為你自己學 系列課程

JavaScript 核心觀念

課程代碼 JS102

五倍紅寶石 ASTRO

常數、變數

小測驗

```
console.log(a) // 會印出什麼？
var a = 1
```

```
let a = 1
let a = 2

console.log(a) // 會印出什麼？
```

```
function checkAge(age) {
  if (age >= 18) {
    const message = "已成年"
  } else {
    const message = "未成年"
  }

  return message
}

console.log(checkAge(16)) // 會印出什麼？
console.log(checkAge(20)) // 會印出什麼？
```

「宣告」的時候發生了什麼事？

```
var a = 1
console.log(a) // 會印出什麼？
```

```
console.log(a) // 會印出什麼？
var a = 1
```

變數提昇

Variable Hoisting

```
var a
console.log(a) // 所以印出 undefined
a = 1
```

Hoisting

JavaScript **Hoisting** refers to the process whereby the interpreter appears to move the *declaration* of functions, variables or classes to the top of their scope, prior to execution of the code.

Hoisting allows functions to be safely used in code before they are declared.

Variable and class *declarations* are also hoisted, so they too can be referenced before they are declared. Note that doing so can lead to unexpected errors, and is not generally recommended.

> **Note:** The term hoisting is not used in any normative specification prose prior to ECMAScript® 2015 Language Specification. Hoisting was thought up as a general way of thinking about how execution contexts (specifically the creation and execution phases) work in JavaScript.

