链滴

对箭头函数的 this 深度理解，它究竟指向谁？

作者：xiluotop
原文链接：https://ld246.com/article/1571540886424
来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20180702.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
<h2 id="简介">简介</h2>
面向对象编程语言中 this 是一个非常重要的关键字，其在函数执行中通过 this 来明确操作的
象，在 JS 中，JS 并非面向对象语言，但是他也有 this 关键字，用来指向函数的调用对象，博主在学
es5 的时候对 this 理解非常容易，因为以前学过面向对象语言，也曾大量的使用，但是学习到 es6
箭头函数，它的 this 指向就让我有点困惑，于是花了一些时间从各个技术博客， MDN 文档，还有
些其他的资料进行了研究，总算是对箭头函数的 this 指向有个深刻的认识了，下面就会讲述下我自己
箭头函数 this 的理解。
<h2 id="this-的重要性">this 的重要性</h2>
this 在编程语言中用的次数非常多，多到你不知不觉就会下意识的写出 this，比如事件绑定，对
件源的操作，比如对一个对象的相关属性操作，继承中的 this 使用等。所以 this 的指向是一个一直
要理解并掌握的一个知识，如果不清楚 this 的指向，那么很多方法就会出现大问题，并且非语法的 b
g 维护起来更是令人头大。严格模式中的 this 在全局中指向 undefined，其他地方下并没有什么影
，所以一下论点也只讨论在非严格模式下的 this 指向。
<h2 id="普通函数中的-this">普通函数中的 this</h2>
普通函数中的 this 很好理解，无非以下四点：

直接调用函数，this 指向全局 window
对象调用函数，this 指向这个对象
构造函数中的 this 指向将要实例化的对象
call,apply,bind 可以改变函数执行时内部的 this 指向

总结一句话就是谁调用函数，this 就指向谁，普通函数的 this 取决于执行时的函数。
<h2 id="箭头函数中的-this">箭头函数中的 this</h2>
<blockquote>
下面重点介绍下箭头函数中的 this 问题
</blockquote>
箭头函数的语法：
<pre><code class="language-js highlight-chroma"><span clas
="highlight-cl">// 无参数直接输出一句话
<span cla
s="highlight-c1">var <span class="highlight-n
">fun1 = ()
=> console<spa
class="highlight-p">.log<span class="highligh
-p">('hello');</spa
>
<span class="high
ight-c1">// 有一个参数并返回 x*x
<span cla
s="highlight-c1">var <span class="highlight-n
">fun2 = x
=> x<span class
"highlight-o">*x;</
pan>
<span class="high
ight-c1">// 有一个参数并返回 y*y
<span cla
s="highlight-c1">var <span class="highlight-n
">fun3 = (<
pan class="highlight-nx">y) <span class="highli
ht-p">=> y*</s
an>y;

<span class="high
ight-c1">// 有两个参数并返回 x+y，也可以简写 (x,y) => x+y;
<span cla
s="highlight-c1">var <span class="highlight-n
">fun4 = (<
pan class="highlight-nx">x,<span class="highli
ht-nx">y) =></s
an> {
  <span class="h
ghlight-k">return x<span class="highlight-o"
+y;
<span class="high
ight-p">}
</code></pre>
如果箭头函数中没有用到 this 的话，那么大可放心的直接使用，因为代码写的更少更方便，但
果需要 this，那么一定得清楚箭头函数中 this 指向谁。箭头函数的 this 指向也有不同的说法，下面
举出不同说法。
<blockquote>
</blockquote>

箭头函数中没有单独的 this，this 值取决于箭头函数所在的环境
箭头函数没有自己的 this, 它的 this 是继承而来; 默认指向在定义它时所处的对象(宿主对象)
箭头函数的 this 遵循词法作用域，指向其所属环境的执行上下文（也可以说是宿主对象）。

