JavaScript 核心觀念（上）

JavaScript 雖然奇怪但很有趣

我在六角學院當工程師唷~

——歡迎加入六角學院

六角學院

提升(Hoisting)

```
// 請問 console.log(a); 結果為何 ?

console.log('a：', a);

var a = 'Ray';
```

1. ‘Ray’　　2.undefined　　3. 直接出錯

```
// 請問 console.log(a); 結果為何 ?

console.log('a:', a);

var a = 'Ray';
```

```
a: undefined
```

提升(Hoisting) 只是用來釐清 JavaScript 在執行階段如何運作的說法

提升（Hoisting）

提升（Hoisting）是在 ECMAScript® 2015 Language Specification 裡面找不到的專有名詞。它是一種釐清 JaveScript 在執行階段內文如何運行的思路（尤其是在創建和執行階段）。然而，提升一詞可能會引起誤解：例如，提升看起來是單純地將變數和函式宣告，移動到程式的區塊頂端，然而並非如此。變數和函數的宣告會在編譯階段就被放入記憶體，但實際位置和程式碼中完全一樣。

Hoisting 帶來什麼優點？

可以在函式宣告前呼叫

```
fn(); // My name is Ray.

function fn () {
  console.log('My name is Ray.')
}
```

只要有宣告變數，就不會出現錯誤。

```
console.log(myName); // undefined;

var myName = 'Ray';
```

如果沒有宣告變數
或函式就呼叫呢？

沒有宣告變數則會出現 not defined

```
console.log(myName); // myName is not defined
```

函式比變數有更高的優先權

```
var a = 'Ray';

function a() {
  console.log('Hello');
}

console.log('a:', a); // ?
```

```
var a = 'Ray';

function a() {
  console.log('Hello');
}

console.log(a); // Ray

// 創造階段（準備記憶體階段）

function a() {
  console.log('Hello');
}

var a;

// 執行階段（也就是賦予值的階段）

a = 'Ray';

console.log(a); // Ray
```

小測驗

解答在此：

```
fn();

function fn() {
  if (myName) {
    myName = '小明';
  }
}

var myName = 'Ray';

console.log('myName: ', myName); // ?
```

undefined

not defined

```
// 請問 console.log(a); 結果為何 ?

console.log('a：', a);

var a = 'Ray';
```

```
// 請問 console.log(a); 結果為何？

console.log('a:', a);

// var a = 'Ray';
```

```
// 請問 console.log(a); 結果為何？

console.log('a:', a);

// var a = 'Ray';
```

```
▶ Uncaught ReferenceError: a is not defined
    <anonymous> debugger eval code:4
  [了解更多]
```

undefined 與
not defined 帶來的問題

```
// 請問以下程式碼是否會正常運作？

console.log('a:', a);

var a = 'Ray';

console.log('這是一段話');
```

1.會　　2.不會

```
// 請問以下程式碼是否會正常運作？

    console.log('a:', a);

    var a = 'Ray';

    console.log('這是一段話');
```

```
a: undefined
這是一段話
```

```
// 請問以下程式碼是否會正常運作？

console.log('a:', a);

// var a = 'Ray';

console.log('這是一段話');
```

1.會　　2.不會


// 請問以下程式碼是否會正常運作？
console.log('a:', a);
// var a = 'Ray';
console.log('這是一段話');
Uncaught ReferenceError: a is not defined
<anonymous> debugger eval code:4
[了解更多]
<anonymous> debugger eval code:4

不要用 undefined 當作值

```
var a = 'Ray';

console.log(a); // Ray

// 假設在幾百行之後
a = undefined;

console.log(a); // undefined

// Ray: WT...?
```

```
var a = 'Ray';

console.log(a); // Ray

// 假設在幾百行之後
a = null;

console.log(a); // null or ''
```

ASI

Automatic Semicolon Insertion

```
var a = 'Ray'

console.log('a:', a)
```

1.錯誤

```
var a = 'Ray';

console.log('a:', a);
```

2.錯誤

```
var a = function() {
  return
  'ASI is Good!'
}

console.log(a())
```

```
» var a = function() {
    return
    'ASI is Good!'
  }

  console.log(a())
undefined
```

