javascript

js介绍运行在js解释器或者浏览器中

组成部分：ECMAScirpt,DOM(文档对象模型，允许操作页面元素).BOM（浏览器对象模型 允许操作浏览器）

js特点 无需编译，直接执行，弱类型 面向对象

语法

浏览器内核 

内容排版解析HTML/CSS

脚本引擎Javacript(v8)

独立js解释器 NodeJS

js脚本嵌入HTML页面中

HTML元素的事件（鼠标单击元素时要执行的操作）调用js

属性 onclick= console.log

alert()

<script>………</script>一般在head中，但是哪里都可以

使用外部脚本文件xxx.js HTML应用 <script src=”url”></script>

js调试

通用语法

；结尾 跟php一样

变量和常量

声明变量 var 变量名=值；+课拼接字符串

声明变量可以不用关键字 声明变量不赋值默认是undefind,可以连续声明多个变量

声明常量 const 名称=值；

变量.toFixed();保留小数

数据类型

原始类型

number类型

int

float/double

string类型x.charCodeAt();查找字符串的unicode码x.toString();转换成进制

\u数字转换成unicode的实体

转义与python一样

Boolean布尔型 布尔值跟python一样

undefined===NULL===none

隐式转换

数字转换为字符串 与php相反，其实就是存在字符串+当成拼接符号

读到字符串才会开始进行拼接

typeof(x)===type(x)==var\_dump(x)

强制转换

类函数

变量.toString()转为字符串

函数

parseInt(变量)转为int 转不成功结果为NaN 如果前面有数字字符会转化并把非数字字符抹除，如果第一个不是数字字符会直接NaN

parseFloat()变量; 结果与parseInt();一致

Number(变量);任意转换为number与parseInt();parseFloat();一致，但是全转换，有字符串会直接NaN

引用类型

运算符

跟php一样，比较运算符含有隐式转换，存在数字的情况下，+为拼接 和加号，但是.没有了

NaN与任何东西比较结果都为false包括自己isNaN(数据);判断是不是数字返回boolean

位运算符：<< >>

&按位与 判断奇偶性 与1按位与 结果1为奇数 0为偶数

|按位或 对小数取整 与0按位或 结果取整

^ 按位异或 交换两个数字

函数

跟php一样

function 函数名(){}

函数形参不用var声明..

作用域

函数作用域 局部变量

全局作用域 全局变量

声明提前

后面有声明的变量，前面调用了，会提前帮你声明为undefined

也就是说js会提前读取所有的var，function变量然后在原位赋值

案值传递

数据处理为副本处理

ECMAScript中的全局函数

parseInt()

parseFloat()

isNaN()

encodeURI() 编码，返回编码后的字符串，将多字节编成单字节

decodeURI() 解码encodeURI数据

encodeURIComponent() 可以多特殊字符进行encodeURI()编辑

decodeURIComponent()同上

eval()执行js代码

递归

分支结构

Math.random();生成0~1的小数

数组

声明 var name = [ ]; var name = new Array();

x.length = count(x) 数组中NULL === empty

关联数组 字典

var array = [ ];

array[“字符串”] = 值； 跟py的字典一样、

PHP和js列表和字典都可以混用，但是读取长度只读取列表中的值

便利关联数组

for（var x in 数组）{}: === py中的 for each in 数组：

数组函数

x.toString();类函数 数组值都转为字符串返回string

x.join(“”); 类函数 用什么拼接数组中的元素

x.concat(要拼接，要拼接的) 类函数 拼接2个或者跟多的数组，返回拼接后的结果

x.slice(start,[end])截取数组组成新数组 === [ : ] 取前不取后

x.splise(start, count, e1, e2…) 允许删除一部分元素，添加另一部分元素,start删除的起始下表 count 元素个数可以为0， e为要增加的新元素 返回删除的数组，会对原来的数组改变

