**一、变量提升的作业**

| | | | |
|---|---|---|---|
| **笔记本:** | 我的第一个笔记本 | | |
| **创建时间:** | 2021/3/15 18:09 | **更新时间:** | 2021/3/17 18:12 |
| **作者:** | 1851816627@qq.com | | |
| **标签:** | Web高级课第三天作业 | | |
| **URL:** | file:///E:/%E8%AF%BE%E7%A8%8B/00=%E5%9C%A8%E7%BA%BFWeb%E9%AB... | | |

# 一、变量提升的作业

- 1, undefined undefined undefined
  10 13 14
  100 13 200
  12 undefined 200
- 2 1 Object
- 3 f a(){}
- 4

# 二、闭包作用域的作业

- 1 10 11 3
- 2 3 10 undefined
- 3 5 5 6 2
- 4 “4”
- 5 9 10 10 1
- 6 11 6 13 10 6
- 7 undefined 0 1 1
- 8 简述你对闭包的理解，以及其优缺点
  - 闭包我在刚开始学习JS基础API的时候理解的是：大函数执行返回小函数，但是总感觉不是太通透，还总是忘记，于是就去深入研究了一下；
  - 函数执行形成一个私有的上下文，然后进栈执行执行完可能会出栈，它里面声明的变量都会存储在AO（私有变量对象中），形成作用域链，形参赋值，变量提升，以及代码执行，正常来说执行完都会出栈，但是由于浏览器的垃圾回收机制导致当前上下文中的东西被上下文以外的东西占用了，就不能被释放，这样就形成了闭包，所以说一方面私有变量不受外面干扰，这些变量也保存下来了，这也是闭包的两大作用【保存】和【保护】
  - 结合实战：详细说，在我后面的实战中也经常用到，比如我实现循环的时候想要把索引保存下来，

    - 其实我们基于闭包的思想，我们还能实现很多JS高阶编程技巧，比如说我们在项目中我们经常用到单例设计模式也是基于闭包来的，包括我们在做一些函数性能优化的时候用到的惰性思想，包括有时候我们会利用他的保存机制用柯理化函数思想去做一些事情，包括用组合函数实现一个执行起来的管道化
    - 比如说JS高阶编程技巧里面的单例设计模式，把描述相同的事物的属性和方法放置在同一个命名空间下，来实现分组特点，减少全局变量污染 =>单例设计模式；而基于闭包的方式，可以实现模块下部分方法的私有化，也可以基于单例实现API之间的共用 -最早期的模块化编程思想
- 9 简述let和var的区别?
    - var 有变量提升，let 没有，（另一种说法是）准确来说，let也有变量提升,只不过由于暂时性死区，不会被提升，
    - var 声明的变量会赋值给window,也就是全局对象GO中，let不会，let声明的变量会存储在VO(G)中;
    - var 可以重复声明，let 不可以；
    - let 声明的变量的作用域是块级的，而var没有
- 10 输出10 20 修改为以下代码输出 20 10

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

- 11
    - 方法一

```
    const fn = function fn(...params){
    return function anonymous(...args){
        params = params.concat(...args)
            return eval(params.join('+'));
        }
    }
let res = fn(1,2)(3);
console.log(res);
```

- 方法二

```
```

## 三、THIS的作业题

- 1 22 23 65 30
- 2 {fn:f()} Window
- 3 undefined language
- 4 Window
- 5 24
  -6 12