### 彻底理解js中的this指向

1. this的指向在函数定义的时候是确定不了的，只有函数执行的时候才能确定this到底指向谁，实际上this的最终指向的是那个调用它的对象

例子1：

|  |
| --- |
| function a(){  var user = "追梦子";  console.log(this.user); //undefined  console.log(this); //Window  }  a(); |

按照我们上面说的this最终指向的是调用它的对象，这里的函数a实际是被Window对象所点出来的，下面的代码就可以证明。

|  |
| --- |
| function a(){  var user = "追梦子";  console.log(this.user); //undefined  console.log(this);　　//Window  }  window.a(); |

例子2：

|  |
| --- |
| var o = {  user:"追梦子",  fn:function(){  console.log(this.user); //追梦子  }  }  o.fn(); |

这里的this指向的是对象o，因为你调用这个fn是通过o.fn()执行的，那自然指向就是对象o，这里再次强调一点，this的指向在函数创建的时候是决定不了的，在调用的时候才能决定。

例子3：

|  |
| --- |
| var o = {  user:"追梦子",  fn:function(){  console.log(this.user); //追梦子  }  }  window.o.fn(); |

这段代码和上面的那段代码几乎是一样的，但是这里的this为什么不是指向window，如果按照上面的理论，最终this指向的是调用它的对象，这里先说个而外话，window是js中的全局对象，我们创建的变量实际上是给window添加属性，所以这里可以用window点o对象。

继续看例子：

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

这里fn也是从o开始点出来的，可最终的this.a并没有指向10；

结论：

情况1：如果一个函数中有this，但是它没有被上一级的对象所调用，那么this指向的就是window，这里需要说明的是在js的严格版中this指向的不是window，但是我们这里不探讨严格版的问题，你想了解可以自行上网查找。

情况2：如果一个函数中有this，这个函数有被上一级的对象所调用，那么this指向的就是上一级的对象。

情况3：如果一个函数中有this，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。

继续看：

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

这里的this指的是o.b这个对象；

还有种特殊情况：

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

这里this指向的是window，因为o.b.fin赋值给了变量j，这里只是赋值，并没有调用，j()这里调用的时候，相当于window.j()，this自然指向的是window。