x.reverse(); === py x.reverse();改变数组，将所有元素反转

x.sort() === py x.sort();改变数组，将数组排序 unicode码第一个排序 可以将函数传入括号中进行自定义

function x(a, b){

return a- b 升序

retrun b – a 降序

}

使用匿名函数

x.sort(function(a, b){return a-b});升序

x.sort(function(a, b){return b-a}); 降序

进出栈操作 数组

后进先出

x.push(n) 入栈 返回新数组长度 == x.append(n);

x.pop() 删栈顶数据，返回删除数据 ===py x.pop()

x.shift() 删除栈底部元素 返回删除数据

x.unshift(n) 在数组头部增加元素

书：javascript 权威指南

公众号：前端大全！

笔记和思维导图

string 字符串组成的数组 跟python的功能一样 字符串API全是对象所以源字符串不会改变

x.slice(a,b) 包头不包尾，字符串必须用’’

str.toLowCase()字符串转小写

str.toUpperCase()字符串转大写

str.charAt(i) == str[i]

str.charCodeAt() 字符转unicode 获取指定位置字符的unicode

String.fromCharCode(xxxxxx) Unicode转字符

str.substring(a, b) == str.slice(a, b)

str.substr(start, n) 从start开始选取那个字符

查找关键词

str.indexOf“关键词“，from) str中，从from开始找关键词，返回关键词第一个字符的位，没查找到返回-1

str.lastIndexOf()最后一个关键词的位置

str.trim

str.trimLeft/Right();去空格

正则表达式 描述一个字符串中字符出现的规律的规则 独立语法

固定关键词语法

[会出现的字符] 字符集：规定一位字符上可以用的备选字符列表

[123456789] == [0-9] [a-z] [A-Z] [A-Za-z] [\u4e00-\u9fa5]中间连续的可以用-代替

[^xxxx] 排除字符集

预定义字符集：

一位数字 \d

一位字母，数字 下划线\_ \w

一位空字符 \s 空格与tab与回车

通配符 .

两次 规定一个字符集出现次数的规则

有明确数量边界：{6，8}6-8 {6,}6以上 {6}必须6

没有明确数量边界： ？ 可有可无，最多一个

\* 可有可无，多了不限

+ 至少一个，多了不限

选择和分组：

选择： | 或者

分组：（规则）量词（规则）量词

转义字符：\

匹配特殊位置

开头 ^

结尾 $

匹配紧挨开头或者结尾的内容

单词边界 \b

js引入正则表达式

str.search(/正则表达式/) == str.indexOf(a)

str.search(//i)忽略大小写

str.match(//)获取所有关键词的内容 返回为数组 无内容返回null 正则默认只查找一个 null要进行if判断不然会报错

str.match(//g) global找全部位置

reg.exec() 关键字的位置加上内容

替换

str = str.replacd(//ig,”替换值”);将所有关键词，替换为统一内容

str= str.replace(/正则/ig, function(kw){});根据关键词动态替换内容

切割 将字符串切割成多段字符串

str.split(“分割的符号”);

str.split(“正则表达式”);

this 当前正在单击的按

链式表达

RegExp 专门保存一条正则表达式，并提供用正则表达式执行验证和查找操作的API的对象

使用正则表达式都要先创建RegExp对象

验证字符串格式

查找关键词内容和位置

创建 var reg=/正则表达式/ig .lastIndex记录下次查找的位置;

var reg=new RegExp(‘正则’,’ig’); 动态生成正则

API

var bool=reg.text(str); 只要存在正则就会通过 bug 所以前后要加边界

reg.exec(str); 在str中找下一个关键测的内容和位置，返回的数组：[0：关键词内容，index:本次的位置] 找不到返回null

math 保存算数的常量与提供API

Math.ceil(num);上取整

Math.floor(num);下取整 parseInt()

Math.round(num);四舍五入 .toFixex();按任意小数四舍五入返回字符串类型

Math.pow(底数，幂);乘方

Math.sqrt(n);开方

Math.max(n,a,b,c)最大值 Math.max(…nums);

