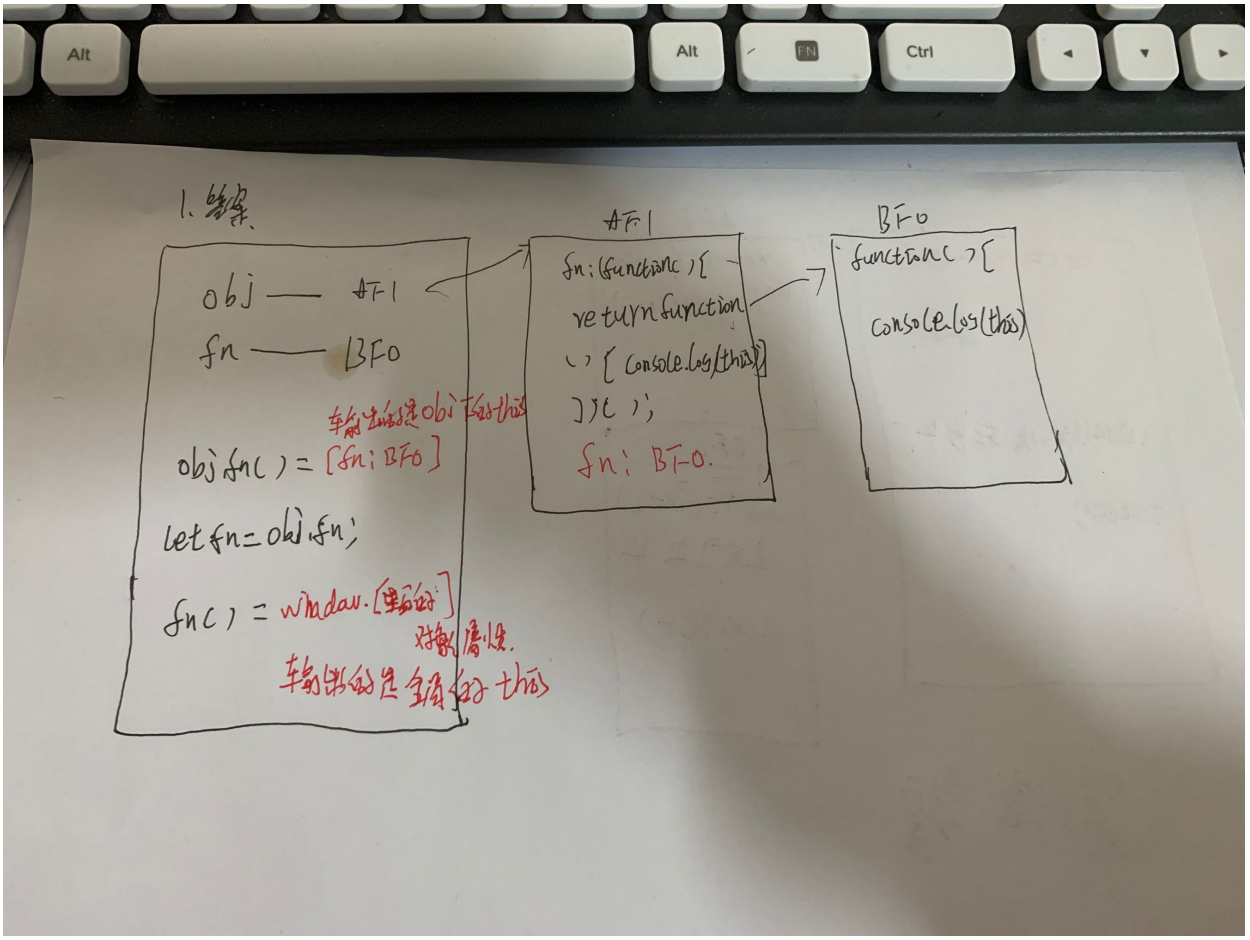
1.答案：



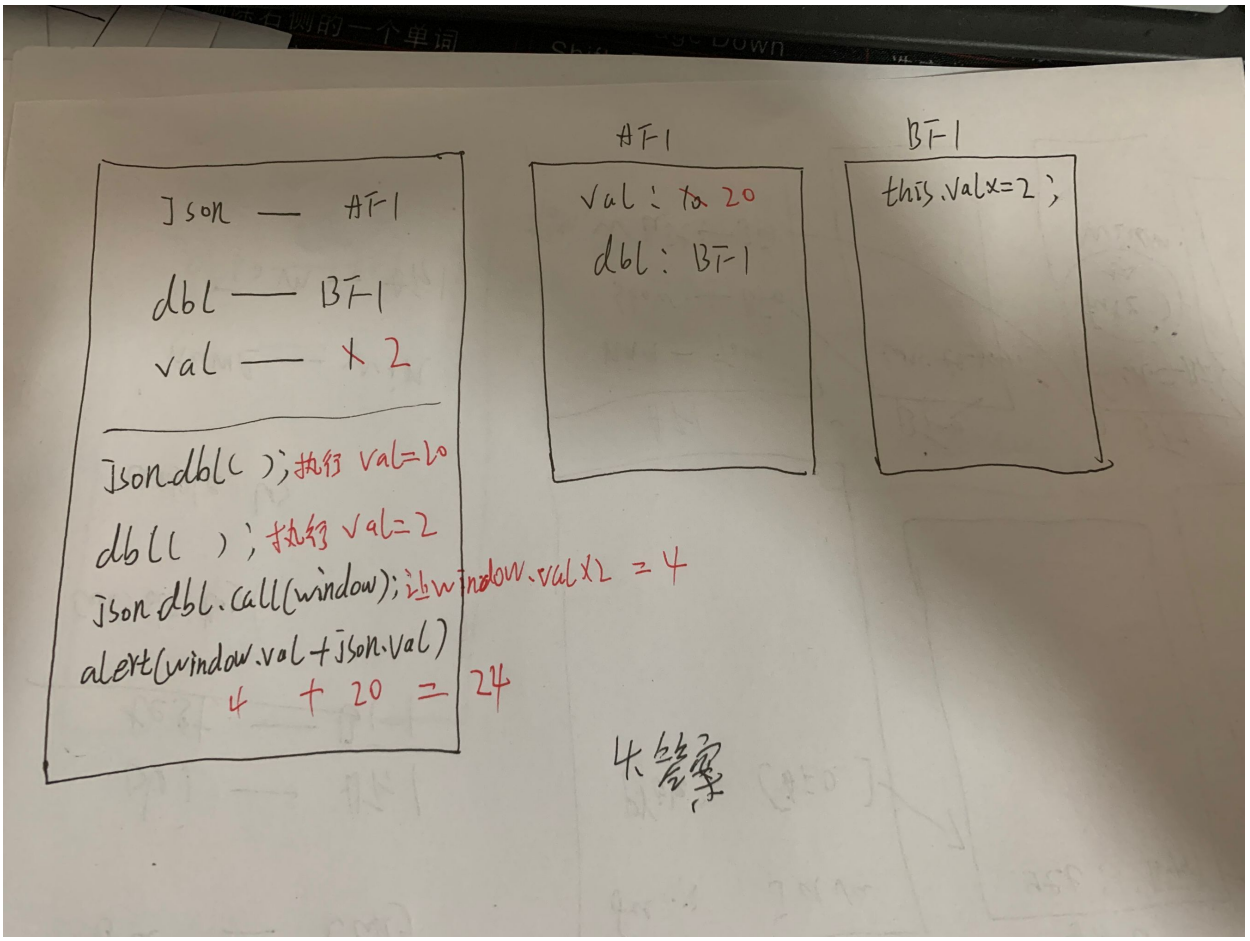
2.答案：

- undefineg
- {fullname:'javascript'}

3.答案：

- 'window'

