正课:

1. OOP

2. ES5

1. OOP

实例方法 vs 静态方法:

实例方法: 必须用一个具体的子对象，才能调用的方法

何时: 绝大多数方法，都是实例方法

比如: arr.sort() str.replace() reg.test()

存储: 当前类型的原型对象中

限制: 只有当前类型的子对象才能访问

静态(static)方法: 不需要子对象，直接用构造函数就可直接调用的方法

何时: 如果无法提前预知使用该方法的对象类型时

比如: String.fromCharCode(unicode)

Array.isArray(obj)

存储: 直接定义在构造函数对象上

强调: 不是构造函数{}内

限制: 没有限制!

自定义继承:

1. 仅修改一个对象的父对象

2. 修改所有子对象的父对象

3. 两种类型间的继承:

何时: 只要发现两种类型间有部分相同的属性和方法

如何: 定义抽象父类型

2. ES5

严格模式:

什么是: 比普通的js运行机制，要求更严格的模式

为什么: js语言本身有广受诟病的缺陷

何时: 今后所有js，都要运行在严格模式下

如何: 2种:

1. 整个<script>或js文件，启用严格模式:

在<script>或js文件顶部，添加"use strict";

何时: 新项目,都要整文件启用严格模式

2. 仅对一个函数内，启用严格模式:

在function内，第一行，添加"use strict";

何时: 旧项目，向严格模式迁移

包括:

1. 禁止给未声明的变量赋值:

2. 静默失败升级为错误

3. 函数中的this不再默认指向window

默认值为undefined

4. 不再允许使用arguments.callee或caller

arguments.callee 指代当前函数自己

几乎专门做递归:

今后尽量不要使用递归！

保护对象:

问题: js中的对象，可以随意修改属性值，添加新属性，删除现有属性

解决: 保护对象的属性和结构

如何: ES5将对象的属性分为2大类，三小类

命名属性: 可用.随意访问的属性

数据属性: 直接存储属性值的属性

访问器属性: 不直接存储属性值，而是对另一个数据属性提供保护

内部属性: 不能用.随意访问的属性

比如: .class

保护命名属性:

保护数据属性:

每个数据属性都有四大特性:

{

value: 实际存储属性值,

/\*三个开关\*/

writable: 控制是否可修改,

enumerable: 控制是否可被for in遍历

强调: 只控制for in遍历，不控制.直接访问属性

configurable: 控制2个:

1. 控制是否可删除当前属性

2. 控制是否可修改另外两个开关

强调: 一旦改为false，不可逆

所以，只要修改前两个开关，必须带上configurable为false

}

查看一个属性的四大特性:

var id\_prop=Object.getOwnPropertyDescriptor(obj,"属性")

修改一个属性的四大特性:

Object.defineProperty(obj,"属性",{

特性名: 值,

... : ...

})

修改多个属性的四大特性:

Object.defineProperties(obj,{

属性名:{

特性:值,

... : ...

},

属性名:{

}

})

问题: 4种特性无法使用自定义规则灵活保护属性

解决: 访问器属性:

使用访问器属性保护数据属性:

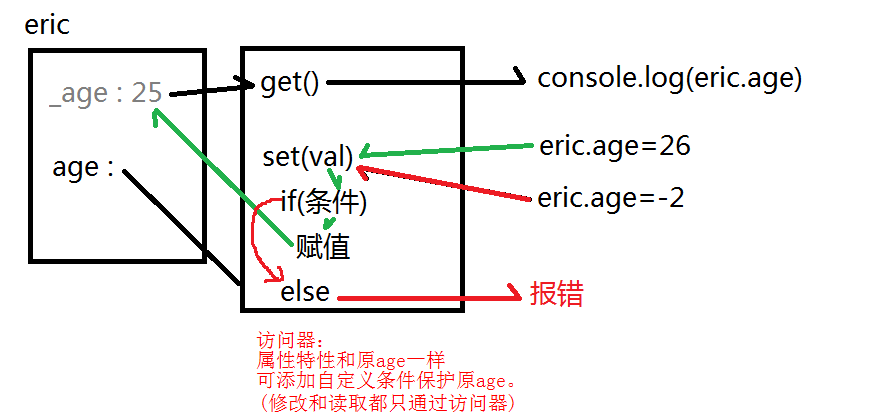
何时: 只要用自定义规则保护数据属性时

如何: 2步:

1. 定义一个隐藏的受保护的数据属性:

2. 定义访问器属性的四大特性:

**总结规律1.get：外界要拿，通过访问器拿；set：外界要存，要通过访问器存；**



{

get:function(){ return this.数据属性 },

set:function(val){

if(val满足要求)

this.数据属性=val;

else

报错!

},

enumerable:

configurable:

}

强调: 访问器属性不能通过直接量添加

必须通过Object.defineProperty()来添加

使用访问器属性: 访问器属性用法和普通数据属性用法安全一样

原理: 试图获取访问器属性的值时，自动调用get方法

试图修改访问器属性的值时，自动调用set方法，并将新值传递给参数val

问题: js的对象依然可以随意添加新属性

解决: 保护对象的结构:

保护对象的结构: 3个级别:

1. 防扩展: 禁止添加新属性

Object.preventExtensions(obj)

原理: 其实每个对象内都有一个内部的extensible开关

默认打开，所以可随意添加新属性

preventExtensions可关闭这个开关

2. 密封: 兼具防扩展的同时，禁止删除现有属性

Object.seal(obj)

原理: 设置extensible开关为false——防扩展

自动设置所有属性的configurable为false

3. 冻结: 兼具密封的同时，禁止修改一切属性值

Object.freeze(obj)

原理: 设置extensible开关为false——防扩展

自动设置所有属性的configurable为false

自动修改所有属性的writable: false

Object.create()

单靠父对象，就能创建子对象

何时: 没有构造函数，也想创建子对象时

如何: var child=Object.create(father,{

//同defineProperties

})