Math.min();最小值

三角函数

Math.sin Math.cos Math.tan Math.asin Math.acos Math.atan Math.atan2

Math.random() 0~1的随机数

data 保存时间提供时间的API

var now = new Date();获得当前时间 无法获得服务器时间

var date = new date(‘1234-13-4 12:12;12’) 创建时间;

var date = new date(ms) 创建毫秒类型时间

time.getTime() 转换为毫秒

var date2 = new Date(date1);复制一个日期对象 日期对象会修改原对象

API :FullYear Month Date Day Hours Minutes Seconds Milliseconds 都有getXXX()获得数值，setxxx()设置数值 Day不能设置,内置日期进制算法

Moth:0~11

日期格式化

.toLcoaleString();

.toLcoaleDateString();

error

错误处理，即使发生错误也能正常运行

try{

可能发生错误的正确代码

}catch(err){

一旦发生错误，就自动执行的错误处理代码

} == try except err也为错误对象

错误类型

SyntaxError语法错误

RefenrenceError引用错误 没找到变量

RangeError 范围错误

TypeError 类型错误 错误使用了（）【】

URIError URI错误

EvalError Eval错误

抛出自定义错误

throw new Error(“xxxxxxxx”);

Function

创建 function 名字(){} 缺点 声明提前，函数与变量会在程序开头提前创建

var函数名 = function(){} 不会被声明提前

var fun = new Function(’参数’,……’函数’);

重载 相同函数名，不同参数列表的多个函数，调用时根据传入的参数不，自动选择匹配的函数执行。

js默认不支持重载

解决：argument 函数调用时，自动创建保存所有传入函数的参数值的类数组对象

每个函数默认都有一个argument[]

