

TypeScriptを勉強しよう

前提知識

- JavaScript を使ったプログラミング経験
- あれば良い
 - 型についての知識・オブジェクト指向

みんな嫌いな

座学

What difference between JS and TS

JavaScriptとTypeScriptにはどのような違いがあるか。

JavaScriptは動的型付け言語
TypeScriptは静的型付け言語



?

JavaScriptはあまり「型」をあまり意識せずにコードを書くことができます。

逆にTypeScriptは型を意識し、明示的に書く必要性があります。

具体にそれがどういうことなのかという話の前に、まず「型」が何なのかを勉強していきましょう。

1. What's "Type"

型って何

1. What's "Type" では型がなにかと、型を意識する重要性について考えます。

型はJavaScriptにかかわらず大抵の言語でも必要となる考え方になります。

型は、変数(定数、引数など含む)に適用できるものであり、型を意識するには抽象的に考えるのが大事です。

型には多くの種類がありますが、
手始めに、`string` 型と `number` 型を例に進めていきましょう。

例えばこのコード

```
const firstName = "Sora";
const lastName  = "Hattori";

const language  = "JavaScript";
```

例えばこのJavaScriptのコードでは

"Sora" や "Hattori" , "JavaScript" など、
具体的に何が入ってると考えるより
どの変数も、 string 型が入ってると抽象的に考えます。

`number` 型の場合も見ていきましょう。

```
const age      = 19;  
const year    = 2024;  
const height  = 185.3;
```

この場合も同じです。

`19` や `2024`, `185.3` が入ってると考えるより、`number` 型が入つ
てると考えます。

And...?

それで...?

型を意識する必要がある理由の一つがわかる例があります。
それは数値同士の計算です。

```
// 三角形の面積を求める
function triangle(bottom, height) {
  return bottom * height / 2;
}

const area = triangle(21, 5);
```

この `triangle` 関数の引数 `bottom` と `height` が求める型は、どちらも `number` 型になります。

```
// 三角形の面積を求める
function triangle(bottom, height) {
  return bottom * height / 2;
}
const input = prompt("底辺を入力してください");
const area = triangle(input, 5);
```

もし、このようなコードを書いてしまった場合。

引数 `bottom` には文字列 `1` が入ってしまい、`string * number` の計算を行おうとしてエラーが発生します。
このようなエラーを防ぐには、常に型を意識をしなければなりません。

*1 厳密には違いますが今回は便宜上 `string` として説明します。

2. In TypeScript

TypeScriptでは

2. In TypeScript では、JSではだめな理由とTypeScriptに書き換えたときのコードを比較していきます。

型を意識するだけでエラーを防げるなら意識すればいいじゃん。と思った方はいますか？

その場合考え方改めてください。

コードが1000行、ソースファイルも10個以上のような状況になったときあなたとあなたのチームメイトは、**完璧に型を意識してコードを書くことは出来ますでしょうか？**



そう、出来ません。

けれどそれを実現しなければ大量のバグの元になります。

例えばボタンを押したときに先程の `triangle` 関数のような呼び方を気づかず行ってしまうと、ボタンを押すまでエラーがわからないの事に加え、必ずしもそのボタンを押してデバックするとは限らないため、更に発見が遅れる可能性が高いです。

そのような事故を防ぐため、我々が型を明示的に書き実行する前にTypeScriptに型があるか自動的に確認してもらおう！

というのがTypeScriptです。



型と問題点がわかったところで

座学終わり

型とJSの問題点がわかったところで今までのコードを
TypeScriptに書き換えていきましょう。

次のURLからTypeScriptを試せるページに飛びます。

[TypeScript Playground](#)

先程の問題のあるJavaScriptのコードをTypeScriptで書き直して見ましょう。

以下の差分を参考にコードを修正してください。

[Github - Compare](#)

こっから下は未完成

とこのように具体的より、抽象的に考えます。

今回は `string` 型と `number` 型で説明しましたが、他にも型が無数にあります。
が、今回は

Why TypeScript

実はTypeScriptはJavaScriptのように直接実行出来ません。
ブラウザで実行するには**TypeScriptをJavaScriptに変換して**
実行します。

「ええ～～じゃあわざわざTypeScript使う必要はないじゃん！」と、そう思いますよね。

しかし今の時代、JavaScriptで開発をすることはほとんどありません。

じゃあ何でそんな面倒なことしてTypeScriptを使うのか。

2. Learn from example.

実際に、JavaScriptとTypeScriptのコードを見比べて見ましょう。

ここで三角形の面積を求める関数について見てましょう。

```
function circle(bottom, height) {  
    return bottom * height / 2;  
}  
  
const circleArea1 = circle(10, 20);  
const circleArea2 = circle(2, 100);  
const circleArea3 = circle(50, "10");  
const circleArea4 = circle(50);
```

はい。おわかりの通り、このコードは実行するとエラーが起きます。

はい実行するとです。実行するまでエラーが出てきません。
ではTypeScriptはどうか。

```
function circle(bottom: number, height: number): number {
  return bottom * height / 2;
}
const circleArea1 = circle(10, 20);
const circleArea2 = circle(2, 100);
const circleArea3 = circle(50, "10");
const circleArea4 = circle(50);
```

↓ このURLで実際の挙動を確認

このコードがエラーを吐く理由は「型」について意識していないと理解できないです。

ですので一度基礎知識の型について最初から学習していきましょう。

3. What's good about it.

じゃあそれの何がいいのかっていう話。

先ほども書いたが