ref: https://developer.mozilla.org/en-US/docs/Glossary/Hoisting

建立期 vs. **執行**期

Creation Phase vs. Execution Phase

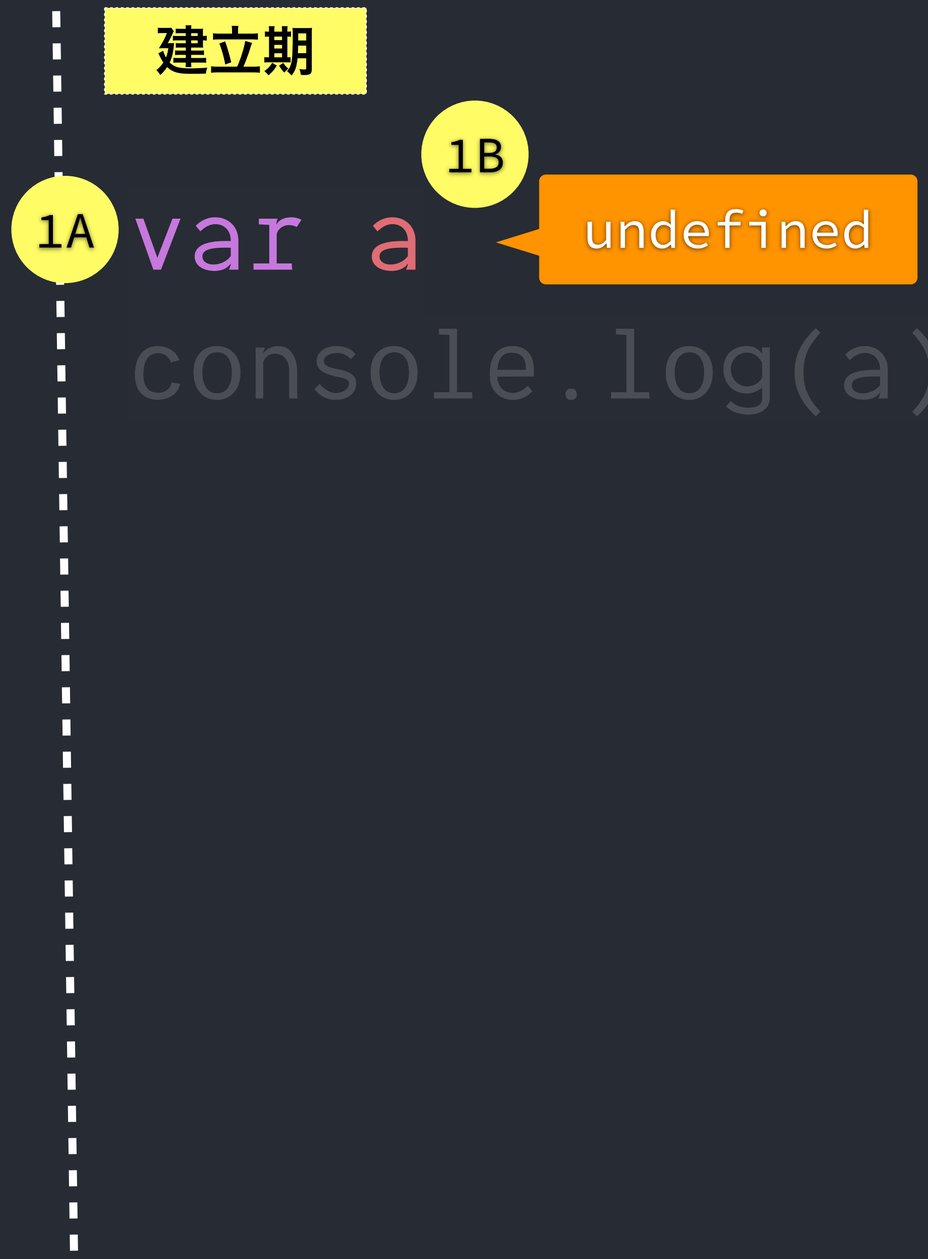
建立期

1A 註冊名稱（Identifier）+ 進行初始化 1B

執行期

2 執行函數 / 賦值

```
var a = 1
console.log(a)
```



```
var a = 1
console.log(a)
```

執行期

2 a = 1

2 console.log(a)

1

兩階段

```
