any と never と unknown

Press Space for next page →

any型とは

どのような値でも代入できる

```
let value: any;
value = 1; //OK
value = 'string'; //OK
value = { name: "object" }; //OK
```

any型とは

コンパイラーが型チェックを行わないため、実行してからエラーになる。

```
const str: any = 123;
str.toLowerCase(); //TypeError: str.toLowerCase is not a function
```

どんな時に使う？

- どんな時でも使える
- が、any型の濫用はJavaScriptを書いていることと同じなのでできるだけ使わない

unknown型とは

どのような値でも代入できるのはany型と同じ

```
let value: unknown;
value = 1; //OK
value = 'string'; //OK
value = { name: "object"}; //OK
```

unknown型とは

unknwon型は実行ができない

```
const unknown1 = 0.8;
console.log(unknown1.toFixed()); //Object is of type 'unknown'.
const unknown2 = "riibekuntoofuro";
console.log(unknown2.length) //Object is of type 'unknown'.
const unknown3 = {
  x: 0,
  y: 1,
  name: "origin",
};
console.log(unknown3.name) //Object is of type 'unknown'.
```

どんな時に使う？

動的に型が変わる場合に使う

```
declare const maybe: unknown;
// 'maybe' could be a string, object, boolean, undefined, or other types
const aNumber: number = maybe;
//Type 'unknown' is not assignable to type 'number'.

if (maybe === true) {
  // TypeScript knows that maybe is a boolean now
  const aBoolean: boolean = maybe;
  // So, it cannot be a string
  const aString: string = maybe;
Type 'boolean' is not assignable to type 'string'.
}

if (typeof maybe === "string") {
  // TypeScript knows that maybe is a string
  const aString: string = maybe;
  // So, it cannot be a boolean
  const aBoolean: boolean = maybe;
Type 'string' is not assignable to type 'boolean'.
}
```

never型とは

値を持たない型。
never型はあらゆる型に代入可能であるが、never型には代入可能な型はない。（never型自身を除く）

どんな時に使う？

エラー処理の戻り値の型など、

- 実行される可能性のあるreturn文が存在しないと判断できるとき
- この関数は最後まで到達することはないと判断できるとき

```
// 最後まで到達しない関数はnever型の返り値となる
// 必ず通過するthrow文により最後まで到達しない（到達する実行パスが存在しない）
function error(message: string): never {
    throw new Error(message);
}
// 推論される返り値はnever型
// 上で作ったerror関数を返り値に指定してるので、必ず通過するthrow文により最後まで到達しない(到達する実行パスが存在しない）
function fail(): never {
    return error("Something failed");
}
// 最後まで到達しない関数はnever型の返り値となる
// → 無限ループに必ず入る場合も throw 文を通る実行パスが無い場合と同様
function infiniteLoop(): never {
    while (true) {
    }
}
```

どんな時に使う？

switch文の中でneverの出現を頼りに網羅的なチェックをすることができる

```
type Shape = Circle | Square;

function getArea(shape: Shape) {
  switch (shape.kind) {
    case "circle":
      return Math.PI * shape.radius ** 2;
    case "square":
      return shape.sideLength ** 2;
    default:
      const _exhaustiveCheck: never = shape;
      return _exhaustiveCheck;
  }
}
```

どんな時に使う？

switch文の中でneverの出現を頼りに網羅的なチェックをすることができる

```
interface Triangle {
  kind: "triangle";
  sideLength: number;
}

type Shape = Circle | Square | Triangle;

function getArea(shape: Shape) {
  switch (shape.kind) {
    case "circle":
      return Math.PI * shape.radius ** 2;
    case "square":
      return shape.sideLength ** 2;
    default:
      const _exhaustiveCheck: never = shape;
//Type 'Triangle' is not assignable to type 'never'.
      return _exhaustiveCheck;
  }
}
```

Void型と何が違うの？

- Void型は関数が正常に終了した結果何も返さない時に使う型
- Never型はそもそも関数が正常に終了しない時につかう型

おわり