

this和arguments的引入

函数的调用

严格模式下的函数调用

js里面一些很少用的特性

arguments

this和arguments的引入

call的第一个参数可以用this得到;

call的后面的参数可以用arguments得到;

`f.call(undefined, 1, 2) // 3`

↓ ↓
this [1, 2] arguments

1. call 的第一个参数可以用 this 得到
2. call 的后面 arguments

函数的调用

先记住这种形式:

```
1 function f(x,y){return x+y}
2 undefined
3 f.call(undefined,1,2)
4 3
5 f.call(undefined,2,4)
6 6
7
```

```

>> function f(x,y){return x+y}
< undefined
>> f.call(undefined,1,2)
< 3
>> f.call(undefined,2,4)
< 6
>>

```

在普通模式下面,如果this是undefined,浏览器会自动把this变成window;

```

>> f = function(){
  console.log(this)
  console.log(arguments)
}
< ▶ function f()
>> f.call(undefined,1,2,3)
< undefined
  ▶ Window about:newtab
  ▼ Arguments
    0: 1
    1: 2
    2: 3
    ▶ callee: function f()
      length: 3
    ▶ Symbol(Symbol.iterator): function values()
    ▶ <prototype>: Object { ... }
>>

```

注意是小写的window,不是大写的window!!!如下:

```

>> f = function(){
  console.log(this)
}
< ▶ function f()
>> f.call(undefined)
< undefined
  ▶ Window https://home.firefoxchina.cn/
>> f = function(){
  console.log(this)
  console.log(this===window)
}
< ▶ function f()
>> f.call(undefined)
< undefined
  ▶ Window https://home.firefoxchina.cn/
  true
>> |

```



严格模式下的函数调用

```

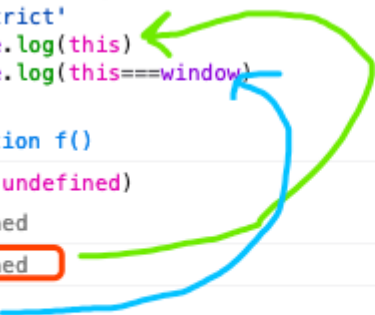
1 f = function(){ 'use strict' console.log(this) console.log(this===window) }
2 function f()
3
4 f.call(undefined)
5 undefined
6 undefined debugger eval code:3:9

```

```
7 false
```


```
8
```

```
>> f = function(){  
  'use strict'  
  console.log(this)  
  console.log(this===window)  
}  
< ▶ function f()  
>> f.call(undefined)  
< undefined  
undefined  
false  
>>
```




对比一下上下两个图:

```
>> f = function(){  
  console.log(this)  
  console.log(arguments)  
}  
< ▶ function f()  
>> f.call(undefined,1,2,3)  
< undefined  
▶ Window about:newtab  
▶ Arguments { 0: 1, 1: 2, 2: 3, ... }  
>>
```



由下面的例子可以看出,this指的就是第一个参数:

```
>> f = function(){  
  'use strict'  
  console.log(this)  
  console.log(this===window)  
}  
< ▶ function f()  
>> f.call(undefined)  
< undefined  
undefined  
false  
>> f.call(1)  
< undefined  
1  
false  
>> f.call('good')  
< undefined  
good  
false  
>>
```



```

1 f = function(){ 'use strict'console.log(this) console.log(this===window) }
2 function f()
3
4 f.call(undefined)
5 undefined
6 undefined debugger eval code:3:9
7 false debugger eval code:4:9
8 f.call(1)
9 undefined
10 1 debugger eval code:3:9
11 false debugger eval code:4:9
12 f.call('good')
13 undefined
14 good debugger eval code:3:9
15 false
16

```

js里面一些很少用的特性

因为当初老板需要js之父将js长的像“java”,所以才有this,其实没什么太大意义,在js里面

75825正在观看

var n=1

var n = new Number(1) X

长得像Ja+

15195575825正在观看

var f = (x, y) => x+y

f = new Function('x', 'y') X

15195575825正在观看

f.call(1, 2)

arguments[1, 2]

15195575825正在观看

var n = 1

var n = new Number(1) X

长得像 Java

var f = (x, y) => x + y

f = new Function('x' —) X

new
this

f.call(1, 2, 3)

this

[2, 3] arguments

15195575825正在观看

15195575825正在观看

arguments

arguments是伪数组,其没有数组的共有属性;

或者说伪数组是没有push方法的!

```
>> f = function(){
  // console.log(this)
  console.log(arguments)
}
```

```
< ▶ function f()
```

```
>> f.call(undefined, 1, 2, 3)
```

```
< undefined
```

▼ Arguments

0: 1

1: 2

2: 3

▶ callee: function f()

length: 3

▶ Symbol(Symbol.iterator): function values()

▶ <prototype>: Object { ... }

```
>> f = function(){
  console.log(arguments)
  arguments.push(4)
}
```

```
< ▶ function f()
```

```
>> f.call(undefined, 1, 2, 3)
```

```
❌ ▶ TypeError: arguments.push is not a function [详细了解]
```

```
▶ Arguments { 0: 1, 1: 2, 2: 3, ... }
```

```
>>
```