1、如何获取老师的讲义

- git clone https://github.com/zhufengpeixun/javascript201701.git
- git pull origin master 每天和老师的远程仓库保持同步更新

2、如何把老师的讲义放到自己的github仓库中

- 先git init（把本地文件夹初始化成一个仓库）会多一个.git隐藏文件夹
- 在自己的github仓库新建空的仓库（不能勾选readme）
- git remote add 自定义名字 自己仓库地址 让本地仓库和自己的远程仓库链接
- git remote -v 查看当前仓库和哪些远程仓库链接
  提交
- git add -A全部提交也可以是具体的文件名 暂存区
- git commit -m'本次提交完成哪个求' 历史区 生成版本号
- git push 链接名 master 推送到github 需要输入用户名和密码

如何忽略文件

新建.gitignore文本文件
把忽略掉的文件名字输入在.gitignore 就可以 比如.idea 等webstorm生成的文件

预解释

定义：在代码执行之前浏览器会先把所有带var 和function 的提前声明。
预解释发生在哪里

- 使用var 关键字声明的变量
- 使用function关键字声明的函数
- **区别：**var声明的变量只声明但是没有赋值，默认undefined。使用function声明的函数在预解释阶段就已经赋值结束。

私有变量和全局变量:

- 私有变量：定义在函数体内的变量
- 全局变量：定义在全局作用域（没有任何函数包含的变量）

私有作用域和全局作用域:

> - 私有作用域：只要函数运行，就会形成一个私有作用域
> - 全局作用域：只要打开一个网页就会形成全局作用域

堆内存和栈内存：

> - 堆内存：用来存储数据类型值占用的内存
> - 栈内存：代码运行时刻占用的内存是栈内存（也叫作用域：用来运行代码的作用域）

```
var a = [100,200];//数组占用的内存就是堆内存
function(){
    console.log(123);
}
fn();//函数执行时候产生的作用域也就是代码的运行环境就是栈内存

1.
console.log(a);
var a = [100, 200];
function fn(a) {
    //a 获取到的就是[100,200]这个数组的引用地址
    a[a.length] = 300;//操作的就是传进来引用地址所对应的的那个数组a=>[10
0,200,300]返回新数组a=[100,200,300]
    a = a.slice();//返回一个和原数组相同的数组（新数组）
    a[a.length] = 400 ;// [100,200,300,400]
    console.log(a);
    //        var a;
}
fn(a);//函数
console.log(a);//  [100,200,300]
```

函数的执行过程:

- 先形成一个私有作用域（栈内存）
- 形参赋值（形参相当于在函数体内部声明的私有变量）
  1)把实参代表的值赋值给形参一份；如果是引用数据类型，那么就把这个引用类型值得堆内存地址赋值给形参一份。
  2）形参相当于定义在函数体内的私有变量
  3）形参和私有变量重名，私有变量前面的 var 没用
- 预解释函数体内部的var 和function
- 代码执行

作用域链：

如果当前私有作用域内不存在这个变量，那么到上一级作用域去查找，如果上一级作用域也没有，那么还会继续向上查找。如果查找到顶级作用域window还没有，那么报错。
* ps : 如果当前作用域内存在私有变量那么就不会到其他作用域去查找了。

闭包：

函数执行就会形成一个私有作用域，而私有作用域保护私有变量不受外界干扰这种机制就叫闭包。

内存不释放：

内存释放 : 只要这个堆(栈)内存没有被占用就可以释放

堆内存释放

```
var obj = {
    name : 'zf',
    age : 8
};
//obj = null;
```

栈内存(函数执行)释放

> 函数执行的时候会形成一个私有作用域，并且定义在函数中的一部分(引用数据类型)被函数外的变量或者(对象的属性)所占用。那么就符合作用域不释放的条件。其实就是这个函数执行的时候产生的那个作用域不被释放。所以在这个作用域内定义的私有变量仍然会被保存下来。

```
 1) function sum(num){
    var num = 10;
    return function (){
        num++;
        console.log(num);
    }
}
var f = sum(); // 全局变量f占用
// 以上代码：  sum在执行的时候会形成一个作用域，定义在sum函数中的return后的匿名函数(引用数据类型)被sum函数外的全局变量f占用。那么就符合sum执行形成的这个作用域不被释放的条件。所以在这个作用域内定义的私有变量sum就会被保存下来。

2) function fn(){
    var num = 1;
    function fx(){
        num++;
        console.log(num);
    }
    return fx;
}
var f1 = fn();
f1(); // 2
f1(); // 3
var f2 = fn();
f2(); // 2
f1(); // 4
f2(); // 3
var f3 = f2;
f3(); // 4
f1 = f3;
f1(); // 5
```

