正课:

\*\*\*\*ES5

严格模式

\*\*\*\*保护对象: 保护属性和防篡改

Object.create

\*\*\*call,apply,bind

数组API

严格模式:

什么是: 比普通js运行机制更严格的模式

为什么: js中存在很多广受诟病的缺陷

1. 允许给未声明的变量赋值

2. 静默失败

何时: 今后所有项目必须在严格模式下开发

如何:

在当前代码段的顶部添加"use strict";

在两个范围启用严格模式:

在全局启用严格模式: 在全局添加"use strict"

何时: 新项目，必须在全局启用

仅在函数内启用严格模式: 只在函数内顶部添加"use strict"

何时: 旧项目，向严格模式迁移

要求:

1. 禁止给未声明的变量赋值！

2. 将所有静默失败升级为错误！

\*\*\*\*保护对象:

为什么: 传统js中，对象的属性值和结构，可随意更改，毫无自保和验证能力

何时: 今后几乎所有的对象，都要不给严格规定保护

ES5中的变化:

对象的属性也是一个小的对象:

对象的属性，包含四大特性:

{

value: 属性值,

writable:true/false, //控制是否可修改属性

enumerable:true/false,//控制是否可用for in遍历到该属性

//仅控制for in遍历，用.依然可以访问

configurable:true/false //控制:

//1. 禁止删除该属性

//2. 禁止修改其它特性的值

//一旦改为false，不可逆！

//今后，只要修改其它特性，都要伴随修改configurable为false——双保险

}

如何:

获取属性的四大特性:

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

修改属性的四大特性:

Object.defineProperty(obj,"属性名",{

特性名:值,

... : ... ,

})

问题: defineProperty每次只能修改一个属性

解决: defineProperties 一次可同时修改多个属性

Object.defineProperties(obj,{

属性名:{

特性名:值,

... : ...

},

属性名:{

特性名:值,

... : ...

},

})

问题: 普通属性的四大特性，只能提供基本的保护

无法使用自定义规则保护属性

解决: 用访问器属性:

访问器属性:

ES5将对象的属性分为: 数据属性和访问器属性

数据属性: 直接存储属性值的属性

访问器属性: 不直接存储数据，仅提供对其它数据属性的保护

为什么: 数据属性只能对自己提供基本的保护，无法使用自定义逻辑保护自己。

何时: 只要用自定义属性保护数据属性时，就只能用访问器属性

如何: 2步:

1. 首先定义一个隐藏的实际存储数据的数据属性

2. 然后定义访问器属性保护数据属性

Object.defineProperties(obj,{

\_age:{ //数据属性

value:25,

writable:true,

enumerable:false, //隐藏

configurable:false

},

age:{ //访问器属性

get(){ //在试图用访问器属性取值时，自动调用

return this.\_age;

},

set(val){ //在试图给访问器属性赋值时，自动调用

//val可自动获得要赋的新值

if(val在范围内)

this.\_age=val;

else 报错！

},

enumerable:true,

configurable:false

}

})

使用访问器属性: 用法和普通数据属性完全一样！

防篡改：禁止修改对象的结构

三个级别:

1. 防扩展: 禁止为对象强行添加新属性:

Object.preventExtensions(obj)

原理: 对象都有一个内部属性: extensible, 默认true

preventExtensions其实是将extensible改为false！

2. 密封: 在防扩展的同时，再禁止删除所有属性

Object.seal(obj)

原理: 将所有属性的configurable特性都改为false

3. 冻结: 在密封的同时，再禁止修改所有属性值

何时: 如果一个对象，连属性值都不让随便改，就要冻结

如何: Object.freeze(obj)

原理: 将所有属性的writable都自动改为false

Object.create:

三件事: 基于现有父对象，创建一个子对象，再扩展新属性

何时: 如果只有父对象，没有构造函数时，也想创建子对象

如何:

var child=Object.create(father,{

//defineProperties

属性名:{

特性名:值,

... : ...

},

属性名:{

特性名:值,

... : ...

},

})