリ内に新しく”css2_2.html”というファイルを作り、”html2_2”に対して新しく作った css2_2.html を参照できるよう link タグを追加しましょう。html2_2.html はこれ以上触れず、css2_2.html を編集することで、ブラウザに表示される文字を全て赤色にしてみましょう。

演習問題解答例

演習問題 3

../../blogs/activity1.html

演習問題 7

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <header>

        </header>
        <nav>

        </nav>
        <div id="contents">
            <h1>プログラミング教室の時間と場所</h1>
            <table>
                <tr>
                    <th>開催時間</th>
                    <td>20:00</td>
                    <td>22:00</td>
                </tr>
                <tr>
                    <th>開催場所</th>
                    <td>教室 1</td>
                    <td>教室 2</td>
                </tr>
            </table>
        </div>
        <footer>
```

```
        </footer>
    </body>
</html>
```

演習問題 8

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <header>

        </header>
        <nav>

        </nav>
        <div id="contents">
            <a href="html2_5.html"><img src="images/IMG_20220115_094346.jpg"></a>
        </div>
        <footer>

        </footer>
    </body>
</html>
```

演習問題 9

12×16+12=204 なので、204 である。

演習問題 10

128÷16=8 と、ちょうど 16 で割り切れる。そのため、16 進数で表すと、80 になる。(16 進数であることを強調する表記は省略)

演習問題 12

html2_2.html

```
<!DOCTYPE html>
<html lang="ja">
	<head>
		<meta charset="UTF-8">
		<link rel = "stylesheet" href="css/css2_2.css"/>
		<title></title>
	</head>
	<body>
		<header>

		</header>
		<nav>

		</nav>
		<div id="contents">
			<h1>HTML の基礎</h1>
				<p>本章では、HTML の基礎を学びます。</p>
		</div>
		<footer>

		</footer>
	</body>
</html>
```

css2_2.css

```
#contents{
	color: #ff0000;
}
```