匿名函数 不指定函数名的函数 只使用一次的函数

回调函数

匿名函数自调 （functioin(){}）() 不会产生全局变量节约内存空间

作用域和作用域链 变量的可用范围

全局变量

函数作用域 局部变量

函数生命周期

程序开始前 创建执行函数环境栈（ECS） 记录有多少函数正在被调用

记录调用浏览器的主程序

创建全局作用域对象window 保存主程序执行所需要的全局变量

定义函数 用函数名定义变量

创建函数对象保存函数的定义

函数对象的scope属性，指回调用函数的作用域

调用函数 在ECS中加入本次函数调用记录

创建函数作用域对象AO

在AO中保存局部变量

AO的parent属性指向scope引用的作用域对象

AO中没有变量才去全局找

函数调用 函数出栈

释放闭包

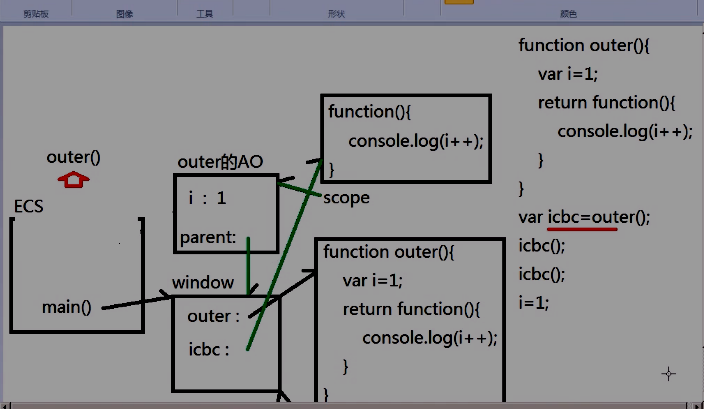
释放一根线，赋值为null

作用链 由多级作用域对象逐级形成的链式结构

闭包 即重用变量，又保护变量不被污染的一种机制

用外层函数包裹内层函数

外层函数return 内层函数对象



oop 面向对象 程序中描述现实中一个具体事物属性和功能的结构

三大特点 封装 继承 多态

创建

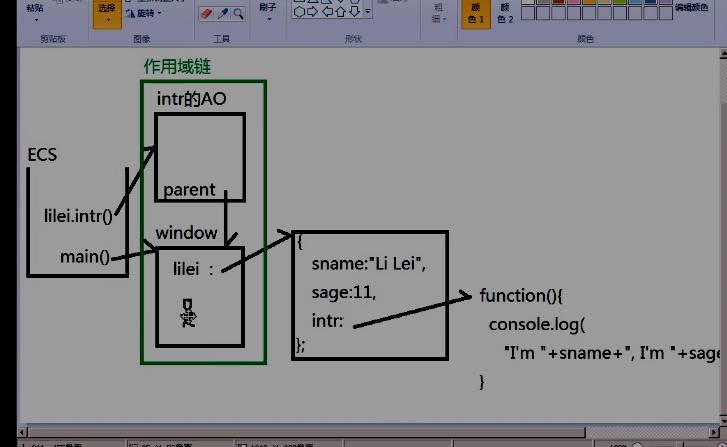
var obj = {

属性：值，

方法：function(){}

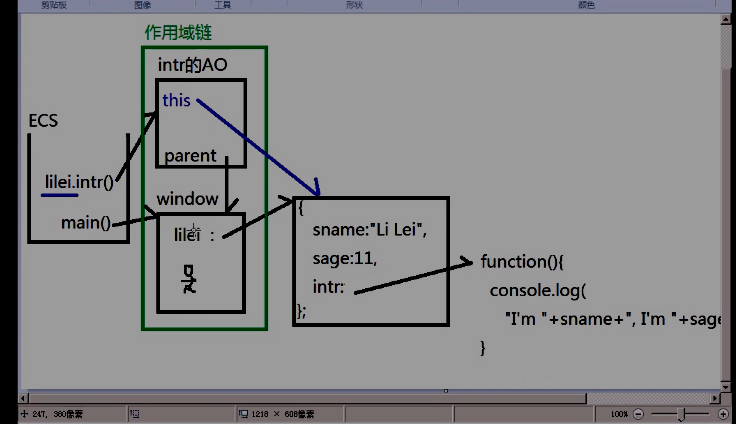
}

访问对象成员：obj.属性/方法



对象内查找对象属性

this.x == self.x this指向属性方法前的对象本身



var obj = new Object()

obj.属性名 = 值

obj.方法名 = function(){}

js对象底层就是关联数组

构造函数创建相同结构的对象

构造函数

定义构造函数，描述一类对象的统一结构

function 类型名（参数）{

this.属性名=属性参数；

this.方法名=function（）{}

}

调用构造函数反复创建统一类型的对象

var xx = new Student(传参);

在构造函数中添加的方法，每调用一次都会创建一个副本、

继承 js中都是继承原型对象prototype

添加父级属性 父级类.prototype.属性/方法 = xxx;

共有属性和自有属性

共有属性：保存在原对象中，所有子对象共有的属性

自有属性：仅子对象才有的属性

修改自有属性：子对象.属性=x

修改共有属性：对父级对象进行修改

内置对象的原型对象

js内置对象11个：string number Boolean – 包装类型

array date math regexp

function object

error

global (浏览器中被window代替)