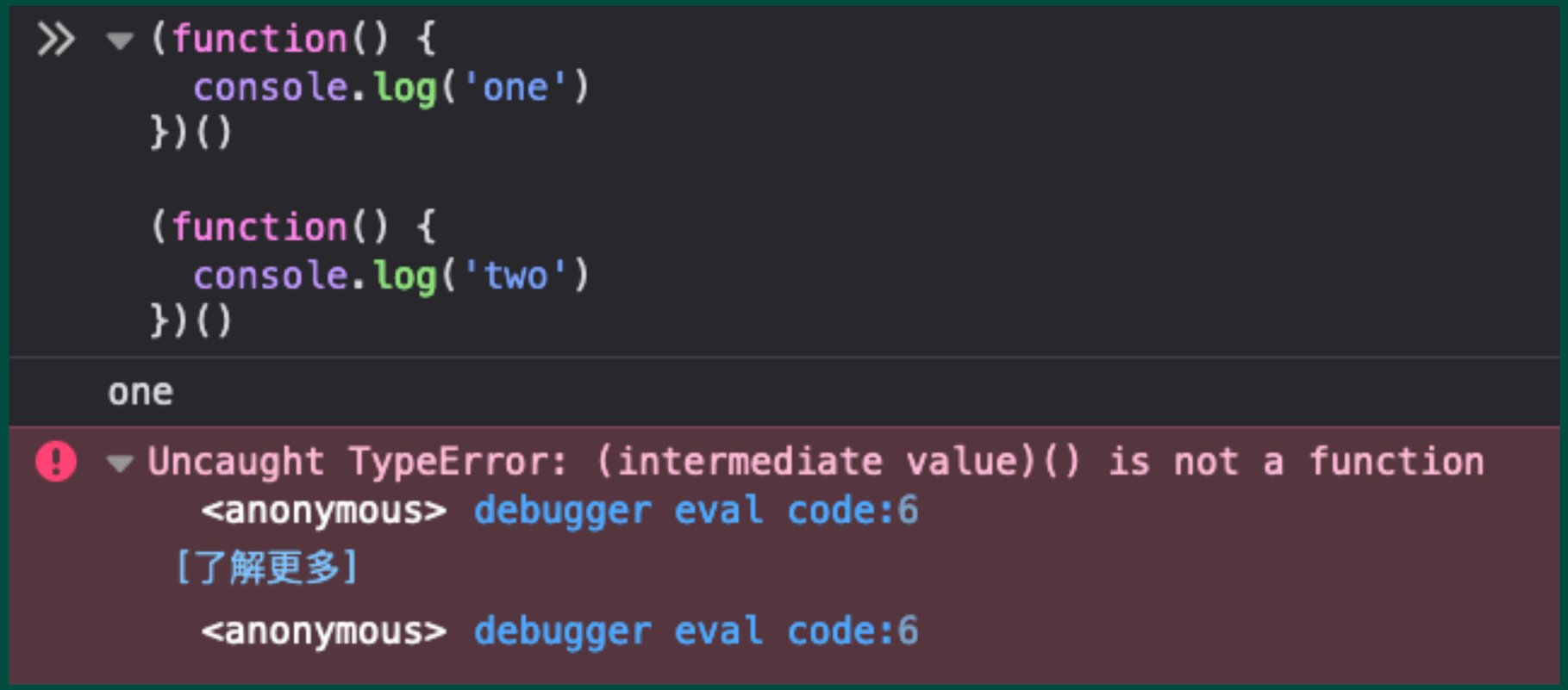
```
var a = function() {
  return;
  'ASI is Good!';
};

console.log(a());
```

```
(function() {
  console.log('one')
})()

(function() {
  console.log('two')
})()
```

1.直接出錯　　2.one、two 一起出現　　3.只有one


(function() {
console.log('one')
})()
(function() {
console.log('two')
})()
one
Uncaught TypeError: (intermediate value)() is not a function
<anonymous> debugger eval code:6
[了解更多]
<anonymous> debugger eval code:6

```
// 第一種

(function() {
  console.log('one')
})();

(function() {
  console.log('two')
})();

// 第二種

;(function() {
  console.log('one')
})()

;(function() {
  console.log('two')
})()
```

ASI 容易出現錯誤的規則

- 開頭是 (、[、/
- 大部分算數運算子 (+、–、*、%)
- 點運算子(.)
- 逗號(,)

更多 ASI 介紹可詳見：

是 Ray 不是 Array

六角學院