**简述call，callee，apply，bind四个方法的区别：**

**参考答案**

call,apply,bind都是函数对象的方法。

call方法：在调用函数时，更换函数执行上下文对象中的this引用的对象地址。

通常用于一个对象的方法，被另一个对象临时借用时使用。

call方法不但可以在调用函数时更换this引用的对象，而且，如果调用的函数如果需要参数，还可以同时传入参数值。

apply方法的作用和call方法完全相同。都是在调用函数时，临时更换执行上下文的中的this引用的对象。唯一的差别就是call方法在传入参数值时，要求每个参数值作为独立的参数传入，用逗号分隔。而apply则要求所有传入的参数值必须放在一个数组中，作为一个整体传入函数。

bind方法的作用是基于现有函数，创建一个新的函数副本。创建新函数副本的同时，为新函数副本绑定this引用的对象。绑定的结果是，新函数无论何时调用，this都引用绑定时指定的对象。属于持久绑定。

bind方法不但可以持久绑定一个对象到函数的this。而且还可以在调用前提前绑定个别参数的值。调用时，已经绑定的参数可不必传入。

bind方法和call以及apply的差别是：

显著差别是：call和apply是调用一个函数对象执行任务，而bind则是创建一个新的函数对象副本。bind方法不负责调用函数。

另外，call和apply方法，仅在调用时才能更换this引用的对象以及传入函数需要的参数值。一旦执行完就释放了。再调用时，还需要重新指定this引用的对象和传入参数。

而bind方法可将对象持久绑定在一个函数对象上。同时提前指定的参数值也可持久绑定在函数对象上。即使反复调用函数对象，this引用的对象和提前绑定的属性值依然与函数对象持久绑定。

bind方法基本实现原理模拟如下：

1. Function**.prototype.**bind **=** **function** **(**scope**)** **{**
2. **var** fn **=** **this;** // this 指向调用bind方法的一个函数对象
3. **return** **function** **()** **{**
4. //使用闭包将原函数对象fn，绑定给this的对象scope和参数对象arguments牢牢保护在新函数对象内。达到持久保存的目的。
5. **return** fn**.**apply**(**scope**,** arguments**);**
6. **}**
7. **};**

需要强调的是,因为闭包的原因，bind返回的函数对象中绑定的this对象的引用，无法用call或apply在调用时临时改变。bind返回的函数对象执行时的this，始终只会引用最初绑定的对象。

Callee是arguments对象的一个属性，引用了arguments所在的函数对象本身。

Callee属性和call以及apply方法的差别是，callee不会修改this的引用。Callee只是简单的引用当前arguments所在的函数对象而已。而call和apply反而不关心函数具体定义，而只负责修改调用函数时的this引用的对象。

Callee属性主要用于实现函数定义与外部的分离。比如实现匿名函数递归调用等操作。

总结：call和apply负责在调用函数时，临时更换函数执行上下文的this引用的对象。bind则专门用于基于现有函数定义，返回一个新函数对象，并提前绑定函数对象执行时的this引用。且是持久绑定，无法被call和apply替换。Callee是arguments的一个属性，与this无关。仅引用arguments所在的函数对象本身。