内置对象浏览器已经定义好的

包装类型（box） 专门封装一个原始类型的值，并提供操作原始类型值的API

一中了类型包含构造函数：创建当前类型的子对象 原型对象：保存当前类型的所有子对象 解决浏览器兼容问题

继承

原型链：由多级父元素，，逐级继承，形成的链式结构

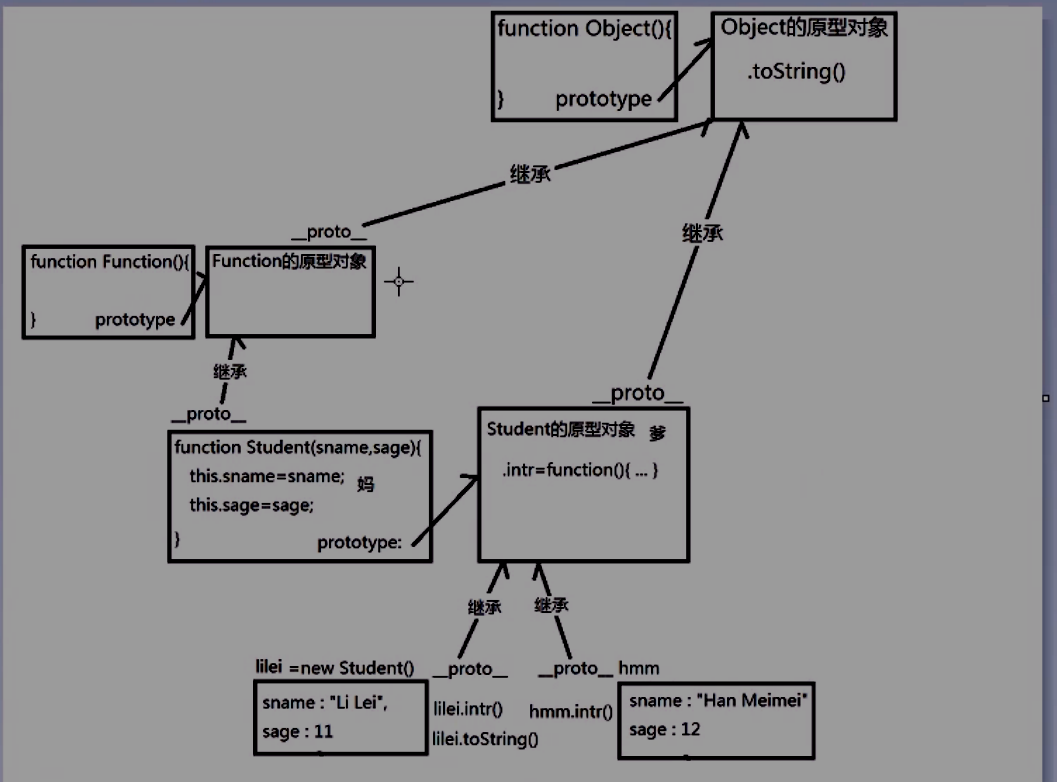
保存对象的属性和方法 控制对象属性方法的顺序与共享范围

先找自己再往上找

上层成员能共享给下层成员

作用域：保存了不加。就可以直接方位的变量

控制着变量的使用顺序，先全局后全部



判断原型对象

arr.\_\_proto\_\_ == Array.prototype;== Array.prototype.isProtypeOf(obj);

判断构造函数

obj.constructor ==Array == var bool = obj instanceof Array;

缺点：改子级\_\_proto\_\_就会失真

解决：检查class属性

每个对象创建都有class属性，而且不会随\_\_proto\_\_改变而改变，不能用,直接访问

解决：Object.prototype.toString();才可以输出

绕过多态的方法：call

想调用的函数.call(对象)

Object.prototype.toString.call(obj);

直接调用 Array.isArray(obj); ES5新增 构造函数自带

实例方法：父级定义。子级调用

静态方法：构造函数定义，构造函数自己调用

多态：同一个方法，在不同情况下表现不同的状态

重载overload

重写override 子对象觉得从父对象继承来的成员不好用时，可重写重名覆盖。

自定义继承：

修改一个对象的父对象 obj.\_\_proto\_\_ = father; === Object,setPrototypeOf(子，父);

修改多个子对象的父对象：在子元素创造前更换构造函数的prototype。 构造函数.prototype = father;

