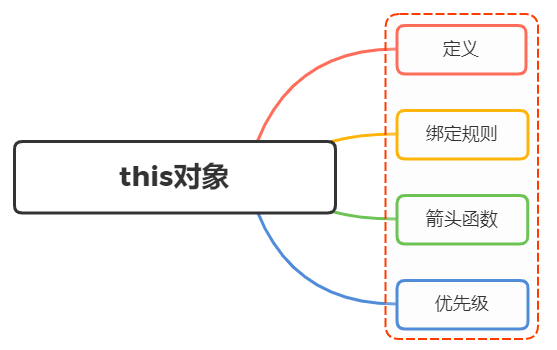
# 面试官：谈谈this对象的理解



## 一、定义

函数的 this 关键字在 JavaScript 中的表现略有不同，此外，在严格模式和非严格模式之间也会有一些差别

在绝大多数情况下，函数的调用方式决定了 this 的值（运行时绑定）

this 关键字是函数运行时自动生成的一个内部对象，只能在函数内部使用，总指向调用它的对象

举个例子：

function baz() {  
 // 当前调用栈是：baz  
 // 因此，当前调用位置是全局作用域  
   
 console.log( "baz" );  
 bar(); // <-- bar的调用位置  
}  
  
function bar() {  
 // 当前调用栈是：baz --> bar  
 // 因此，当前调用位置在baz中  
   
 console.log( "bar" );  
 foo(); // <-- foo的调用位置  
}  
  
function foo() {  
 // 当前调用栈是：baz --> bar --> foo  
 // 因此，当前调用位置在bar中  
   
 console.log( "foo" );  
}  
  
baz(); // <-- baz的调用位置

同时，this在函数执行过程中，this一旦被确定了，就不可以再更改

var a = 10;  
var obj = {  
 a: 20  
}  
  
function fn() {  
 this = obj; // 修改this，运行后会报错  
 console.log(this.a);  
}  
  
fn();

## 二、绑定规则

根据不同的使用场合，this有不同的值，主要分为下面几种情况：

* 默认绑定
* 隐式绑定
* new绑定
* 显示绑定

### 默认绑定

全局环境中定义person函数，内部使用this关键字

var name = 'Jenny';  
function person() {  
 return this.name;  
}  
console.log(person()); //Jenny

上述代码输出Jenny，原因是调用函数的对象在游览器中位window，因此this指向window，所以输出Jenny

注意：

严格模式下，不能将全局对象用于默认绑定，this会绑定到undefined，只有函数运行在非严格模式下，默认绑定才能绑定到全局对象

### 隐式绑定

函数还可以作为某个对象的方法调用，这时this就指这个上级对象

function test() {  
 console.log(this.x);  
}  
  
var obj = {};  
obj.x = 1;  
obj.m = test;  
  
obj.m(); // 1

这个函数中包含多个对象，尽管这个函数是被最外层的对象所调用，this指向的也只是它上一级的对象

var o = {  
 a:10,  
 b:{  
 fn:function(){  
 console.log(this.a); //undefined  
 }  
 }  
}  
o.b.fn();

上述代码中，this的上一级对象为b，b内部并没有a变量的定义，所以输出undefined

这里再举一种特殊情况

var o = {  
 a:10,  
 b:{  
 a:12,  
 fn:function(){  
 console.log(this.a); //undefined  
 console.log(this); //window  
 }  
 }  
}  
var j = o.b.fn;  
j();

此时this指向的是window，这里的大家需要记住，this永远指向的是最后调用它的对象，虽然fn是对象b的方法，但是fn赋值给j时候并没有执行，所以最终指向window

### new绑定

通过构建函数new关键字生成一个实例对象，此时this指向这个实例对象

function test() {  
　this.x = 1;  
}  
  
var obj = new test();  
obj.x // 1

上述代码之所以能过输出1，是因为new关键字改变了this的指向

这里再列举一些特殊情况：

new过程遇到return一个对象，此时this指向为返回的对象

function fn()   
{   
 this.user = 'xxx';   
 return {};   
}  
var a = new fn();   
console.log(a.user); //undefined

