链滴

实现一个功能和 `Function.prototype.bind` 相同的函数

作者：Vanessa

原文链接：https://ld246.com/article/1546958569887

来源网站：链滴

2019-01-08
<h3 id="题目">题目</h3>
<pre><code class="language-mysql highlight-chroma">/**
  请按要求实现 `bind` 函数：以下代码执行时，需返回正确结果且运行过程中无常
*/
let
example = function () {
  console.log(this)
}
const boundExample = bind(example, { a: true })
boundExample.call({ b: true }) // { a: true }
</code></pre>
<h3 id="回答">回答</h3>

<code>bind</code> 需接受两个参数：函数和上下文
<code>bind</code> 需创建一个新函数，并且在调用时设置上下文为 <code>this</code>

<pre><code class="highlight-chroma">const bind = (fn, context) => () => fn.apply(context)

</code></pre>
<h3 id="加分回答">加分回答</h3>

如 <code>example</code> 函数需要传参的话，可如下进行实现：

<pre><code class="highlight-chroma"><span class="highlight
cl">const bind = (fn, context) => (...args) => fn.apply(context, args)
</code></pre>

<code>Function.prototype.bind()</code> 会创建一个新的函数，在调用时设置 <code>this
/code>。会返回一个原函数的拷贝，并拥有指定的 <code>this</code> 和初始参数。
<code>Function.prototype.bind()</code> 可解决 <code>setTimeout</code> 中 <code>
his</code> 为 window 的问题。还可使一个函数拥有预设初始参数的能力，如：

<pre><code class="highlight-chroma"><span class="highlight
cl">// setTimeout
class LateBloomer

    constructor() {
        this.petalCou
t = Math.ceil(Math.random() * 12) + 1;
    }
    declare() {
        console.log('I
am a beautiful flower with ' + this.petalCount + ' petals!');
    }
    bloom() {
        window.setT
meout(this.declare.bind(this), 1000);
    }
}
var flower = new
ateBloomer();
flower.bloom();
// 预设初始参数
const list = functi
n () {
    return Array.pro
otype.slice.call(arguments);
}
// 创建一个函数，
拥有预设参数列表。
const leadingThirt
sevenList = list.bind(null, 37);
console.log(leadi
gThirtysevenList()); // [37]
console.log(leadi
gThirtysevenList(1, 2, 3)); // [37, 1, 2, 3]
</code></pre>

<code>Function.prototype.apply()</code> 会调用一个具有给定 <code>this</code> 的
数，如果这个函数处于非严格模式且给定为 null 或 undefined 时，<code>this</code> 将指向全
对象，除此还提供一个数组作为该函数的参数。他会返回调用给定 <code>this</code> 值和参数
函数结果。

<code>Function.prototype.call()</code> 和 <code>apply()</code> 类似。唯一区别就是 code>call()</code> 方法接受的是参数列表，而 <code>apply()</code> 方法接受的是一个参数组。

<h3 id="返回总目录">返回总目录</h3>
每天 30 秒