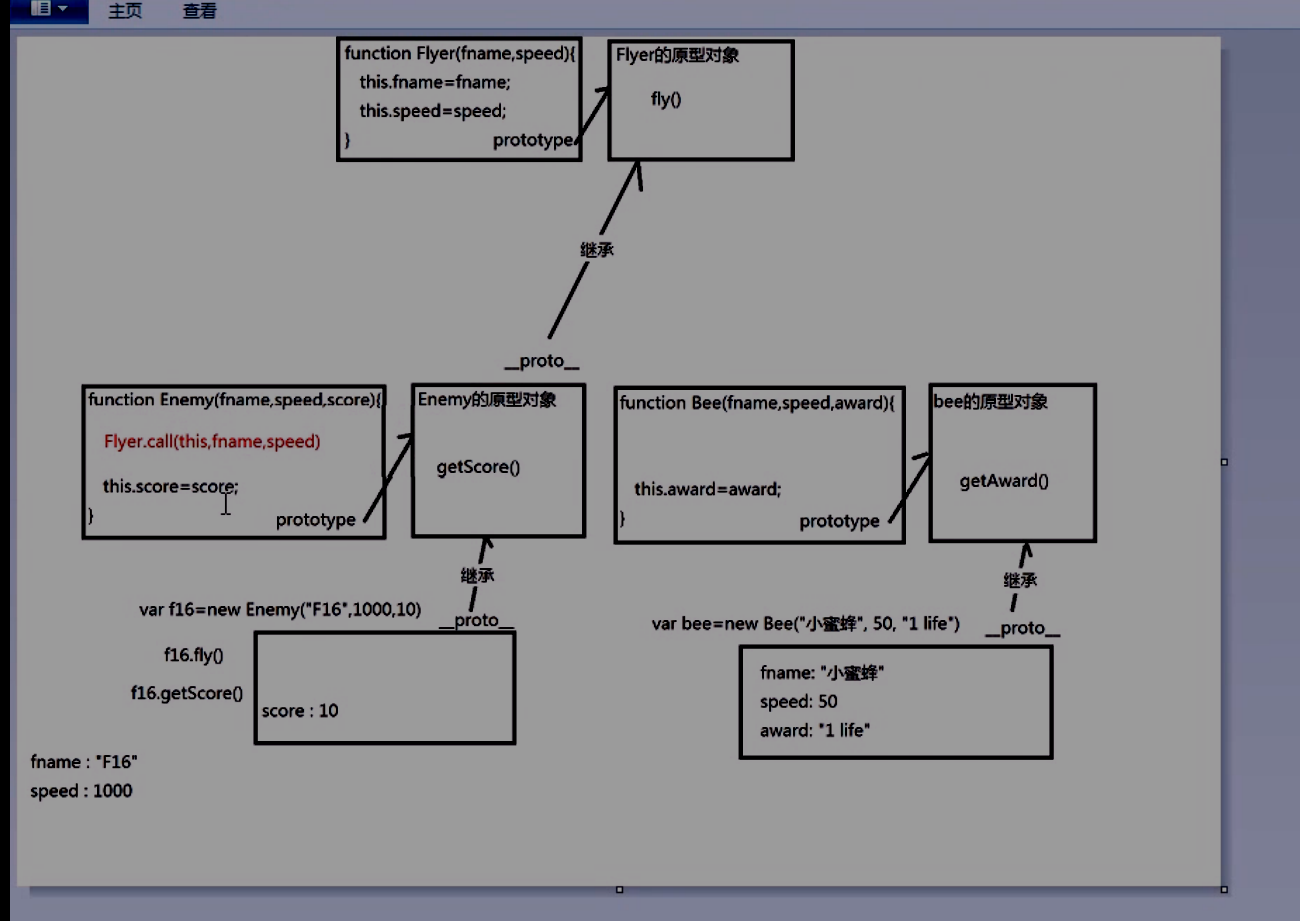
两种类型的继承：定义抽象父类型 集中储存多个类型间相同属性方法的类型

继承：在子类型构造函数中，先调用父类型构造函数

一个函数既没有用new调用和.调用，this就是指的全局变量

call：替换函数内不想用的this

设置子类型原型对象继承父类型原型对象



ES5

严格模式：比普通js运行更严格，所有代码开发都必须使用

1:、<script>”use strict”</script>script开头

2、 在一个函数内启用 函数内的顶部加：“use strict”

规定：

禁止给未声明的变量赋值

匿名函数自调和普通函数的this不再默认指向window。而是undefined

禁止用arguments.callee.在函数调用时，在函数内部禁止引用当前函数本身

将静默失败升级为错误！

保护对象：

保护属性

2大类 命令属性：用.直接访问的属性 服务器端语言共有public

保护命名属性：

数据属性：实际保存属性值的属性

四大特性：value:储存值

writable:true 控制是否可修改

enumerable:true 控制是否可被for in 便历到 可用.访问

configurable:true 控制是否可删除属性，能否修改其他特性 改为false不可逆

获得四大特性：

var attrs = Object.getOwnPropertyDescriptor(obj，属性名);