第一种说法较为模糊，概念不是那么的清晰，第二种说法通过继承而来有点牵强的感觉，而且我
人觉的有点误导的感觉，因此我看到第三种说法时，虽然觉得有点陌生，说法非常官方的感觉，但是
觉得自习深入了解第三种说法，应该能完全掌握箭头函数的 this，所以就仔细研究了一下，下面将展
对第三种说法的论点，也是本篇的核心（前面一堆废话，凑字数<img alt="huaji" class="emoji" src
"https://unpkg.com/vditor/dist/images/emoji/huaji.gif" title="huaji">）
<h2 id="词法作用域-执行上下文">词法作用域，执行上下文</h2>
<blockquote>
词法作用域简单来说指的是函数作用域的一种工作模式，所以词法作用域
法则是基于作用域的概念。ES6 之前作用域分为全局作用域、局部作用域，变量遵循词法作用域，ES6
引入了块级作用域，使得 JS 也能像其他的编程语言有了真正的块级代码。执行上下文</str
ng>其实就是执行环境，也就是当前的 this，这里有点绕了吧，其实没关系，下面会说明的，只不过
时是把上面的第二种继承方式的原理说明了，箭头函数的 this 就取决于这个执行上下文的 this，因此
说他是通过继承而来。
</blockquote>
有了上面的知识作为根基，那么究竟怎么理解此法作用域，和执行上下文，以及如果确定箭头函
的 this 指向，接下来继续说明。
先来一段代码：
<pre><code class="language-js highlight-chroma"><span clas
="highlight-cl">var
num </spa
>= <span cl
ss="highlight-mi">100;
<span class="high
ight-kd">var <span class="highlight-nx
>obj =</sp
n> {
<span class="high
ight-err">   fun1<sp
n class="highlight-o">: <span class="hi
hlight-kd">function <span class="highli

ht-p">() {</
pan>
<span class="high
ight-err">   <span class="hi
hlight-nx">num <span class="highlight
o">= 200<
span>;
<span class="high
ight-err">   <span class="hi
hlight-nx">console.
og(num<s
an class="highlight-p">);
<span class="high
ight-err">   },
<span class="high
ight-err">   fun2<sp
n class="highlight-o">: <span class="hi
hlight-kd">function <span class="highli
ht-p">() {</
pan>
<span class="high
ight-err">   <span class="hi
hlight-kd">var <span class="highlight-n
">num =</
pan> 300<
pan class="highlight-p">;
<span class="high
ight-err">   <span class="hi
hlight-nx">console.
og(num<s
an class="highlight-p">)
<span class="high
ight-err">   }
<span class="high
ight-p">}
<span class="high
ight-nx">obj.fun1<
span>(); &nb
p;   // N
.2 200
<span cla
s="highlight-c1">obj<span class="highlight-p
>.fun2();<
pan class="highlight-err">   <
span>// No.3 300
<span cla
s="highlight-c1">console<span class="highlig
t-p">.log(
span class="highlight-nx">num);<span class="h
ghlight-err">   // No.1
00
</code></pre>
<p>从上述代码中，fun1 执行时为 num 赋值，但是可以从全局中寻找到 num，因此对全局的 num 进行操作，fun2 执行时在自己的局部作用域（函数）声明了一个 num，此时的 num 为局部的，与

局的 num 无关，fun2 执行完毕后局部 num 就消失了，所以全局的 num 最终结果为 200，这一段码中的变量使用的法则，遵循的就是此法作用域，说白点就是寻找变量的过程和其生命周期的范围受法作用域约束，另一个隐藏的知识点就是 obj.fun1(),obj.fun2() 执行时的执行上下文就是 obj, this 是 obj，执行环境就是 obj。
<hr>
现在真正进入箭头函数的 this 讨论，如果有点忘了箭头函数 this 指向的第三种说明，现在可以马向上翻滚看一下 <img alt="huaji" class="emoji" src="https://unpkg.com/vditor/dist/images/moji/huaji.gif" title="huaji">
看如下 Demo：
<pre><code class="language-js highlight-chroma">var obj</san> = {};
var fun1
 function ()
{
  console.<span class="highlight-nx>log(this<san class="highlight-p">);
}
var fun2
 () => <san class="highlight-p">{
  console.<span class="highlight-nx>log(this<san class="highlight-p">);
}
console.lg('normal----------
);
// 全局环境下直接调用
<span cla s="highlight-c1">fun1();
fun2();
console.lg('call----------'</pan>);
// 通过 call 进行执行环境的绑定
<span cla s="highlight-c1">fun1.call(<spn class="highlight-nx">obj);
<span class="high

