正课:

1. Function

\*\*\*\*作用域和作用域链

\*\*\*\*\*闭包

2. \*\*\*OOP

1. Function

\*\*\*\*作用域和作用域链

作用域(scope): 变量的可用范围

为什么: 避免内部的数据影响外部

包括: 2种: 其实所有作用都是内存中一个对象

全局作用域(window): 保存全局变量

全局变量: 优: 可重复使用! 随处可用

缺: 容易被篡改

函数作用域(活动对象AO): 保存局部变量

局部变量: 优: 不会被篡改

缺: 不可重用! 不可跨函数使用!

强调: vs java: 三级作用域: 全局, 局部, 块作用域

js中没有块作用域

函数生命周期:

1. 程序开始执行时:

浏览器创建执行环境栈ECS

ECS: 保存将来要调用的函数的数组

默认浏览器启动会先调用自己的主函数main()

main()为自己创建存储变量的空间window

2. 函数定义时:

创建函数对象

创建函数名变量引用函数对象

函数对象的scopes属性应用自己来自的作用域

3. 函数调用时:

在ECS中记录本次函数调用

为本次函数调用创建活动对象AO

在活动对象中保存函数定义中的所有局部变量

自动让活动对象AO的parent指向函数来自的作用域

变量的使用顺序:

先局部(活动对象)，再全局(window)

4. 函数调用后:

本次函数调用的记录从ECS数组中出栈(pop)

导致: 活动对象AO被释放

导致: 活动对象AO中的所有局部变量都释放

——局部变量不可重用!

作用域链scope chain:

什么是: 由多级作用域对象逐级引用形成的链式结构

保存着: 所有变量(全局和局部)

控制着: 变量的使用顺序

\*\*\*\*\*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: 全局变量和局部变量都有不可调和的优缺点.

何时: 只要希望即重用变量，又保护变量不被污染

如何: 3步:

1. 用外层函数包裹受保护的变量和内层函数

2. 外层函数将内层函数返回到外部

3种: 1. return

2. 直接给全局变量赋值

3. 将内层函数包裹在对象或数组内返回

3. 调用者调用外层函数，获得内层函数的对象

闭包如何形成: 外层函数的作用域对象无法释放!

鄙视: 画简图:

1. 先找受保护的变量,确定变量值

2. 找操作变量的内层函数

缺点: 1. 比普通函数占用更多的内存空间

2. 造成内存泄漏

解决: 用完闭包，立刻释放: 将引用内层函数的外部变量赋值为null

2. \*\*\*OOP:

什么是对象:

使用: 程序中描述现实中一个具体事物的程序结构

本质: 程序中同时存储多个数据和函数的存储空间

什么是面向对象: 程序中，都是先用对象描述现实中一个具体事物。再按需调用对象的功能或修改对象的属性。

为什么: 便于大量数据的维护和使用

何时: 今后几乎所有开发都用面向对象方式

如何: 三大特点: 封装，继承，多态

1. 封装: 用对象结构保存显示中一个具体事物的属性和功能:

事物的属性，会成为对象的属性

事物的功能，会成为对象中的函数(方法)

为什么: 便于维护

何时: 只要使用面向对象编程，都必须先封装对象，再按需访问对象的属性和功能

如何: 3种:

1. {} : var obj={

属性名:值,

... : ... ,

方法名(){//ES6可省略:function

... this.属性名 ...

}

}

问题: 对象自己的方法如何访问对象自己的属性:

错误1: 直接将属性名当变量用

原因: 函数中所有不加对象.就直接访问的变量只能在作用域链中查找。无法自动进入对象中。

错误2: 用对象名.属性名

原因: 对象名只是一个变量名，随时可能发生变化

正确: 用this.属性名

this:自动指向正在调用当前函数的.前的对象

今后，只要对象自己的方法，想访问自己的属性，都必须加this.前缀！

js中对象的本质: 对象底层就是关联数组

对象就是关联数组的简写方式

2.new: var obj=new Object();//{}

obj.属性名=值; //如果属性名是固定的

obj["属性名"]=值; //如果属性名需要动态获得(变量,表达式)

问题: 一次只能创建一个对象，如果反复创建多个相同结构的对象时，代码极其冗余，不便于维护

3. 用构造函数创建多个相同结构的对象：

什么是构造函数: 描述一类对象，统一结构的函数

何时: 只要反复创建同一类型相同结构的多个对象时，都要先定义构造函数。

如何定义:

function 类型名(属性参数列表){

this.属性名=属性参数;  
 this. ... = ... ;

this.方法名=function(){

... this.属性 ...

}

}

如何用构造函数反复创建多个相同结构的对象:

var obj=new 类型名(属性值列表);

如何访问对象的成员: 成员=属性+方法

obj.属性名 用法和普通变量完全一样!

obj.方法名 用法和普通函数完全一样!