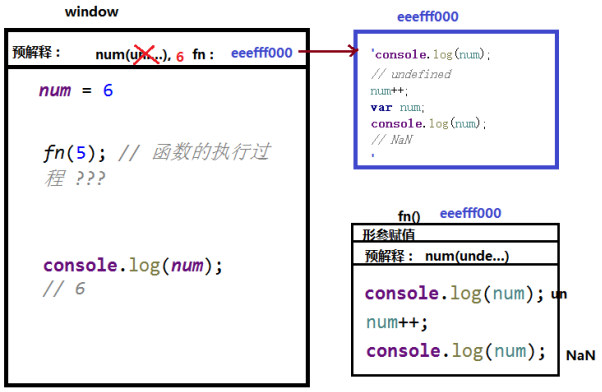
作用域不释放：

- 1 用全局变量或者对象的属性去占用一个函数执行返回的引用数据类型值 return
- 2 用一个dom对象的事件属性去占用一个定义在函数执行内部的引用数据类型值

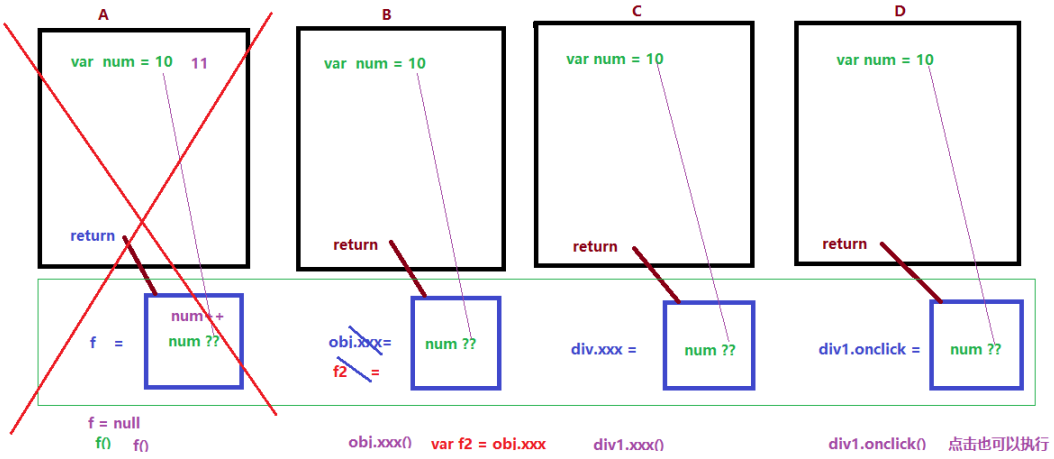
```
(function (){
document.body.onclick = function (){}
// 虽然没有return这个匿名函数，但是仍然符合占用关系
})()
```

- 3 暂时不释放 函数执行结束返回的函数立刻执行 fn()()

函数执行过程


window
预解释： num(un...), 6 fn： eeefff000
num = 6
fn(5); // 函数的执行过程 ???
console.log(num);
// 6
eeefff000
'console.log(num);
// undefined
num++;
var num;
console.log(num);
// NaN
'
fn() eeefff000
形参赋值
预解释： num(unde...)
console.log(num); un
num++;
console.log(num); NaN

作用域


A
var num = 10 11
return
f =
num++
num ??
f = null
f() f()
B
var num = 10
return
obi.xxx=
f2 =
num ??
obi.xxx() var f2 = obi.xxx
C
var num = 10
return
div.xxx =
num ??
div1.xxx()
D
var num = 10
return
div1.onclick =
num ??
div1.onclick() 点击也可以执行