ight-nx">fun2.call<
span>(obj<span cla
s="highlight-p">);
</code></pre>
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:
/b3logfile.com/file/2019/10/image-c5a7ff6b.png?imageView2/2/interlace/1/format/jpg"><b
>
从执行结果来看，call 不会对箭头函数进行绑定影响，也就是说箭头函数从他定义的那一刻时，它的 t
is 就已经确定了，无法通过 call 更改，apply 也是同样的。
再看下一段代码：
<pre><code class="language-js highlight-chroma"><span clas
="highlight-cl">var obj</s
an> = {
  <span class="h
ghlight-nx">fun1: f
nction () {
    <span class=
highlight-nx">console.<span class="highlight-n
">log(this<
pan class="highlight-p">);
  <span class="h
ghlight-p">},
  <span class="h
ghlight-nx">fun2: ()
/span> => {
    <span class=
highlight-nx">console.<span class="highlight-n
">log(this<
pan class="highlight-p">);
  <span class="h
ghlight-p">}
<span class="high
ight-p">}
<span class="high
ight-nx">obj.fun1<
span>();
<span class="high
ight-nx">obj.fun2<
span>();
</code></pre>
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:
/b3logfile.com/file/2019/10/image-23f32920.png?imageView2/2/interlace/1/format/jpg"><b
>
结果看出普通函数执行时 this 取决于执行环境（执行上下文）也就是 obj，而箭头函数的 this 却指向
window，使用 call 能改变吗？
<pre><code class="language-js highlight-chroma"><span clas
="highlight-cl">obj.</span
fun1();
<span class="high
ight-nx">obj.fun2<
span>();
<span class="high
ight-nx">obj.fun2<
span>.call<span cla

s="highlight-p">(obj<span class="highlight-p
>);
</code></pre>
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:
/b3logfile.com/file/2019/10/image-f07ee098.png?imageView2/2/interlace/1/format/jpg"><b
>
很显然不能。
根据第三种说法解释：<code>箭头函数的 this 指向也遵循**词法作用域**，指向当前环境的**执行
下文**</code>

当前的词法作用域：依赖作用域，当前作用域是全局作用域。
当前环境上下文：全局作用域的环境上下文 this 就是 window

再来一段代码巩固下：
<pre><code class="language-js highlight-chroma"><span clas
="highlight-cl">var obj</s
an> = {
  <span class="h
ghlight-nx">fun1: f
nction () {
    <span class=
highlight-nx">setTimeout(<span class="highligh
-kd">function () {</
pan>
      <span clas
="highlight-nx">console.<span class="highlight
nx">log('普通函数'<
span>, this<span cla
s="highlight-p">);
    <span class=
highlight-p">})
  <span class="h
ghlight-p">},
  <span class="h
ghlight-nx">fun2: f
nction () {
    <span class=
highlight-nx">setTimeout(() <span class="highli
ht-p">=> {
      <span clas
="highlight-nx">console.<span class="highlight
nx">log('箭头函数'<
span>, this<span cla
s="highlight-p">);
    <span class=
highlight-p">})
  <span class="h
ghlight-p">}
<span class="high
ight-p">}
<span class="high
ight-nx">obj.fun1<
span>();
<span class="high

ight-nx">obj.fun2<
span>();
</code></pre>
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:
/b3logfile.com/file/2019/10/image-38ec88f9.png?imageView2/2/interlace/1/format/jpg"></
>

普通函数没得说，计时器时间到执行回调函数的话则在全局环境中执行，因此 this 指向 window
/li>
箭头函数（this 在箭头函数定义时就确定，遵循词法作用域，指向执行上下文对象（也可以说宿
对象））

词法作用域：当前所属环境为局部作用域，因为被定义在 function 内
执行上下文：function 的执行上下文将来是在 obj 环境（除非用 call,apply,后面还会说明），所以
this 已经在箭头函数定义时被绑定为 obj 了。又因为 fun2 的 this 指向的是 obj、箭头函数通过此法
用域依赖 fun2，所以才会有那个第二种说法说箭头函数的 this 会继承执行环境的执行上下文。

再来最后一段代码：<img alt="huaji" class="emoji" src="https://unpkg.com/
ditor/dist/images/emoji/huaji.gif" title="huaji">
<pre><code class="language-js highlight-chroma"><span clas
="highlight-cl">var obj</s
an> = {
  <span class="h
ghlight-c1">// 普通函数中定义一个立即执行函数输出 this
<span cla
s="highlight-c1">   fun1<span class="highligh
-o">: function ()</
pan> {
    <span class=
highlight-p">(function <span class="highlight
p">() {
      <span clas
="highlight-nx">console.<span class="highlight
nx">log(this</span
);
    <span class=
highlight-p">})();
  <span class="h
ghlight-p">},
  <span class="h
ghlight-c1">// 普通函数中定义一个立即执行箭头函数输出 this
  <span cla
s="highlight-c1">   fun2<span class="highligh
-o">: function ()</
pan> {
      <span class=
highlight-p">(() => <span class="highlight-p
>{
        <span clas
="highlight-nx">console.<span class="highlight
nx">log(this</span
);

