TypeScript

coders
for causes

WA DATA SCIENCE INNOVATION HUB

Powering an AI Future

coders
for causes

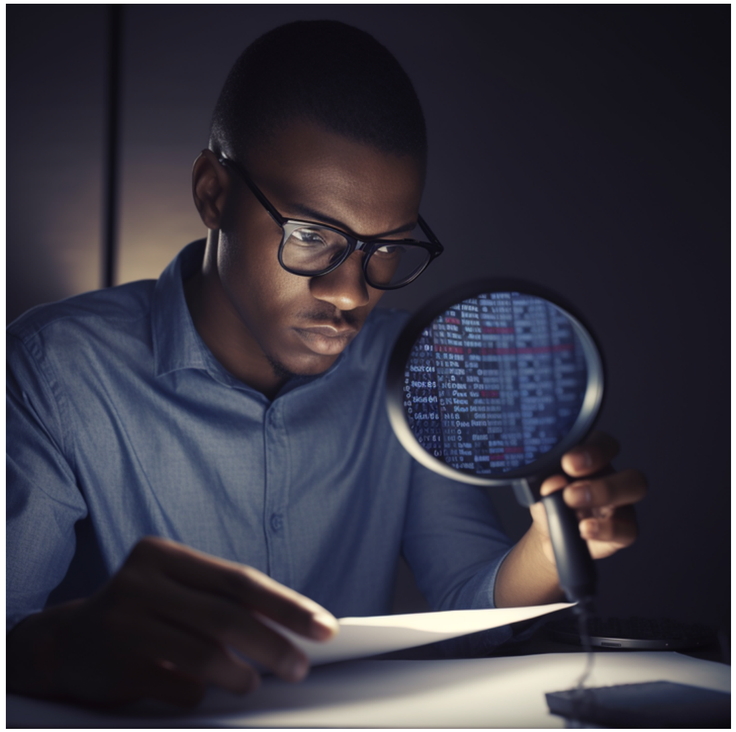
Why Should I Care?

coders
for causes

coders
for causes

2012

Developed and maintained by Microsoft and first released in 2012.



It is a superset of JavaScript that adds static types to it

coders
for causes

```typescript
let minutes: number

minutes = 'hello world'
// causes type error and code will not to
// compile for prod. In TS enabled IDEs,
// it will show you the error as below
```

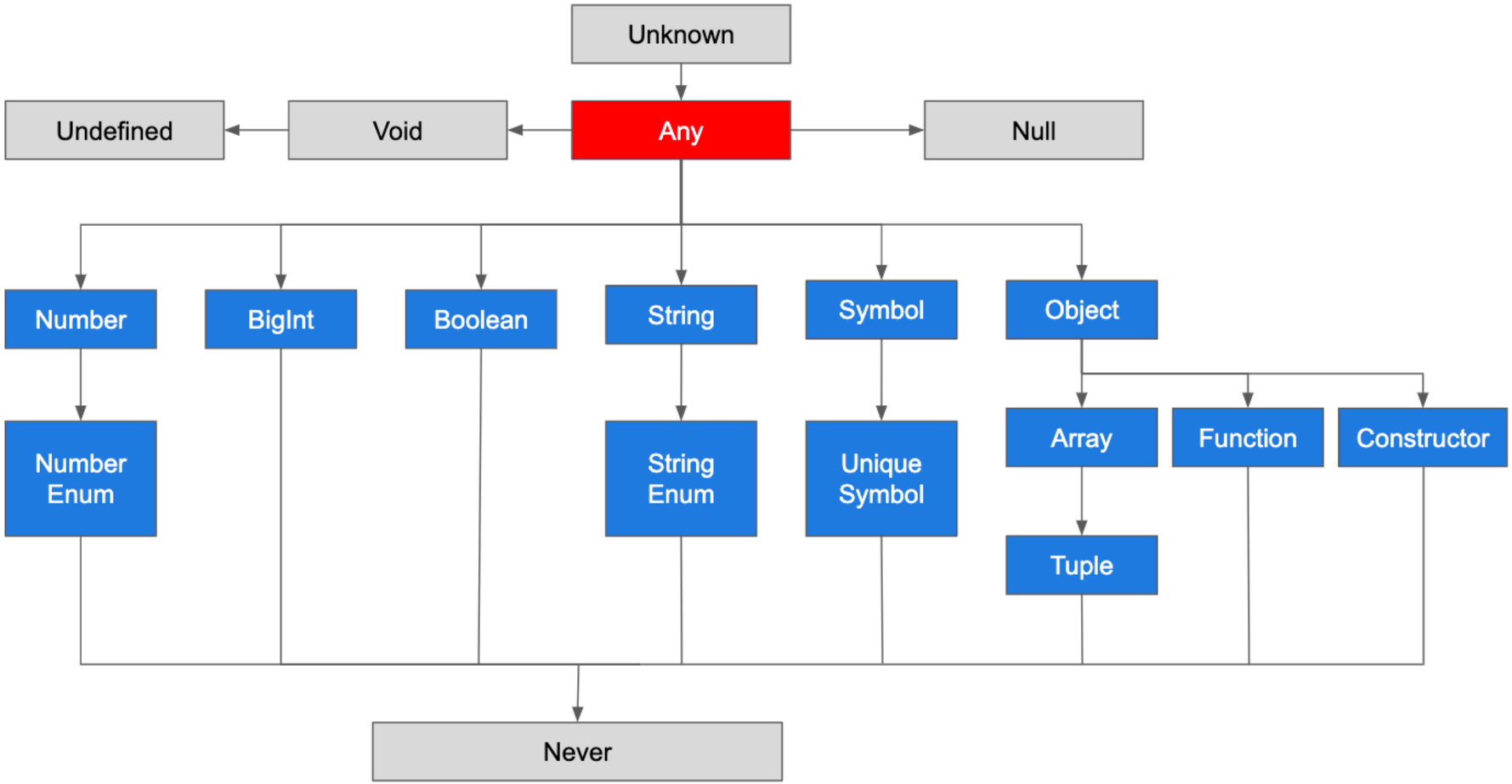
```
let minutes: number
Type 'string' is not assignable to type 'number'. ts(2322)
View Problem (Alt+F8)    No quick fixes available
```

coders
for causes

Features

- Comes with Intellisense, vastly improving developer experience.
- Worth mentioning that, TypeScript does not work natively in the browser and must be compiled back to JavaScript.

