```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>jQuery的封装原理</title>
		<!-- 引入外部声明的js文件 -->
		<script src="js/my.js" type="text/javascript" charset="utf-8">
</script>
	<!--声明js代码域  -->
		<script type="text/javascript">
			function test(){
				alert("我是test");
			}
			var bjsxt=123;

			//闭包原理:在全局区不能够获取函数体内的数据,使用更大作用域的变量来记录小作用域变量的值
			function testA(){
			function test2(){
				var n=999;
				var a="haha";
				alert(bjsxt);
			}
			return test2;
			}
		</script>
	</head>
	<body>
		<h3>jQuery的封装原理</h3>
		<hr/ >
		<input type="button" name="" id="" value="测试"onclick="bjsxt.test()" />
		<ul>
			<li>1.js的全局代码区只有一个，这样就会造成同名变量的值会被覆盖。</li>
```

```
                        <li>2.使用对象封装，将代码封装到对象中。但是对象如果被覆盖，则全部失效，风险极高。</li>
                        <li>3.使用工厂模式，将代码进行封装，但是并没有解决问题</li>
                        <li>4.将封装的函数名字去除，避免覆盖。但是函数没有办法调用了</li>
                        <li>5.函数匿名自调用,可以在页面加载的时候调用一次。但是不能重复调用，并且数据没有办法获取</li>
                        <li>6.使用闭包,将数据一次性挂载到window对象下</li>
                </ul>
        </body>
</html>
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>jQuery的封装原理</title>
                <!-- 引入外部声明的js文件 -->
                <script src="js/my.js" type="text/javascript" charset="utf-8">
</script>
            <!--声明js代码域  -->
                <script type="text/javascript">
                        function test(){
                                alert("我是test");
                        }
                        var bjsxt=123;

                        //闭包原理:在全局区不能够获取函数体内的数据,使用更大作用域的变量来记录小作用域变量的值
                        function testA(){
                        function test2(){
                                var n=999;
                                var a="haha";
                                alert(bjsxt);
                        }
```

```
                return test2;
                }
        </script>
    </head>
    <body>
        <h3>jQuery的封装原理</h3>
        <hr/ >
        <input type="button" name="" id="" value="测
试"onclick="bjsxt.test()" />
        <ul>
            <li>1.js的全局代码区只有一个，这样就会造成同名变量的值会
被覆盖。</li>
            <li>2.使用对象封装，将代码封装到对象中。但是对象如果被覆
盖，则全部失效，风险极高。</li>
            <li>3.使用工厂模式，将代码进行封装，但是并没有解决问题
</li>
            <li>4.将封装的函数名字去除，避免覆盖。但是函数没有办法调
用了</li>
            <li>5.函数匿名自调用,可以在页面加载的时候调用一次。但是
不能重复调用，并且数据没有办法获取</li>
            <li>6.使用闭包,将数据一次性挂载到window对象下</li>
        </ul>
    </body>
</html>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>jQuery的封装原理</title>
        <!-- 引入外部声明的js文件 -->
        <script src="js/my.js" type="text/javascript" charset="utf-8">
</script>
    <!--声明js代码域  -->
        <script type="text/javascript">
            function test(){
```

```
                alert("我是test");
            }
            var bjsxt=123;

            //闭包原理:在全局区不能够获取函数体内的数据,使用更大作用域的变量来记录小作用域变量的值
            function testA(){
            function test2(){
                var n=999;
                var a="haha";
                alert(bjsxt);
            }
            return test2;
            }
        </script>
    </head>
    <body>
        <h3>jQuery的封装原理</h3>
        <hr/ >
        <input type="button" name="" id="" value="测试"onclick="bjsxt.test()" />
        <ul>
            <li>1.js的全局代码区只有一个，这样就会造成同名变量的值会被覆盖。</li>
            <li>2.使用对象封装，将代码封装到对象中。但是对象如果被覆盖，则全部失效，风险极高。</li>
            <li>3.使用工厂模式，将代码进行封装，但是并没有解决问题</li>
            <li>4.将封装的函数名字去除，避免覆盖。但是函数没有办法调用了</li>
            <li>5.函数匿名自调用,可以在页面加载的时候调用一次。但是不能重复调用，并且数据没有办法获取</li>
            <li>6.使用闭包,将数据一次性挂载到window对象下</li>
        </ul>
    </body>
```

</html>

my.js.note