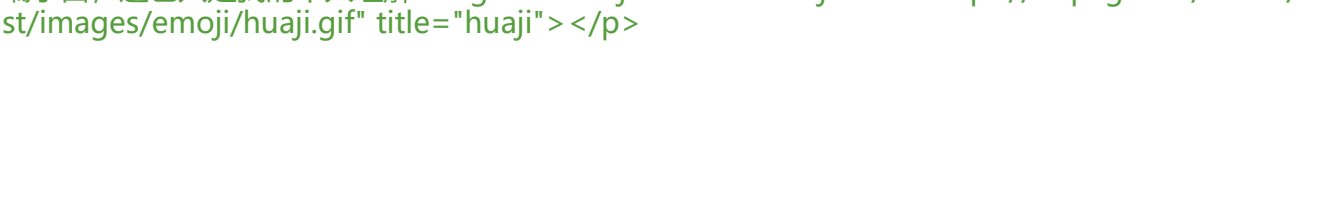
```
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-p">})();</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-p">},</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-c1">// 箭头函数中定义一个立即执行的普通函数输出 this
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span>    <span class="highlight-nx">fun3</span><span class="highligh
-o">:</span> <span class="highlight-p">()</span> <span class="highlight-p">=&gt;</spa
> <span class="highlight-p">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-p">(</span><span class="highlight-kd">function</span> <span class="highlight
p">()</span> <span class="highlight-p">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">            <span clas
="highlight-nx">console</span><span class="highlight-p">.</span><span class="highlight
nx">log</span><span class="highlight-p">(</span><span class="highlight-k">this</span
<span class="highlight-p">);</span>
</span></span><span class="highlight-line"><span class="highlight-cl">        <span class=
highlight-p">})();</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-p">},</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-c1">// 箭头函数中定义一个立即执行的箭头函数输出 this
</span></span></span><span class="highlight-line"><span class="highlight-cl">    <span cla
s="highlight-c1">//</span>    <span class="highlight-nx">fun4</span><span class="highligh
-o">:</span> <span class="highlight-p">()</span> <span class="highlight-p">=&gt;</spa
> <span class="highlight-p">{</span>
</span></span><span class="highlight-line">    <span class="highlight-cl">        <span class=
highlight-p">(()</span> <span class="highlight-p">=&gt;</span> <span class="highlight-p
>{</span>
</span></span><span class="highlight-line">    <span class="highlight-cl">            <span clas
="highlight-nx">console</span><span class="highlight-p">.</span><span class="highlight
nx">log</span><span class="highlight-p">(</span><span class="highlight-k">this</span
<span class="highlight-p">);</span>
</span></span><span class="highlight-line">    <span class="highlight-cl">        <span class=
highlight-p">})();</span>
</span></span><span class="highlight-line">    <span class="highlight-cl">    <span class="h
ghlight-p">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-p">}</span>
</span></span></code></pre>
<ul>
<li>正常通过 obj 调用的结果：</li>
</ul>
<pre><code class="language-js highlight-chroma"><span class="highlight-line"><span clas
="highlight-cl">console.log('正常执行----------');
</span></span><span class="highlight-line"><span class="highlight-cl">obj.fun1(); window
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1">//</span><span class="highlight-nx">obj</span><span class="highlight-p
```

>.fun2();<
pan class="highlight-err"> // obj
<span cla
s="highlight-c1">obj<span class="highlight-p
>.fun3();<
pan class="highlight-err"> // window
<span cla
s="highlight-c1">obj<span class="highlight-p
>.fun4();<
pan class="highlight-err"> // window
</code></pre>
<p><img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:
/b3logfile.com/file/2019/10/image-f43c433e.png?imageView2/2/interlace/1/format/jpg"></
>

obj.fun1()：内部立即执行函数因为直接调用，执行环境为 window 所以 this 是 window
obj.fun2()：内部的立即执行箭头函数因为定义时 this 根据词法作用域绑定执行上下文，因此箭
函数的作用域为 fun2，绑定 fun2 的执行上下文，this 绑定为 obj
obj.fun3()：可以不分析箭头函数，因为内部立即执行的普通函数直接调用，执行环境是 windo
，this 指向 window
obj.fun4()：分析步骤同 fun2 分析，内部的立即执行箭头函数根据词法作用域约束，其属于 obj.
un4 的箭头函数，要绑定 obj.fun4 所在的执行上下文，但因为 obj.fun4 也是一个箭头函数，所以也
样受词法作用域的约束，根据之前的示例，obj.fun4 的执行上下文指向的 window,因此内部的立即
行箭头函数也指向的是 window
通过 call 强行改变执行环境的结果：

