正课：

1. 面向对象

2. ES5

1. 面向对象:

继承:

原型链:

什么是: 由多级父对象，逐级继承，形成的链式结果。

作用:

1. 保存了一个对象可用的所有成员

将来判断一个对象可以使用哪些成员，就看它的原型链上有什么成员

如果一个成员在子对象的原型链的某个父级对象上，就可以: 子对象.成员 访问

如果一个成员不包含在子对象的原型链的任意一个对象上，就不能用 ~~子对象.成员~~访问

2. 控制着成员的使用顺序:

先自有，再共有！

多态:

什么是: 一个函数，在不同情况下表现出不同的状态

包括:

1. 重载:

2. 重写:

什么是: 在子对象中定义和父对象中成员完全相同的成员。来覆盖父对象中的成员。

为什么: 因为从爹继承来的成员不一定都是好用！

何时: 只要子对象觉得从爹继承来的成员不好用，就要重写！

如何: 只要在子对象中定义和父对象同名的成员

自定义继承:

何时: 只要觉得现在的爹不好用！就可以换。

如何:

1. 只更换一个自对象的爹:

子对象.\_ \_proto\_ \_=新爹

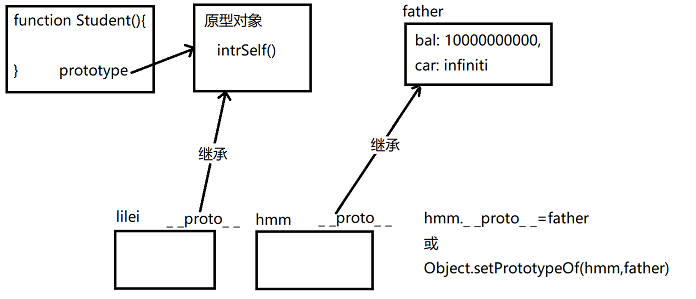
问题: \_ \_proto\_ \_浏览器不推荐使用

解决: setPrototypeOf()代替\_\_proto\_\_

Object.setPrototypeOf(子对象, 新爹)

set of Prototype

设置 子对象 的 原型对象 为 新爹



2. 同时换多个对象的爹:

构造函数.prototype=新老公

时机: 在定义完构造函数后，立刻更换！

2. ES5

什么是: ECMAScript标准的第五个升级版本

为什么: 因为js这门语言，有很多广受诟病的缺陷

包括哪些新的规定:

1. 严格模式:

什么是: 比普通js运行要求更严格的机制

为什么: 因为js这门语言，有很多广受诟病的缺陷

何时: 今后所有js程序都要运行在严格模式下！

如何: 在这段程序的顶部，写 "use strict";

严格模式的新要求:

1. 禁止给未声明的变量赋值：

旧js中: 强行给未声明的变量赋值

结果: 会自动在全局创建

全局污染，内存泄漏

严格模式中: 强行给未声明的变量赋值

结果: 报错: xxx is not defined!

避免内存泄漏和全局污染！

2. 静默失败升级为错误

静默失败: 执行不成功，也不报错！

极其不便于调试

严格模式中: 静默失败，也会报错！

便于调试！

3. 普通函数调用和匿名函数自调中的this不再默认指向window。而是指向undefined

防止内存泄漏和全局污染。

4. 禁止使用arguments.callee

什么是arguments.callee: 是在函数运行时，专门获得当前正在执行的函数本身。

何时: 递归！

为什么: 避免紧耦合，避免在函数内写死函数名。

问题: 递归的效率极低——重复计算量太大！

所以: ES5严格模式，出于性能考虑，禁用了arguments.callee，就暗示着不推荐使用递归算法。

解决: 绝大多数递归，都可用循环代替

2. 保护对象:

什么是: 控制对对象的属性值或对象结构进行的随意的篡改。

为什么: 在旧的js中，对象的属性和结构毫无自保能力。任何人可在任何时候修改属性值为任何值，也可以随意添加属性和删除属性。

何时: 需要控制对对象的访问时

如何: 2个层面:

1. 保护对象的属性

ES5对对象的属性进行了重新的分类:

命名属性: 可用.访问到的属性

又分为:

数据属性: 实际存储属性值的属性。

访问器属性: 不实际存储属性值，仅提供对另一个数据属性的保护。

内部属性: 无法用.访问到的属性

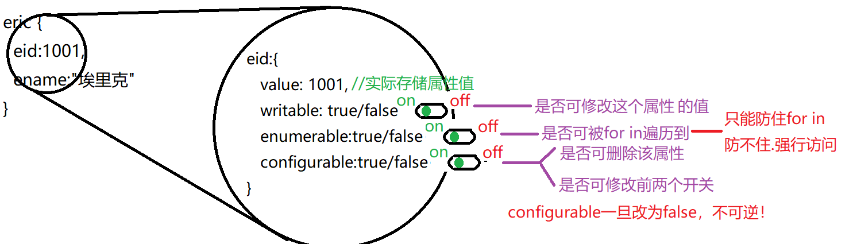
如何保护数据属性:

旧js中: 数据属性其实就是一个普通的变量。毫无自保能力。

ES5标准中: 每个数据属性都是一个缩微的小对象。每个小对象中，保存一个属性值和三个开关！

比如: eric={ eid: 1001 }

eid不再是一个变量而是:



如何修改属性小对象内的开关属性来保护一个属性呢？

问题: 不能用.直接修改属性对象内部的开关属性！

解决: 必须用专门的函数;

Object.defineProperty(

对象名,

"属性名",

{

开关: true/false

}

)

问题: writable和enumerable两个开关，任何人都可以随意开关。所以，仅设置这两个属性，起不到保护的作用。

解决: 只要设置writable和enumerable，都要同时关闭configurable，目的是禁止修改前两个开关。且configurable一旦被改为false，不可逆！

问题: enumerable只能禁止for in遍历这个属性。无法阻止用.直接访问这个属性。——只是半隐藏

解决: 无法解决

问题: Object.defineProperty()一次只能修改对象中的一个属性。如果修改多个属性，就要重复写多遍。

解决: Object.defineProperties(

对象名,

{

属性名:{

开关:true/false

},

属性名:{

开关:true/false

},

}

)

问题: 无法使用自定义规则保护属性

解决:使用访问器属性保护数据属性

什么是访问器属性: 自己不保存数据，专门提供对其它数据属性的保护。

何时: 只要使用灵活的自定义规则保护数据属性时，就要请保镖！

如何:

1. 先将要保护的数据属性，隐形埋名，并且半隐藏！

2. 为对象添加访问器属性，保护数据属性

访问器属性要冒名顶替使用原属性名。

访问器属性中包含get和set两个函数

get中负责读取受保护的属性值

set中负责先验证再修改受保护的值。

访问器属性不再需要value和writable。但是，需要设置enumerable和configurable

访问器属性如何发挥作用:

因为访问器属性是冒名顶替的，所以，外人访问访问器属性时，和访问普通属性是一样的

只不过访问器属性执行时，会自动根据外人做到操作，来选择get或set来调用。

如果外人试图获取属性值时，自动调用get函数

如果外人试图修改属性值时，自动调用set函数

2. 保护对象的结构

单词列表

1. strict 严格

2. assign 赋值

Cannot assign to read only property 'eid'

not can to read only property eid assign

不 能 给 只读 属性 赋值

3. property 属性。特指程序中的对象中的属性

比如: lilei.sname lilei.sage

4. ...able结尾 表示可以xxxx的

writable: 可修改的

configurable: 可配置的

enumerable: 可遍历的 - for in

5. define: 定义