用 any 斬殺所有錯誤的勇者轉生
靠 let 和 const 覺醒型別之力展開
TypeScript 之旅

Agenda

- 自動轉型：TypeScript 與 JavaScript 差異
- 型別建立：Type Inference 型別推論、Type Annotation 型別註記
- 型別範圍：值不可變、型別不可變

TypeScript 解決了什麼問題？

自動轉型

JS

```
function multiply(a, b) {
  return a * b
}
console.log(multiply('5', 2))
console.log(multiply('five', 2))
```

結果為 NaN, 因為 'five' 無法轉換為數字

TS

```
function multiply(a: number, b: number) {
  return a * b
}

console.log(multiply('5', 2))
```

Argument of type 'string' is not assignable to p

```
console.log(multiply('five', 2))
```

Argument of type 'string' is not assignable to p

型別檢查時機

JS

runtime 執行階段

```
function
getLength(str) {
  return str.
  length
}

console.log
(getLength(42))
```

```
undefined
```

TS

compile 編譯階段

```
function getLength(str: string) {
  return str.length
}

console.log(getLength(42))
```

```
Argument of type 'number' is not assignable to parameter of type 'string'.
```

IntelliSense 提示

Hint

```
function getLength(str: string) {
  return str.length;
}

getLength(str: string): number

getLength(42)     應有 1 個引數，但得到 0 個。
```

Auto Completion

Types of completions

型別建立

Function Parameters

JS

函式參數預設為 optional（可選）

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

console.log(increment(1))
console.log(increment())
```

JavaScript 行為
缺少參數時自動帶入 undefined

TS

函式參數預設為 required（必填）

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

console.log(increment(1))
console.log(increment())
```

```
Expected 1 arguments, but got 0.
```

TypeScript 檢查
缺少參數時會報錯，避免非預期結果

Default Parameters

參數是 undefined 時才套用預設值

JS

undefined 套用預設值，避免執行時錯誤

```
function increment(number = 1) {
  return number + 1
}

console.log(increment(1))
console.log(increment())
```

```
2
2
```

TS

預設值會影響型別推斷，避免型別檢查失敗

```
function increment(num = 1) {
  return num + 1
}

console.log(increment(1))
console.log(increment())
```

```
2
2
```

Implicit Any

Implicitly Type

- 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

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

noImplicitAny

- any 屬於最寬鬆的型別允許變數是任何型別，導致失去了型別檢查機制
- noImplicitAny 參數預設開啟 (strict: true)，檢查 implicit any 並標記錯誤

any[]、any

- 發生在初始值為 null、undefined 的變數推論情境

Type Inference 型別推論

編輯器會根據程式碼上下文自動推斷出型別

賦值變數型別

變數依照賦予的初始值進行推論

```
const name = 'Kent'
const age = 30
const isActive = true
const data = null
```

回傳值函式型別

函式依照 return 值進行推論

```
function getName() {
  return 'Kent'
}

function calculate(a: number, b: number) {
  return a + b
}
```

Type Annotation 型別註記

無法根據上下文推斷型別時，需要手動標註明確的型別

加上 `：型別` 來標註型別

宣告變數、參數時在後方

function

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

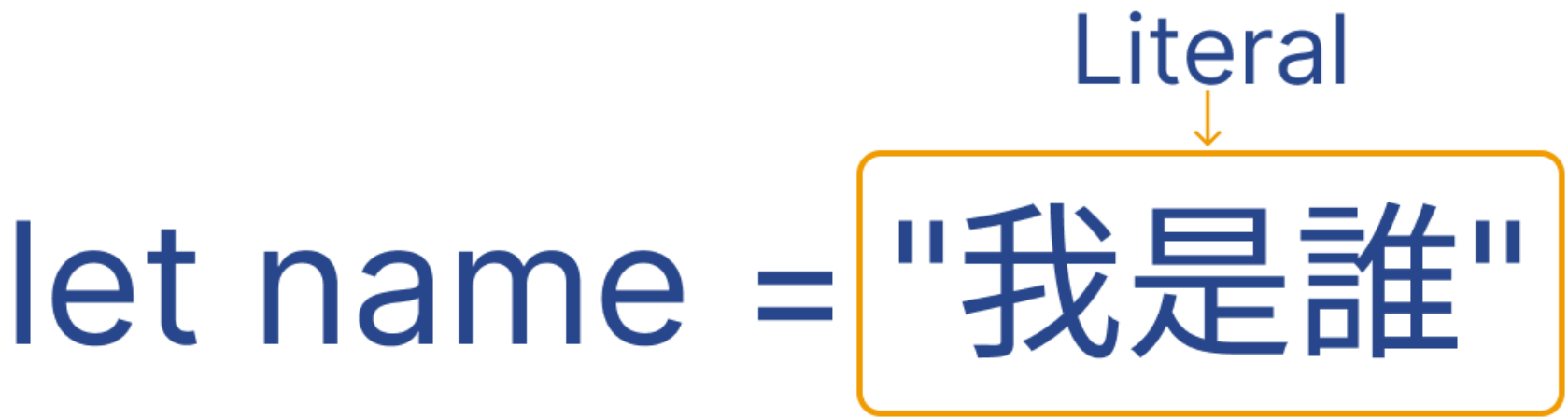
variable

```
let name = 'Kent'
let data = null

type User = {
  name: string
  age: number
}

let user: User = {
  name: 'Kent',
  age: 30
}
```