<pre><code class="language-js highlight-chroma"><span clas
="highlight-cl">// 下面 this 是全局的 window
<span cla
s="highlight-c1">console<span class="highlig
t-p">.log(
span class="highlight-s1">'使用call执行----------');</sp
n>
<span class="high
ight-nx">obj.fun1<
span>.call<span cla
s="highlight-p">(this
; // window
<span cla
s="highlight-c1">obj<span class="highlight-p
>.fun2.<sp
n class="highlight-nx">call(<span class="highli
ht-k">this); // wind
w
<span cla
s="highlight-c1">obj<span class="highlight-p
>.fun3.<sp
n class="highlight-nx">call(<span class="highli
ht-k">this); // wind
w
<span cla
s="highlight-c1">obj<span class="highlight-p
>.fun4.<sp
n class="highlight-nx">call(<span class="highli

ht-k">this); // windw
</code></pre>
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:/b3logfile.com/file/2019/10/image-52056831.png?imageView2/2/interlace/1/format/jpg"><b>
从结果上来看，只有 fun2 的结果被改变了，其他的结果没有影响，根据上面总结的判断方法，应该以自行对除 fun2 的其他结果进行分析，那么现在回顾下 fun2 的代码
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https://b3ogfile.com/file/2019/10/image-525a8633.png?imageView2/2/interlace/1/format/jpg">
为什么箭头函数里的 this 发生了变化！前面提到过箭头函数一定定义后就会绑定 this，是无法通过 cal 和 apply 进行改变，为什么这里发生了变化？是不是这里比较特殊，不会遵循箭头函数 this 的指向则？
其实并不是，这里仍然遵循之前说的法则，正因为它遵守规则，所以输出的 this 发生了变化，只不过里绕了一个弯，因为这里是函数内部，这里的立即执行普通剪头函数在每次 fun2 调用时会重新进行次函数的定义，然后执行，这里 fun2 的代码等价于
<pre><code class="language-js highlight-chroma">fun2:</spa> function () {
  var testFun = () =></spa> {
    console.log(this<pan class="highlight-p">);
  }
  testFun();
}
</code></pre>
再次根据箭头函数 this 绑定的法则来看（箭头函数的 this 遵循词法作用域，指向其所属环境的行上下文（也可以说是宿主对象）。），每当 fun2 被调用时，会重新定义箭头函数，当前箭头函数词法作用域是 fun2，其指向 fun2 的执行上下文，正常情况是 obj，但我们通过 obj.fun2.call(this)行改变了 fun2 的执行上下文为 window，所以 fun2 的箭头函数重新定义时则指向了 fun2 的执行上文 window，也就是通过 call 的结果，所以这并不矛盾。
验证代码如下：
<pre><code class="language-js highlight-chroma">var obj</san> = {
  // 普通函数中定义一个立即执行箭头函数输出 this
> fun2: function ()</pan> {
    (() => <span class="highlight-p>{
      <span clas

="highlight-nx">console.<span class="highlight
nx">log(this</span
); // 正常调用 fun2 时，this
经给被绑定为 obj 了
<span cla
s="highlight-c1"> }).<span class="highlight
nx">call(window</
pan>); // 无法通过 call
行绑定 window
<span cla
s="highlight-c1"> },
<span class="high
ight-p">}
<span class="high
ight-nx">obj.fun2<
span>();
</code></pre>

/b3logfile.com/file/2019/10/image-549b0f08.png?imageView2/2/interlace/1/format/jpg"><b
>
结果并没有强行改变箭头函数的 this ，证明上面说法时正确的。
同理，假如将 fun2 里面的立即执行函数改成计时器 + 箭头函数的格式，那么每次也是调用 fun2 重生成计时器和箭头函数，箭头函数的内部 this 照样依据词法作用域绑定执行上下文。
<hr>
如果读者看到哪里有误或哪里说的模糊还请说明，我会及时学习更正的，我也只是一个刚入门的端小白，这也只是我的个人理解
dst/images/emoji/huaji.gif" title="huaji">

原文链接：对箭头函数的 this 深度理解，它究竟指向谁？