udemy TypeScript はむさん

セクション 1

4 typescript とは

- TypeScript PlayGround MS が出しとるやつ遊び場

7

```
 git checkout -b create-package-json
```

チェックアウトとリポジトリ作成を両方やる

8 TypeScript をインストールする

```
npm i typescript@3.7.5 --save-dev
```

Powershell でファイルを新規作成

```
  New-Item src/install-typescript.ts
```

- ts をコンパイル tsc はグローバルにインストールされていないので`node_modules`から指定する

```
./node_modules/.bin/tsc src/install-typescript.ts
```

- gitbash でコードを見る

```
cat install-typescript.ts


```

- 生成された js ファイルを削除するまでの流れ 全部加える

```
 git add .
```

status 見る

```

```

```
git status
```

git の追跡から外したくない場合は`git reset`する

```
makito.mori@PC790 MINGW64 ~/Desktop/self_study/udemy/udemy-typescript (install-ts)
$ git reset src/install-typescript.ts

makito.mori@PC790 MINGW64 ~/Desktop/self_study/udemy/udemy-typescript (install-ts)
$ git status
On branch install-ts
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   memo.md
        modified:   package.json
        new file:   src/install-typescript.js

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        src/install-typescript.ts
```

ファイル削除(参考：https://qiita.com/k0uh0t/items/ae885bf2d5e05614b80f)

```
$ git clean -f
Removing src/install-typescript.ts
```

- master にマージする

```
git checkout -
```

ハイフンは前回のブランチ

```
git merge -
```

9 ts-node の導入

コンパイルと実行を同時に行う。

- インストール

```
npm i ts-node
```

- 実行

```
npx ts-node src/install-typescript.ts
```

10 ts-node-dev の導入

ファイルが修正されるたびにコンパイルと実行を行う。

- インストール

```
npm i ts-node-dev --save-dev
```

- 実行(github:https://github.com/whitecolor/ts-node-dev)

```
npx ts-node-dev --respawn src/install-typescript.ts
```

11 vs-code インストール

settings.json を変更する

```
  "prettier.semi": true,
  "prettier.singleQuote": false,
```

- ts-config.json を作成する

```
 npx tsc --init
```

セクション 2 基本的な型

12 boolean

src 下にファイルを作成する これで作ると変な文字が入ってコンパイルされない。

```
echo "export {};" > src/boolean.ts
```

代わりにこっちを使う

```
 New-Item .\src\test.ts -Value "export{}"
```

- boolean.ts

```
export {};
let name = "TypeScript";
```

export がないと name は警告が出る。変数 name は ts 側ですでに宣言されているので使えない。 export してモジュール化してあげることで警告を回避できる。 これからの講義で export{}をしばしば書く

15 Array 型

書き方 ただし 2 つ目は非推奨

```
let numbers: number[] = [1, 2, 3];
let numbers2: Array<number> = [1, 2, 3];
```

- array.ts
    - 1 次元 配列

```
let strings: string[] = ["TS", "JS", "CS"];
```

- 2 次元配列

```
let nijigenHairetsu: number[][] = [
[50, 100],
[150, 300],
];
```

- 型が混合した配列(union type 共用型) `|`は「または」の意味なので`1`,`false`,`Japan`の順番は任意

```
let hairetsu: (string | number | boolean)[] = [1, false, "Japan"];

```


16 tuple 型
共用型と違って順番も制限を入れる
```
```

```
let profile: [string, number] = ["Ham", 43];
profile = [43, "ham"]; //エラーが出る

```

17 any型
axiosのgithub:https://github.com/axios/axios

自分の好きな型を作れる。（研修でinterfaceやったけど忘れた）
```
interface Article {
    id: number;
    title: string;
    description: string;
  }
let data: Article[]; //Article型を定義する
```


18 void型
関数の戻り値に対するアノテーション
関数の横に`:~~`でアノテーションできる
* void.ts
```
function returnNothing(): void {
  console.log("I don't return anythings");
}
```
19 null undefined 型
```
let absence: null = null;
// absence = "123"; //エラー出る
let data: undefined = undefined;
// data = 123;//エラー出る
```

20 never型

never:return されない
void :空がreturnされる

* never.ts
```
function error(message: string): never {
  throw new Error(message);
}

try {
  let result = error("test");
  console.log({ result });//errorという関数を実行した時点でエラーが投げられるのでconsole.logの処理は実行されない
} catch (error) {
  console.log({ error });
```

```
}

```

voidは`undefined`と`null`は入れられるけどnever型は無理。
```
let foo: void = undefined; //エラーでない
let foo2: void = null; //エラーでない
let bar: never = undefined; //エラー出る
let bar2: never = error("only me"); //never型のみ代入できる。使い道はない。
```

21 object型
アノテーションで`:object`と書ける。ただし`:object`は幅が広くてキーが上書き出来て、バリューも違う型のもので上書きできる。
```
let profile: object = { name: "Ham" };
profile = { birthYear: 1976 }; //上書きできる
```

もっとアノテーションを限定的にする
```
let profile2: {
  name: string;
} = { name: "makito" };
// profile2 = { birthYear: 1976 }; //上書きできない
profile2 = { name: "mori" };//上書きできる
```
セクション22 型エイリアス

###
セクション 3

###

セクション 4

###

セクション 5

###
```
```