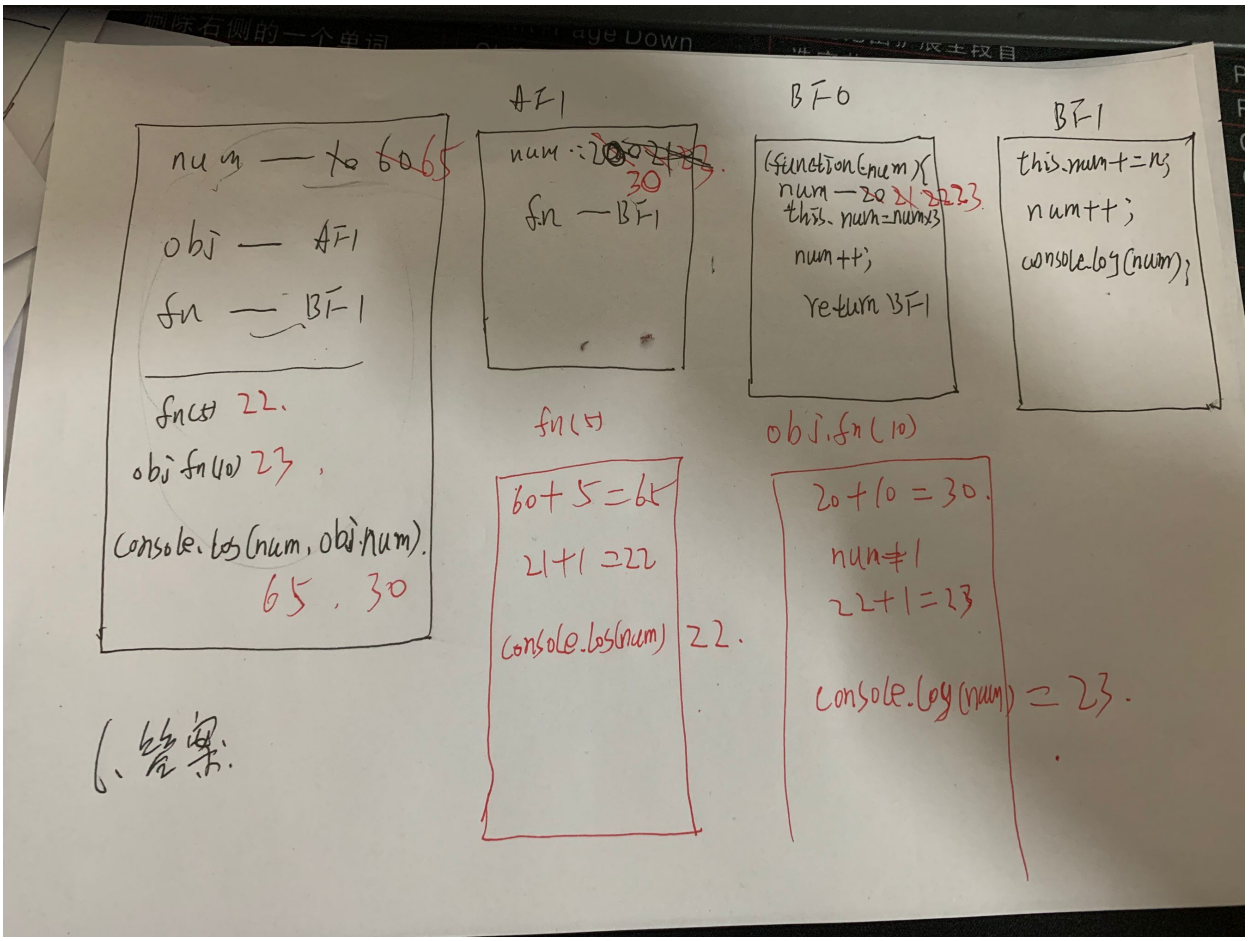
4.答案：


AF1
BF1
Json — AF1
dbl — BF1
val — x 2
val: 为 20
dbl: BF1
this.val x=2;
Json.dbl(); 执行 val=10
dbl(); 执行 val=2
Json.dbl.call(window); 让window.val x2 = 4
alert(window.val + Json.val)
4 + 20 = 24
大管家

5.答案：

- 21

6.答案：



7.答案:

不能。最后输出的结果i是5

给它加一个自执行函数就可以了

```
var btnBox = document.getElementById('btnBox'),
            inputs = btnBox.getElementsByTagName('input');
        var l = inputs.length;
        for (var i = 0; i < l; i++) {
            inputs[i].onclick = (function (i) {
               return function(){
                alert(i)
               }
            })(i)
         };
```

8.答案

优点:

1.团队开发中防止代码冲突
2.防止定义变量冲突
3.不会被浏览器自动清除掉
缺点：
因为闭包里面的函数变量被保存在内存中，内存消耗大，消耗性能。
9.答案：
var 存在变量提升 而let不存在
var允许重复声明，let不允许重复声明
var声明的变量既是给全局变量，也相当于给window 设置了一个属性。而let只是给全局变量声明了和go没关系。
10.答案：

```
for (var i = 0; i < 10; i++) {
            (function (i) {
                console.log(i)
            })(i)
        }
```

11.答案：
第一次输出的结果是 一个函数 function（b）{b=20;console.log(b)}
第二次输出的是10
因为执行自执行函数时创建了一个堆 里面包含函数b 里面的b=20这个b是在全局里面创建的一个属性名为b 值为20 而执行console.log (b)时首先是查找自己里面有没有b

没有了再去它自己的上一级作用域查找，而自己里面有一个函数b 所以输出的是函数 b

如果执行时给他var声明一下让这个b 创建时在自己的上下文中 然后在执行输出结果时就是20了。

```
var b = 10;
        (function b() {
            var b = 20;
            console.log(b);
        })();
        console.log(b);
```

给它var一下 输出结果就是 20 10了。