# Web 講義(2/8)

## リンク一覧

・MDN
「ウェブ開発を学ぶ」
・js-lecture リポジトリ(discord 参照)
・React 公式サイト(https://ja.react.dev)
・Next.js 公式サイト
・サル先生の Git 入門

## サイトを作るには

サイトに関する技術は複数あります。

### クライアント側

・HTML
サイトの「構造」を決める。
・CSS
サイトの「デザイン」を決める。
・JavaScript
サイトの「動作」を決める。

### サーバー側

・Node.js
プログラミング言語のうちサーバー側では Node.js が利用されることが多いです。
・React
「コンポーネントの再利用」「インタラクティブ」「State 管理」
HTML,CSS,JavaScript(client)のコードを生成する。
・Next.js
サーバーの Routing を管理する。

これらの技術をすべて学ばなければなりません。

特にクライアント側の技術を学ぶには MDN というサイトが有用です。このサイトを毎週読んでから、実際に自分でプログラミングしてもらいます。

(サイト以外の技術)
Git,GitHub,VsCode,WSL,クラウドなど

これらの実践的なプログラミング技術以外にも、いわゆる情報の授業でやるような技術も知識として知っておく必要があります。

Js-lecture repository の情報リテラシー1~3 を参照。
(パワーポイントしかないです)

今回は環境構築を進めたのち、これらの技術の内容を解説します。

### 今週の課題

「ウェブ入門」
「サル先生の Git 入門」(マージ以外)
「HTML 概論」※

WEB 講義 参考プリント No.1

・setTimeout/setInterval

Ex1

```
setTimeout(()=>{
    console.log("hello world")
},1000)

setInterval(()=>{
    console.log("hello world(繰り返し)")
})
```

(1000 ミリ秒= 1 秒)

setTimeout(関数,ミリ秒)でミリ秒後に関数を実行する。

setInterval(関数,ミリ秒)でミリ秒ごとに実行する。

・async/await 非同期処理

Ex2 promise

```
const promise = new Promise((resolve,
reject) => {
    setTimeout(() => {
        resolve("hello world")
    }, 1000)
})

promise.then(data => {
    console.log("log:" + data)
}).catch(e=>{
    console.log("エラーが出ました。"+e)
})
```

```
>>> "log:hello world"
>>> "hello world" (一秒後)
```

Promise とは非同期の処理をするためにある。

Resolve は解決、reject はエラーを出すための関数。

今回ではこの promise は Promise<string>型になる。

string を後で返す、という意味の関数。

Then はこの後に実行する。Catch はエラーが出たときの処理。

Ex3 1 秒待つコード

```
async function main() {
    console.log("1")
    await new Promise((resolve) => {
        setTimeout(resolve, 1000)
    })
    console.log("2")
}
main()
```

```
>> "1"
>> "2"(1秒後)
```

Promise を待機した後に実行するには、async を関数の前につけ、非同期関数にする必要がある。

Ex4 非同期関数

```
async function getStringInAMinute(){
    await new Promise((resolve) => {
        setTimeout(resolve, 1000)
    })
    return "hello world"
}
const result= await getStringInAMinute()
```

非同期関数に return をつけて返り値をつけるようにすると、Promise と同様に await を用いることができる。(今回は await が非同期関数の中で実行されていないため、エラーが出る。Ex3 にように main 非同期関数内で実行すること。)

Ex5 axios で post,get 通信をする

```
const axios=require("axios")
async function main(){
    const
response=axios.get("https://google.com")
    // const
response=axios.post("https://google.com")
    // post も実行できる。
    const data=response.data
    console.log(data)
}
main()
```

axios というパッケージを用いてコードを実行する。
post 通信: データを URL に渡して、データをもらう。
get 通信: 単に URL からデータを得る。普段サイトをみるときの html は get 通信で取得されている。

WEB 講義 参考プリント No.2
TypeScript の基礎
基本的にはサバイバル TypeScript(https://typescriptbook.jp/)を参考にすること。

