·this的优点：提供了一种更优雅的方式来隐式‘传递’一个对象引用。因此可以将API设计的更加简洁并且易于复用。

this在任何情况下都不指向函数的词法作用域。

在JS内部，作用域确实和对象类似，可见的标识符都是它的属性。但是作用域无法通过JS代码访问，它存在于JS引擎内部。

this是在运行时绑定的，它的上下文取决于函数调用时的各种条件。

当一个函数被调用时，会创建一个活动记录。这个记录包含了函数在哪里被调用、函数的调用方法等信息。

this就是记录的其中一个属性，会在函数执行中用到。

如何查看一个函数的调用栈：可以用JS调试器工具给函数加断点，会展示当前位置的函数调用列表，这就是调用栈。

this的几个绑定：默认绑定、隐式绑定、显示绑定、new绑定

对象属性引用链中只有最顶层或者说最后一层会影响调用位置

例：function foo() {console.log( this.a );} var obj2 = {a: 42,foo: foo}; var obj1 = {a: 2, obj2: obj2}; obj1.obj2.foo(); // 42

this四个绑定的优先级：new绑定>显示绑定>隐式绑定>默认绑定。

如果把null或者undefined作为this的绑定对象传入call\apply\bind，这些值在调用时会被忽略，实际应用的是默认绑定规则。

另一个需要注意的是，你有可能（有意或者无意地）创建一个函数的“间接引用”，在这种情况下，调用这个函数会应用默认绑定规则。间接引用最容易在赋值时发生。

箭头函数并不是使用 function 关键字定义的，而是使用被称为“胖箭头”的操作符 => 定义的。箭头函数不使用 this 的四种标准规则，

而是根据外层（函数或者全局）作用域来决定 this 。

对象定义的两种形式：文字形式和构造形式。文字形式： var myObj={key:value;……};

typeof null//Object，实际上，null本身是基本类型。

对象的内容是由一些存储在特定命名位置的（任意类型的）值组成的，我们称之为属性。

属性描述符：

1.writable 决定是否可以修改属性的值。对于属性值的修改静默失败（silently failed）了。

2.Configurable：只要属性是可配置的，就可以使用 defineProperty(..) 方法来修改属性描述符。

即便属性是 configurable:false ， 我们还是可以把 writable 的状态由 true 改为 false ，但是无法由 false 改为 true 。

除了无法修改， configurable:false 还会禁止删除这个属性

3.用户定义的所有的普通属性默认都是 enumerable（可枚举）。