修改四大特性：

Object.defineProperty(obj,’属性名’, {特姓名：值。。。}) 每次只能修改一个

Object.defineProperties(obj, {

属性名：{特性名：值，。。。}，

属性名：{特性名：值，…}

});

问题：保护固定

解决：访问器属性：不实际存储属性值，仅对其他数据属性提供保护

自定义规则保护属性

计算属性/虚拟属性

如何：四大特性：{

get:function(){return this.受保护的数据属性}

set(value){value 自动收到要赋的新值 if(规则){this.属性 = value}else 报错}

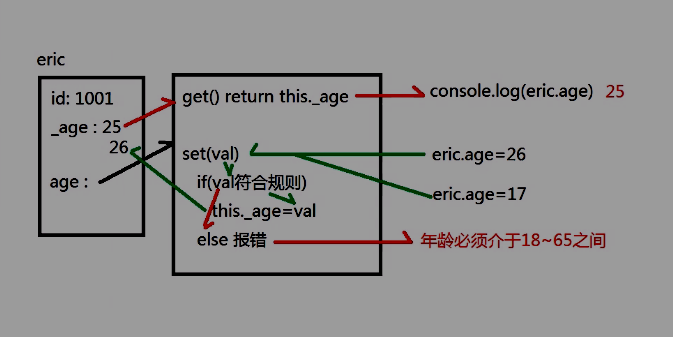
enumerable:true/false,

configurable:true/false

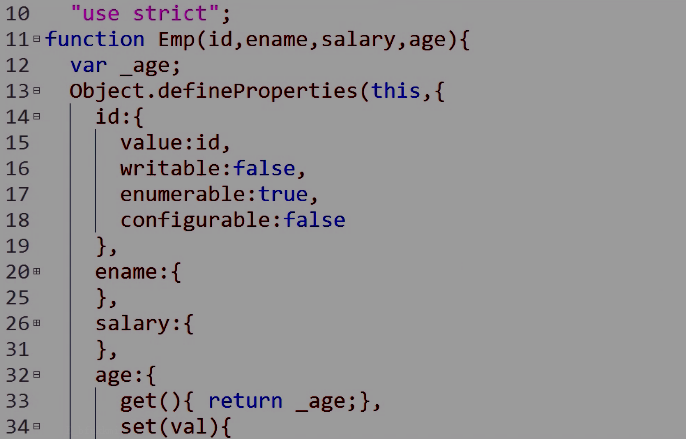
}

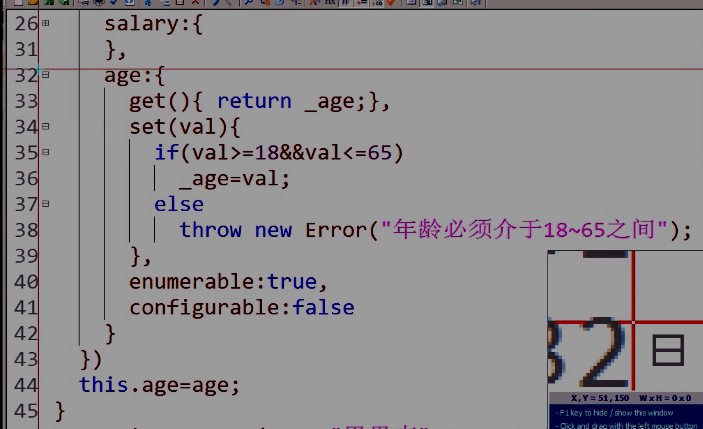
定义访问器属性：

只能通过Object.defineProperty或Object.defineProperties创建



构造函数保护实例





内部属性：不能用.访问 私有private

防纂改：

1. 防扩展 禁止给对象添加新属性

Object.preventExtensions(this);