console.log(a)
var a = 1
```

建立期

```
console.log(a)
var a
```

1A

undefined

1B

```
console.log(a)
var a = 1
```

執行期

```
2 console.log(a)
2 a = 1
```

1

undefined

小測驗

```
var a = 1
var a

console.log(a) // 會印出什麼？
```

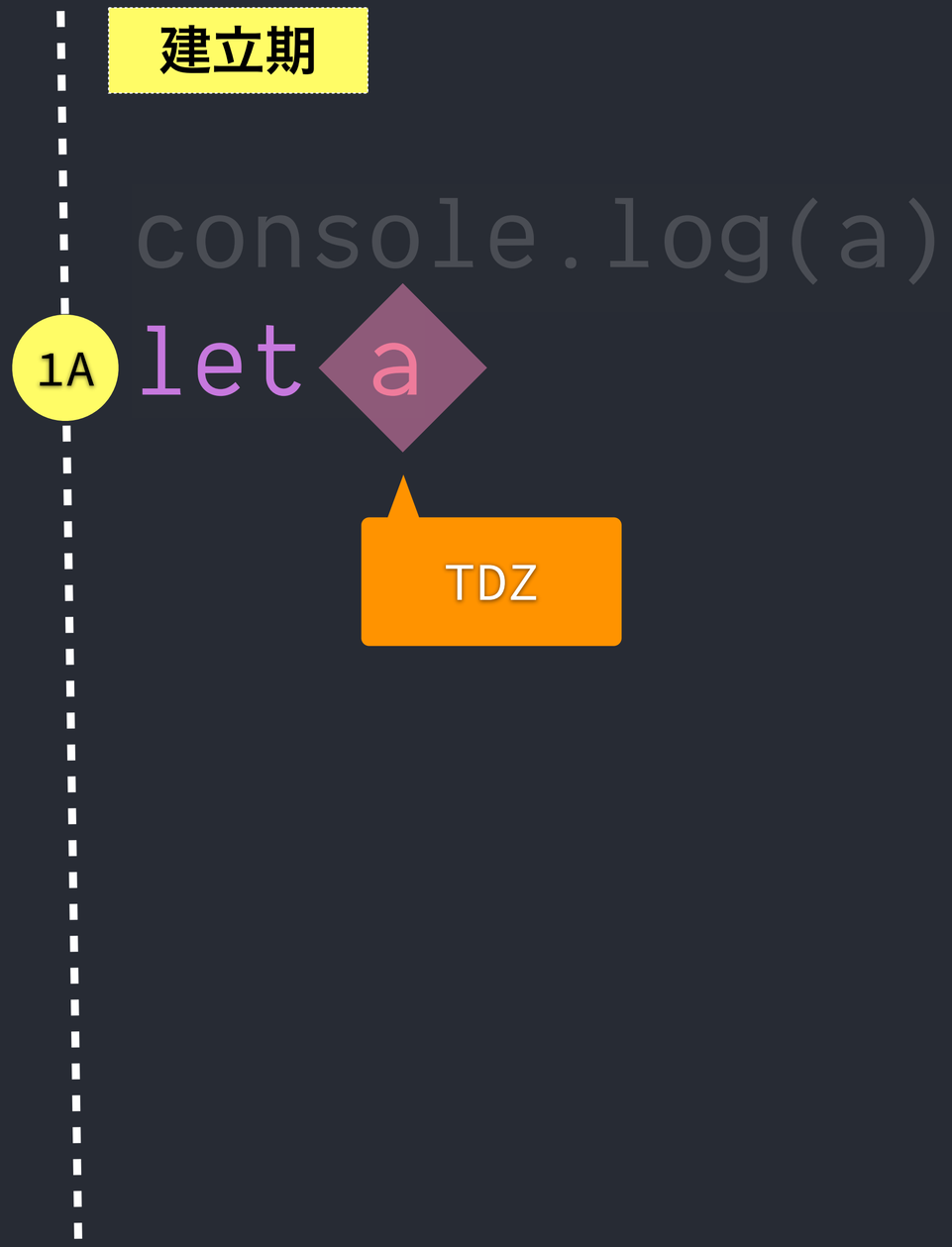
```
if (false) {
  var a = 1
}

console.log(a) // 會印出什麼？
```

