型別範圍的限縮和擴展

型別的標註與推論

Type Inference 型別推論

隱性型別 Implicitly Type

- 編譯器會根據程式碼上下文，自動推斷出型別

```
function increment(num = 0) {
  return num + 1
}
// 表達式類型依照回傳的結果

let name = 'Kent'
// 變數、參數依照賦予的值
```

any[]、any

- 發生在變數初始值為 null、undefined、空陣列

```
let data = null
let isActive
// 變數、參數依照賦予的值
```

noImplicitAny

- 當參數沒有標註型別，且無法從上下文推斷時，預設將型別指派為 any
- noImplicitAny 參數預設開啟 (strict: true)，檢查函式參數的 implicit any 錯誤

```
function increment(num) {
  Parameter 'num' implicitly has an 'any' type.
  return num + 1
}
```

Type Annotation 型別標註

無法根據上下文推斷型別時，就需要透過手動標註明確的型別

顯性型別 Explicit Type

```
function increment(num: number) {
  return num + 1
}
```

Literal 字面值

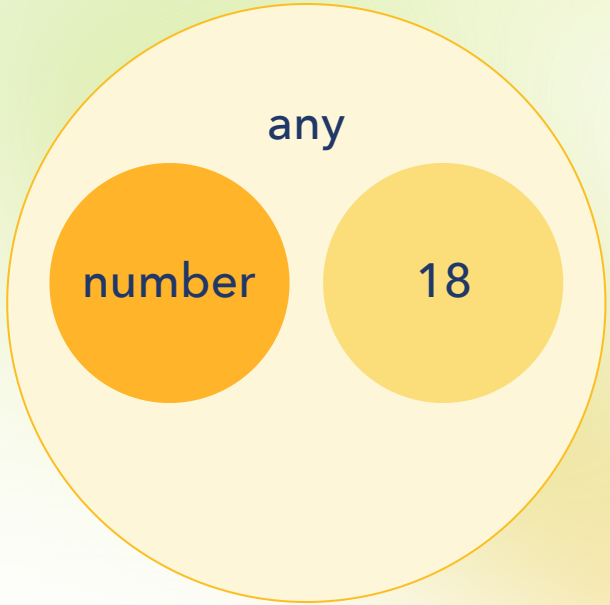
- JavaScript 的值
- 屬於固定值
- 值本身不可變

重新指派 re-assignment、值可變性 Mutability

- let 宣告的變數允許重新指派
- 基本型別 primitive 值本身 immutable 不可變
- const 宣告的變數無法重新指派
- 物件型別 object 值內部預設 mutable 可變的

Type Widening 型別擴展

```ts
let wideNum = 5          // 推斷型別為 number（廣泛的原始型別）
const narrowNum = 18     // 推斷型別為 18（具體的字面值型別）
wideNum = narrowNum      // 賦值成功：18 是 number 的子集合
```

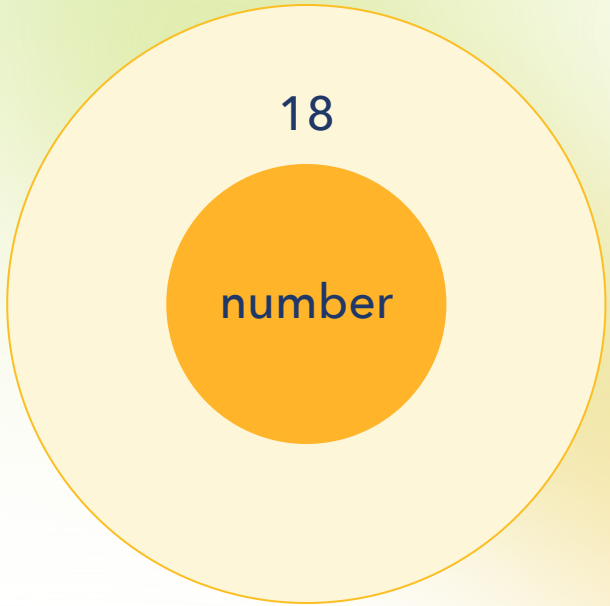


型別無法重新賦予 - 型別被推斷後就無法改變

```
let wideNum = 5
const narrowNum = 18
wideNum = narrowNum
```

```
let narrowNum: 18 = 18
let wideNum = 5
narrowNum = wideNum
```

> Type 'number' is not assignable to type '18'.



縮小型別範圍

Type Narrowing 型別守衛

```
function getLength(something: string | number): number {
  if (something.length) {
```

```
Property 'length' does not exist on type 'string | number'.
  Property 'length' does not exist on type 'number'.
```

```
    return something.length
```

```
Property 'length' does not exist on type 'string | number'.
  Property 'length' does not exist on type 'number'.
```

```
  } else {
    return something.toString().length
  }
}
```

typeof 限縮型別

```ts
function getLength(something: string | number): number {
  if (typeof something === 'string') {
    return something.length
  } else {
    return something.toString().length
  }
}
```

as 斷言

```ts
function getLength(something: string | number): number {
  if (typeof something !== 'string') {
    return something.toString().length
  }

  return (something as string).length
}
```

謝謝