Types



coders for causes

```typescript
const pi = 3.14
// don't have to define type as it will be inferred

const HoursInDay = 24

let minutes: number
minutes = 40

console.log({ pi, HoursInDay, minutes })
// pi: 3.14, HoursInDay: 24, minutes: 40

let time: string

time = 'minute'
console.log(time) // minute

time = 'second'
console.log(time) // second
```

coders
for causes

Arrays and Union Types

Union Types

```
let value: string | number | null = null

console.log(value) // null

value = 12
console.log(value) // 12

value = 'cfc'
console.log(value) // cfc
```

Array

```
const arr = ['hello', 'world']
// inferred as array of strings

// two ways of array declaration
let arr2: string[], arr3: Array<number>


let arr4: Array<string | number>

arr4 = ['hello', 234, 45.7654, 'intro to TS']
```

coders
for causes

Intersection

```typescript
type Human = {
  name: string;
};

type Student = {
  age: number;
};

const Volunteer: Human & Student = {
  name: "Nicholas Choong",
  age: 27,
};
```

There is also a intersection type that combines types using &.

coders
for causes

Objects

Interfaces

```typescript
interface Student {
  name: string;
  age?: number; // shorthand for age: number | undefined
  studentNumber: number;
  gender: "male" | "female" | "other";
}

const student1: Student = {
  name: "Danish bin Azman",
  studentNumber: 1234567,
  gender: "male",
};

// all fields except age must be defined as they are not optional

student1.age = 18;

console.log(student1);
// { name: "Danish bin Azman", studentNumber: 1234567, gender: 'male', age: 18}
```

coders for causes

Objects (cont.)

Suppose you were building the API for the UWA Student Guild

```typescript
interface UWAStudent {
  name: string;
  studentNumber: number;
  DOB: Date;
  gender: "male" | "female" | "other";
}

interface StudentClub {
  name: string;
  isGuildAffiliated?: boolean;
}

interface GuildMember extends UWAStudent {
  isGuildMember?: boolean;
  clubs?: StudentClub[];
}
```

```typescript
const student: GuildMember = {
  name: "Jun Yap",
  studentNumber: 22507198,
  DOB: new Date(2000, 6, 7), // July 7th, 2000
  gender: "male",
  isGuildMember: true,
  clubs: [
    {
      name: "Coders for Causes",
      isGuildAffiliated: true,
    },
  ],
};
```

coders
for causes

Functions

```typescript
// returns type number
function add (num1: number, num2: number) {
    return num1 + num2
}

// returns type number by force or will error if another type
function subtract (num1: number, num2: number): number {
    return num1 - num2
}
```

```typescript
// returns type number
const add = (num1: number, num2: number) => (
    num1 + num2
)

// returns type number by force or will error if another type
const subtract = (num1: number, num2: number): number => {
    return num1 - num2
}
```

coders
for causes

Utility Types

```typescript
interface Props {
  name?: string;
  email?: string;
}

const obj: Props = { name: "nick" };

const obj2: Required<Props> = { name: "nick" };
```

```
Property 'email' is missing in type '{ name: string; }' but required in type 'Required<Props>'. ts(2741)
test.tsx(3, 3): 'email' is declared here.
Error (TS2741)
Property 'email' is missing in type { name: string } but required in type Required<Props> .
const obj2: Required<Props>
View Problem (Alt+F8)    Quick Fix... (Ctrl+.)
```

coders
for causes

Utility Types

```typescript
interface Props {
  name?: string;
  email?: string;
}

type RequiredAProps = Required<Pick<Props, "name">> & Partial<Pick<Props, "email">>;

const obj3: RequiredAProps = { name: "nick" };
```

coders
for causes

Utility Types

keyof

Record

NonNullable

Partial

Required

Pick

Omit

ReturnType

Parameters

coders
for causes

More on https://www.typescriptlang.org/docs/handbook/utility-types.html

Hooks with TS

```
1   const [state, setState] = useState('hello') // state infers type string
2   const [state, setState] = useState<string>('hello') // but you can also tell it!
3   const [user, setUser] = useState<User>(null) // can even use interfaces
```

coders
for causes

Pretty TS Errors

This:

⚠ Error (TS2322) |

Type:

```
() => {
  person: { fullName: string; email: string };
}
```

is not assignable to type `GetUserFunction`

Property `user` is missing in type `{ person: { fullName: string; email: string } }` but required in type:

```
{
  user: {
    name: string;
    email: `${string}@${string}.${string}`;
    age: number;
  };
}
```

Instead of that:

```
Type '() => { person: { fullName: string; email:
string; }; }' is not assignable to type
'GetUserFunction'.
  Property 'user' is missing in type '{ person: {
fullName: string; email: string; }; }' but required
in type '{ user: { name: string; email:
`${string}@${string}.${string}`; age: number; };
}'. ts(2322)

GetUserFunction.ts(2, 3): 'user' is declared here.
```

VS Code Extension

coders
for causes