**Call vs Apply**

# 大綱

這兩個函數的用法和物件有關。

一般呼叫Function時的語法funcName(arg1, arg2, ...)。

[apply()](https://developer.mozilla.org/en/Core_JavaScript_1.5_Reference/Global_Objects/Function/apply)與[call()](https://developer.mozilla.org/en/Core_JavaScript_1.5_Reference/Global_Objects/Function/call)是Function類別的兩個Method。call()與apply()用途相同，差別在於，call() 必須列出每一個參數。例如funcName.call(thisArg, arg1, arg2, ...)。而apply()的用法是funcName.apply(thisArg, argsArray)，第一個傳入的參數在就是函數中的this，而第二個是參數陣列。兩者之間的關係應該是：apply(thisArg,[arg1,arg2,arg3]) 轉成

call(thisArg,arg1,arg2,arg3)

【範例】apply

**<script** type**=**'text/javascript'**>**

$(**function**() {

**function** btnClick() {

 alert("Button[" **+** **this**.value **+** "] clicked!");

}

$("#a").click(btnClick);

**var** virtualButton **=** { value: "V" };

btnClick.apply(virtualButton);

});

**</script>**

**</head><body>**

**<input** type**=**"button" value**=**"A" id**=**"a" **/>**

**</body>**

btnClick 是一個沒有參數的函數（function）。一般的函數呼叫方法是 btnClick() 就可以了。但是，範例中的btnClick() 的設計目的是配合其他物件，(因為函數有"this" )，因此可以配合call或apply 方法。以apply 方法來說，btnClick.apply(virtualButton) 也是呼叫btnClick ()，但是透過apply機制，這裡的virtualButton 被當成this的參數，傳入btnClick() ，被認為是"this"。因此，this.value 指的是 變數virtualButton 欄位value 的值 "V"。

##### <參考操作[[js]06 prototype](%5BJS%5D%2006%20prototype.pptx)>

【測試】利用參數的stack，允許thisp的傳入(不是關鍵字this)

this的傳入，除了利用call,apply 之外，還可以利用javascript對參數處理的彈性：

 **<script>**

 **function** showArg(arg1) {

 console.log(**this**);

 console.log(arg1);

 console.log(arguments[0]);

 console.log(arguments[1]);

 thisp=arguments[1];

 }

 showArg("me", "pine");

 showArg.call("apple", "me", "pine");

 **</script>**



也就是說，雖然showArg的參數只有1個，但是仍然可以再呼叫的時候傳入多於一個的參數，此時，showArg的body可以利用argument[1]來存取。

【例】上述測試的應用

every 是一個Array內建方法，調用使用者提供的函數（這裡isBigEnough），依照說明書，isBigEnough應該有3個參數：元素值，元素索引，整個陣列。

**<html>**

 **<head>**

 **<title>**JavaScript Array every Method**</title>**

 **</head>**

 **<body>**

 **<script** type**=**"text/javascript"**>**

 Array.**prototype**.every **=** **function**(fun /\*, thisp\*/)

 {

 **var** len **=** **this**.length;

 **if** (**typeof** fun !**=** "function")

 **throw** **new** TypeError();

 **var** thisp **=** arguments[1];

 **for** (**var** i **=** 0; i < len; i**++**)

 {

 **if** (i **in** **this** && !fun.call(thisp, **this**[i], i, **this**)) //<1>

 **return** **false**;

 }

 **return** **true**;

 };

 **function** isBigEnough(element, index, array) {

 console.log(**this**);

 **return** (element **>=** 10);

 }

 **var** passed **=** [12, 5, 8, 130, 44].every(isBigEnough,"arg2"); //<2>

 document.write("First Test Value : " **+** passed );

 passed **=** [12, 54, 18, 130, 44].every(isBigEnough,"arg2");

 document.write("Second Test Value : " **+** passed );

 **</script>**

 **</body>**

**</html>**



<註>

1

thisp 會指向[arg2](#refAnker_283439) 。

2

arg2 被當成this 傳入；[isBitEnough](#refAnker_335720)(只有三個參數，不包括arg2)

</註>

範例：陣列合併

**var** a **=** [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ];

**var** b **=** [ "foo", "bar", "baz", "bam", "bun", "fun" ];

// `b` onto `a`:

a.push.apply( a, b );

a; // [1,2,3,4,5,6,7,8,9,"foo","bar","baz","bam","bun","fun"]

// or `a` into `b`:

b.unshift.apply( b, a );

b; // [1,2,3,4,5,6,7,8,9,"foo","bar","baz","bam","bun","fun"]