1. 密封 兼具方扩展基础上禁止删除现有属性

Object.seal(obj) 实现上方并且每个属性的configuarble为false

1. 在密封上禁止修改一切属性

Object.freeze(obj) writable:false

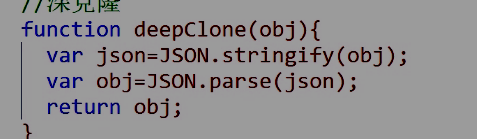
Object.create 创建一个子对象，继承指定父对象，并为子对象扩展自有属性

var child = Object.create(father, {与Object.defineProperties一样语法});

Object.assign() 赋值或合并对象

var clone = Object.assign({}, obj,…) 将{} 与obj,…合并返回给clone

浅克隆：引用类型赋值地址

深克隆：

var clone = JSON.parse(JSON.stringify(obj));

call apply bind: 如果函数中的this不是想要的，可以用这三替换函数中的this

call apply:在调用函数时，临时替换函数中的this

fun.call(obj,参数值列表)

fun.apply(obj,[参数数组]);

bind:基于原函数，创建一个新函数，并永久绑定其中的this为指定对象

数组API

判断数组中元素是否符合条件

.every(): 是否每个元素符合要求 bool = arr.every(function(elem, I, arr){return}) 每个都符合返回true

.some(): 判断数组中是否包含符合要求的元素

遍历API 对数组中每个元素执行相同的操作

.forEach() 对原素组中每个元素执行操作，改变原素组 arr.forEach(function(elem, I, arr){});

.map() 每个元素执行操作返回新数组var arr = arr.map(function(elem, I, arr){return})

过滤 选出原素组中符合条件的元素，组成新素组返回，原素组不变

var subArr = arr.filter(function(elem, I, arr){})

汇总 讲数组中每个元素的值汇总统计出一个最终的接龙

varresult = arr.reduce(function(prev, elem, I, arr){

prev:截止当前位置之前的临时汇总值

}， base（prev的基数）)

ES6

let ：代替var let不会声明提前! 不允许重复声明变量 添加添加块级作用域（一个{}内也是一个作用域）===匿名函数自调 闭包（当let和for的时候回形成闭包）

let会自动形成匿名函数自调 与for一起用时候会形成传参的匿名函数自调（function

（i）{}

）(i)

参数增强

default:在定义函数时。为函数参数添加默认值跟python一样，可以给参数设置默认传参，但是默认参数必须放后面

rest剩余参数：代替arguments：不是数组类型，无法使用数组API,只能获得全部参数

不确定参数个数时，用rest:固定参数之后 …bonus(这个是数组)

spread：讲素组打散分别传入： …数组

箭头函数：省略function 所有匿名函数和回调函数都可以用箭头函数来简写

去function改=> function(){} === ()=>{}

如果只有一个参数可以省略（）

如果只有一句话可省略{}

如果只有一句话还只剩return，可以省略return

箭头函数内外共用同一个this

模板字符串 简化复杂字符串拼接 代替+号拼接

``包裹 想要执行语句放在${}中

解构：简化批量赋值：从一个大的对象中，挑选部分属性值使用时

数组解构：下标对下标

var [变量1，变量2] = [值1，值2]

对象解构：属性对属性

var{想要的属性1：属性1，想要的属性2：属性2} = {属性1：变量1，属性2：变量2 }

参数解构：0 简化普通for循环

for(var 变量 of 数组、类){

变量会自动获得数组、类数组对象中的每个元素

}

class

封装：class 类型名{ constructor(){} 方法 } == python

继承：class father{constructor(){}} class child extends father{constructor(){super(属性名)}}

静态：static不需要创建子对象就可以用构造函数直接调用

访问器属性：get 属性名（）{}

promise 解决回调地狱

function xx(){return new Promise(fn){fn()}}

用.then(回调函数);

promise 的错误处理

new Promise（正常， 异常）=>{

正常 调用正常（）

出现错误 调用错误（）

}

xxx().then()….catch();

让多个一起运行然后等所有任务完成再干其他

Promise.all([x1(), x2()…]).then() => { 之后的任务 })

video/audio对象属性

currentTime 当前播放时长

duration 总时长

paused 当前视频是否处于暂停状态

volume（0~1）音量

playbackRate 回放速率 1正常

方法

play()

pause()

事件

onplay

onpause