另一種變數宣告

```
console.log(a) // 會印出什麼？
let a = 1
```

因為 let 不會變數提昇？

```
console.log(a)
let a = 1
```



```
console.log(a)
let a = 1
```

執行期

2

```
console.log(a)
a = 1
```

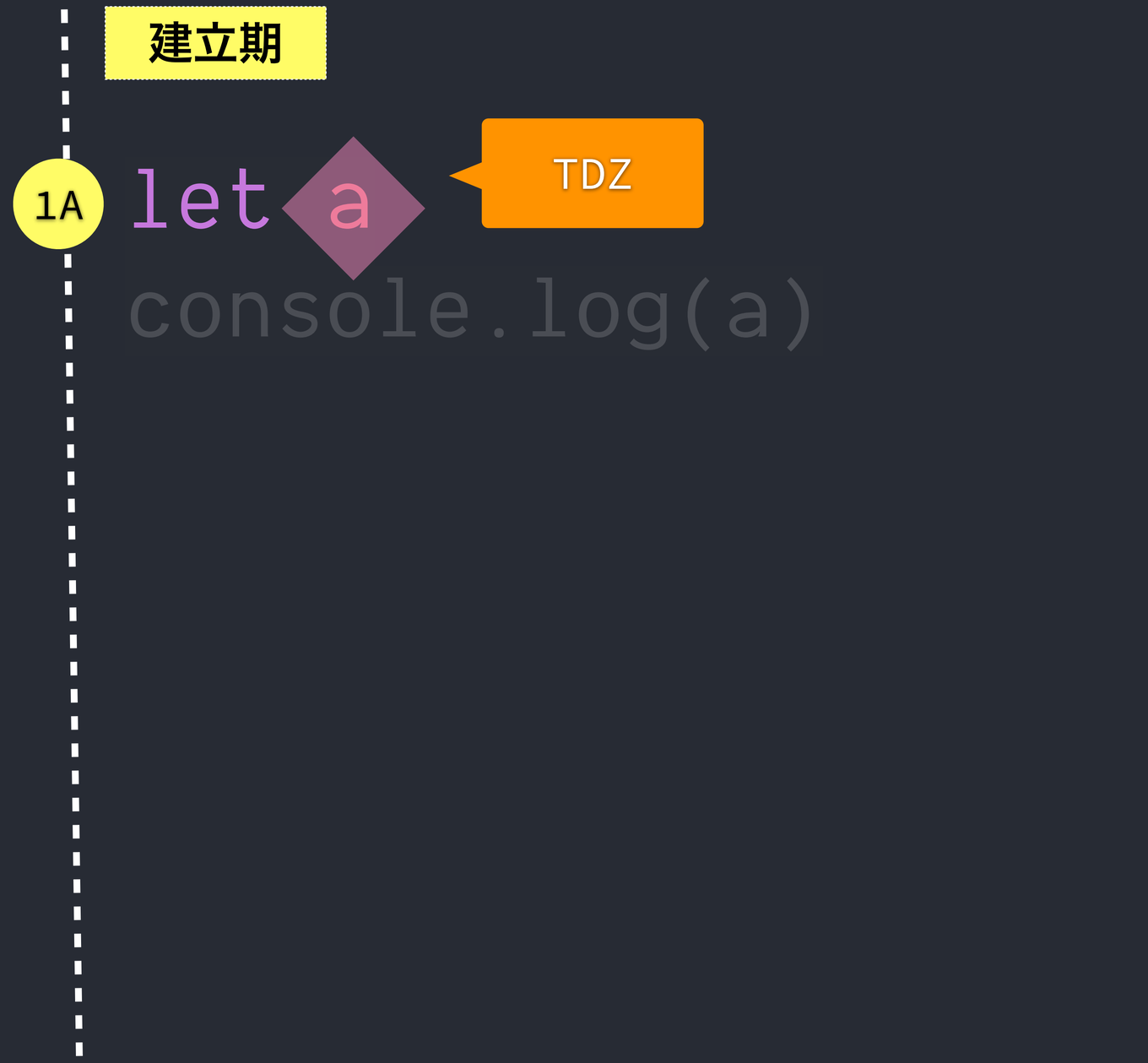
Reference Error

暫時死區

TDZ = Temporal Dead Zone

```
let a = 1
console.log(a) // 會印出什麼？
```

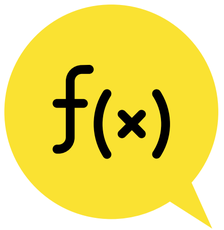
```
let a = 1
console.log(a)
```



```
let a = 1
console.log(a)
```

let = 新版的 var

以後都用 let 就好了？



函數宣告

```
sayHello() // 執行之後會看到什麼結果？

function sayHello() {
  console.log("hello")
}
```

```
sayHello()

function sayHello() {
  console.log("hello")
}
```

建立期

```
sayHello()

function sayHello() {
  console.log("hello")
}
```

1A 1B 2

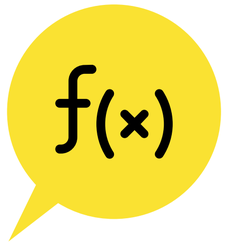
```
sayHello()

function sayHello() {
  console.log("hello")
}
```

執行期

2

```
sayHello()

function sayHello() {
  console.log("hello")
}
```

hello



不同的函數宣告方式

```
hello1() // 執行之後會看到什麼結果？
hello2() // 執行之後會看到什麼結果？

// Function Declaration
function hello1() {
  console.log("hi")
}

// Function Expression
const hello2 = () => {
  console.log("hey")
}
```

var vs let

Scope

什麼是 Scope？

Scope = 要去哪裡找東西

Function Scope vs Block Scope

Block Scope

```
if (true) {
  var age = 18
}

console.log(age) // 會印出什麼？
```

```
if (true) {
  let age = 18
}

console.log(age) // 會印出什麼？
```

Function Scope

```
function hello() {
  var a = 1
  let b = 2
}

hello()

console.log(a) // 會印出什麼？
console.log(b) // 會印出什麼？
```

重複宣告的話...

```
var age = 18
var age = 20

console.log(age) // 會印出什麼？
```

```
let age = 18
let age = 20

console.log(age) // 會印出什麼？
```

```
var age = 18
let age = 20

console.log(age) // 會印出什麼？
```

全域屬性

```
var age = 18

console.log(age) // 印出 18
console.log(window.age) // 會印出什麼？
```

```
let age = 18

console.log(age) // 印出 18
console.log(window.age) // 會印出什麼？
```

如果沒用 var / let / const 宣告的話？

var 跟 let 有什麼差別？

	var	let/const
Scope	Function Scope	Block Scope
重複宣告	可	不可
可能會造成全域屬性	會	不會
變數提昇	有	有（TDZ）

const / let / var 該怎麼選擇？

const / let / var 該怎麼選擇？

- 能用 const 就用 const，不能用 const 就用 let
- var 也還是可以用，只是它的範圍（Scope）比較大一些
- 千萬不要沒宣告就拿來用，那會造成全域屬性的污染

小測驗

```
console.log(a) // 會印出什麼？
var a = 1
```

```
let a = 1
let a = 2

console.log(a) // 會印出什麼？
```

```
function checkAge(age) {
  if (age >= 18) {
    const message = "已成年"
  } else {
    const message = "未成年"
  }

  return message
}

console.log(checkAge(16)) // 會印出什麼？
console.log(checkAge(20)) // 會印出什麼？
```