型別範圍

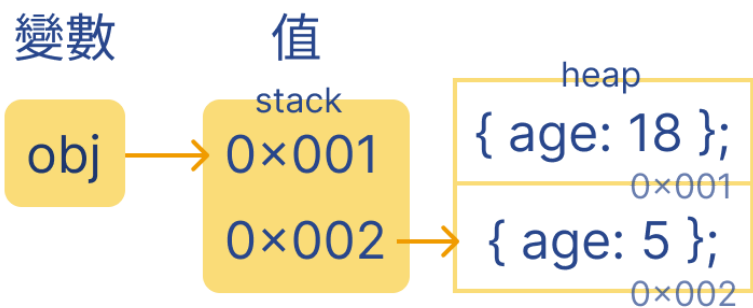

Literal
let name = "我是誰"

值可變性 Mutability

值本身在記憶體中是否可以被修改

```
let name = "我是誰";
```

```
const obj = { age: 18};
obj.age = 5;
```



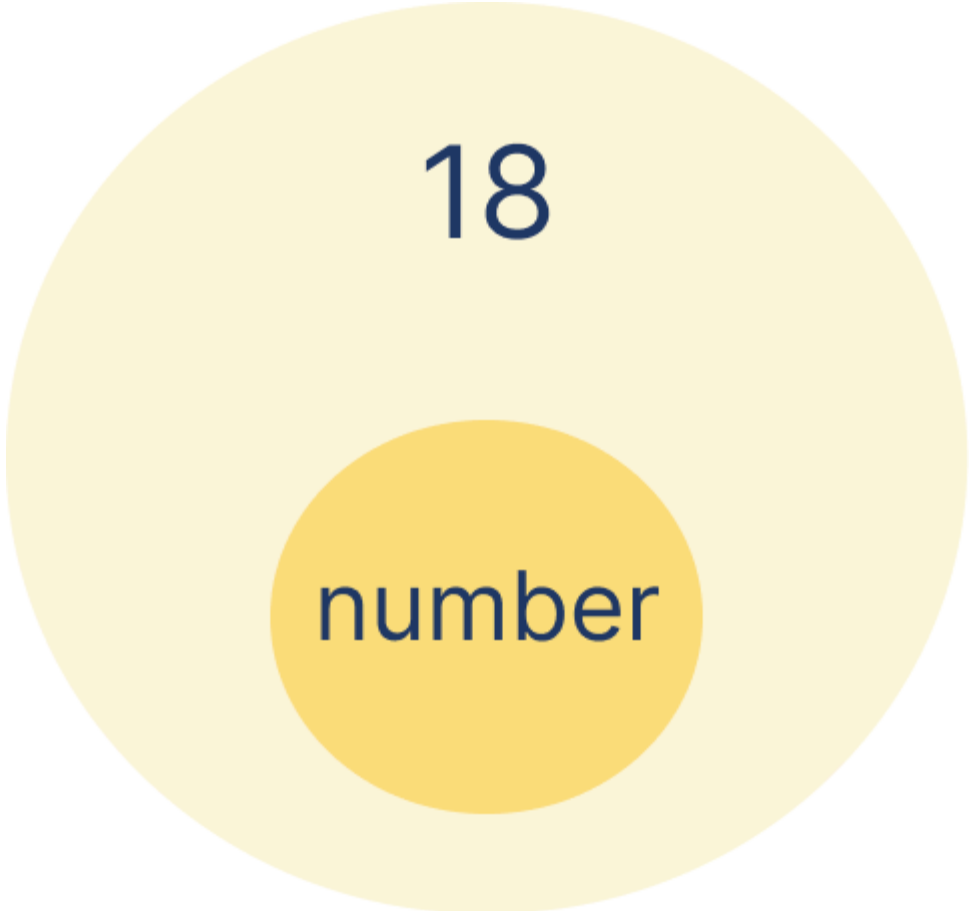
重新指派 re-assignment

變數是否可以被重新指派取決於變數宣告方式


重新指派
name
"我是誰"
重新指派
stack
heap
obj
0×001
{ age: 5 };

集合與型別範圍


any
number
"我是誰"
更寬的型別
擴展
Widening
Narrowing
限縮
更窄的型別


18
number

謝謝