如果返回一个简单类型的时候，则this指向实例对象

function fn()   
{   
 this.user = 'xxx';   
 return 1;  
}  
var a = new fn;   
console.log(a.user); //xxx

注意的是null虽然也是对象，但是此时new仍然指向实例对象

function fn()   
{   
 this.user = 'xxx';   
 return null;  
}  
var a = new fn;   
console.log(a.user); //xxx

### 显示修改

apply()、call()、bind()是函数的一个方法，作用是改变函数的调用对象。它的第一个参数就表示改变后的调用这个函数的对象。因此，这时this指的就是这第一个参数

var x = 0;  
function test() {  
　console.log(this.x);  
}  
  
var obj = {};  
obj.x = 1;  
obj.m = test;  
obj.m.apply(obj) // 1

关于apply、call、bind三者的区别，我们后面再详细说

## 三、箭头函数

在 ES6 的语法中还提供了箭头函语法，让我们在代码书写时就能确定 this 的指向（编译时绑定）

举个例子：

const obj = {  
 sayThis: () => {  
 console.log(this);  
 }  
};  
  
obj.sayThis(); // window 因为 JavaScript 没有块作用域，所以在定义 sayThis 的时候，里面的 this 就绑到 window 上去了  
const globalSay = obj.sayThis;  
globalSay(); // window 浏览器中的 global 对象

虽然箭头函数的this能够在编译的时候就确定了this的指向，但也需要注意一些潜在的坑

下面举个例子：

绑定事件监听

const button = document.getElementById('mngb');  
button.addEventListener('click', ()=> {  
 console.log(this === window) // true  
 this.innerHTML = 'clicked button'  
})

上述可以看到，我们其实是想要this为点击的button，但此时this指向了window

包括在原型上添加方法时候，此时this指向window

Cat.prototype.sayName = () => {  
 console.log(this === window) //true  
 return this.name  
}  
const cat = new Cat('mm');  
cat.sayName()

同样的，箭头函数不能作为构建函数

## 四、优先级

### 隐式绑定 VS 显式绑定

function foo() {  
 console.log( this.a );  
}  
  
var obj1 = {  
 a: 2,  
 foo: foo  
};  
  
var obj2 = {  
 a: 3,  
 foo: foo  
};  
  
obj1.foo(); // 2  
obj2.foo(); // 3  
  
obj1.foo.call( obj2 ); // 3  
obj2.foo.call( obj1 ); // 2

显然，显示绑定的优先级更高

### new绑定 VS 隐式绑定

function foo(something) {  
 this.a = something;  
}  
  
var obj1 = {  
 foo: foo  
};  
  
var obj2 = {};  
  
obj1.foo( 2 );  
console.log( obj1.a ); // 2  
  
obj1.foo.call( obj2, 3 );  
console.log( obj2.a ); // 3  
  
var bar = new obj1.foo( 4 );  
console.log( obj1.a ); // 2  
console.log( bar.a ); // 4

可以看到，new绑定的优先级>隐式绑定

### new绑定 VS 显式绑定

因为new和apply、call无法一起使用，但硬绑定也是显式绑定的一种，可以替换测试

function foo(something) {  
 this.a = something;  
}  
  
var obj1 = {};  
  
var bar = foo.bind( obj1 );  
bar( 2 );  
console.log( obj1.a ); // 2  
  
var baz = new bar( 3 );  
console.log( obj1.a ); // 2  
console.log( baz.a ); // 3

bar被绑定到obj1上，但是new bar(3) 并没有像我们预计的那样把obj1.a修改为3。但是，new修改了绑定调用bar()中的this

我们可认为new绑定优先级>显式绑定

综上，new绑定优先级 > 显示绑定优先级 > 隐式绑定优先级 > 默认绑定优先级

## 相关链接

* https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/this