・環境構築
(コマンド)

```
npm init
npm i -D typescript
npx tsc --init
```

npm i に-D オプションをつけると development(開発用)という意味になる。実行時には使用しないパッケージだから。
npx tsc とは tsc コマンドを実行するためのもの。tsc--init コマンドは tsconfig.json を生成する。tsconfig.json には、typescript のコンパイルオプションが書かれている。
tsconfig.json の中の設定のうち次のようにしておけばよい。
(tsconfig.json)

```
- "target": "es2016",
+ "target": "ESNext",
-  // "outDir": "./",
+ "outDir": "./dist",
```

(場所がわからない場合は Ctrl+F で検索してみよう。)

・コンパイル
index.js ではなく index.ts というファイルを作る。このファイルに下の内容を書く。
(index.ts)

```
function add(a:number, b:number) {
    return a + b;
}
console.log(add(1, 2))
```

(コマンド)

```
npx tsc
```

dist というフォルダが作成され、中に index.js ファイルがあるはず。
これを node コマンドで実行すると..
(コマンド)

```
node dist/index.js
```

もちろん実行される。

・コンパイルの意味
TypeScript は JavaScript に型注釈を与えるための言語であり、実行するには JavaScript ファイルに書き換える必要がある。そのために使う道具が tsc コマンドである。この型注釈について学んでいく。

・コンパイルの自動化
package.json を次のように書き換えてみよう。
(package.json)

```
"scripts": {
--- "test": "echo ¥"Error: no test specified¥" && exit 1"
  },
```

```
"scripts": {
+++ "start": "npx tsc && node dist/index.js"
  },
```

そして npm start コマンドを実行すると

```
> typescript@1.0.0 start
> npx tsc && node dist/index.js
```

このように一つのコマンドで実行までできる。
・座学
サバイバル TypeScript の「読んで学ぶ TypeScript > 値・型・変数 > 変数宣言: let と const」から読んでいこう。
・問題
(1)もちろん次のように宣言するだけで変数の型は勝手に指定してくれるが、
(index.ts)

```
const a=1 //a は number 型
```

自分で指定せよ。(例↓)

```
const a:number=1
```

```
"hello world"
[1, 2, 3]
true
12n
```

(2)関数の返り値の型はどのように指定すればよいか。

(3)高校 3 年生、高校 2 年生、高校 1 年生という string だけが入る type を作成し、変数に注釈をつけることでそれいがいの string が入らないことを確認せよ。

(4)以下の変数に対し getDate()関数を実行するとき、a が undefined の場合実行しないようにするにはどうすればよいか？オプショナルチェーンを用いて記述せよ。

```
let a:Date|undefined;
```

(5)

(i)以下の object に対し a,b という変数を取り出せ。ただし分割代入を使うこと。

(ii)分割代入して取り出した a,b に対し、value1,value2 という別名を定義するにはどうすればよいか？

```
const obj = { a: 1, b: 2 };
```

(6)以下の配列に対しスプレッド構文を用いて console 出力せよ。

```
const a=[1,21,2,23,23233,32,32]
```

ただし次のような出力になる。

```
>  1 21 2 23 23233 32 32
```

(7)D が C と同じ型になるようにせよ。ただし、インターセクション型を用いること

```
type A={
    id:string,
    name:string
}
type B={
    email:string
}

type C={
    id:string,
    name:string,
    email:string
}

type D=???
```

(8) 以下のコードについて、指定された部分を三項演算子を用いることで一行で表せ。

```
const point = 30
type Grade = "A" | "B" | "C" | null



//ここから
let grade: Grade = null

if (point >= 90) {
    grade = "A"
} else if (point >= 80) {
    grade = "B"
} else {
    grade = "C"
}
//ここまで
```

・総合課題

(1)Express というパッケージを使ってみよう。下のコードを動かしたのち、localhost:3000 というサイトをブラウザで開く。(ググりながらやってください。)

```
import express from "express"

const app = express()

app.get("/", (req, res) => {
  res.send("Hello World")
})

app.listen(3000, () => {
  console.log("Server listening on port 3000")
})
```

(i) デベロッパーツールで network タブを見てみよう。

(ii) app.get の get は何を意味しているか？

(iii) console.log(req.query)などでプロパティの意味を調べてみよう。