微信公众号: 前端大全

书: 犀牛书

上届笔记

正课:

\*\*\*\*Function

1. 匿名函数

2. \*\*\*\*作用域和作用域链

3. \*\*\*\*\*闭包

\*\*\*OOP

封装，继承，多态

\*\*\*\*Function

1. 匿名函数:

什么是: 创建函数时，不指定函数名的函数

为什么: 节约内存，划分临时作用域

何时: 2种:

1. 如果一个函数，只使用一次时

2. 如果避免使用全局变量，划分临时作用域时

如何: 2种:

1. 回调: 将一个函数，交给另一个函数去调用

3种: 1. sort比较器函数

2. str.replace高级替换

3. 事件处理函数

2. 自调: 创建函数后，立刻调用自己

为什么: 全局变量最大的问题：全局污染

何时: 避免使用全局变量，造成全局污染时

今后，几乎所有的js程序，都应该放在匿名函数自调中

如何自调: 2种

(function (参数列表){ 函数体 }) ();

!function(参数列表){ 函数体 }();

2. \*\*\*\*作用域(scope): //练习10分钟，下课休息10分钟

什么是: 一个变量的可用范围——用途

一个存储变量的对象——存储/本质

为什么: 避免不同范围的变量间互相干扰

包括: 2种:

1. 全局作用域: 不属于任何函数的，外部的范围

其实: window

全局变量: 反复使用，随处可用！

2. 函数作用域: 函数内的范围

其实: 活动对象(AO)

局部变量: 仅函数内可用，不可反复使用！

函数的生命周期:

1. 开始执行程序前

创建执行环境栈(ECS): 保存正在调用的函数记录

首先自动调用浏览器主程序main()

主程序创建全局作用域对象window

2. 定义函数时:

在window中用函数名创建全局变量

window外创建函数对象，保存函数定义

函数名变量引用函数对象

函数对象的score属性指回函数来自的作用域对象

3. 调用函数时:

在ECS中压入当前函数的调用记录

为本次函数调用创建活动对象AO

AO中保存局部变量

让AO指向函数的scope属性指向的相同对象window

变量的使用顺序:

优先用AO中的局部变量，AO中没有，才去window找

4. 函数调用后:

函数调用的记录从ECS中出栈

导致: AO释放->导致: AO中的局部变量一同释放！

所以，局部变量仅在函数调用时可用，不可重用！

作用域链(scope chain): 由多级作用域连续引用形成的链式结构

存储: 所有变量

控制: 变量的使用顺序: 先局部，后全局

3. \*\*\*\*\*闭包(closure):

什么是: 即重用变量，又保护变量不被污染的一种机制

为什么:

全局变量: 优: 可重用！ 缺: 污染，随处可用!

局部变量: 优: 仅函数内可用，不会污染全局

缺: 不可重用！

何时: 只要即重用一个变量，又保护变量不被污染时

如何: 3步:

1. 用外层函数包裹要保护的变量和操作变量的函数

2. 外层函数将内层函数的对象返回到外部

3. 使用者调用外层函数，获得内层函数对象

闭包的形成原因: 外层函数的作用域对象(AO)无法释放

被内层函数对象引用着

简图: 2步:

1. 找受保护的变量，并确定其最终值

2. 找使用变量的内层函数对象，只有内层函数才能使用受保护的变量

闭包缺点: 比普通函数占用更多内存空间！

解决: 闭包不再使用，要及时释放

如何释放: 将引用内层函数对象的变量赋值为null

\*\*\*OOP

什么是面向对象: 程序中都是用对象结构描述现实中一个具体事物

什么是对象: 程序中专门描述现实中一个具体事物的程序结构

为什么: 现实中，一个数据，必须属于某个具体对象中，才有意义。

何时: 今后，只要描述一个事物，都要将事物的属性和功能集中定义在一个对象中

如何: 面向对象三大特点:

封装，继承，多态

封装:

什么是: 创建一个对象，存储一个事物的属性和功能

其中: 事物的属性，会成为对象的属性

属性其实就是存在对象中的变量

事物的功能，会成为对象的方法

方法其实就是存在对象中的函数

功能和属性，统称为成员

为什么: 便于大量维护数据

何时: 只要使用面向对象的方式编程，都必须先将事物的属性和功能封装在对象中，再按需使用对象的属性和方法。

如何: 3种:

1. 对象直接量:

var obj={

属性名:值,

... : ... ,

方法名 (){//ES6

...

}

}

如何访问对象的成员：

obj.属性名 -> 每个属性的用法和普通变量完全一样！

obj.方法名() -> 每个方法的用法和普通函数完全一样！