正课:

1. OOP

2. ES5

1. OOP

继承:

内置对象: 11个

String Number Boolean —— 包装类型

什么是: 专门封装一个原始类型的值，并提供操作原始类型值的API的特殊类型

为什么: 原始类型的值本身什么功能/属性都没有

何时: 只要用原始类型的值调用函数或访问属性时，都会自动使用包装类型

如何:

比如: var n=345.678; n.toFixed(2)

typeof n number

new Number(345.678).toFixed(2)

345.68

var s=" 张 东 "; s.trim()

typeof s string

new String(" 张 东 ").trim()

"张 东"

鄙视: 判断一个对象是不是数组类型，共有几种方式

typeof 可区分原始类型，对象和函数

不能进一步细致区分对象的具体类型

1. 判断原型对象:

obj.\_\_proto\_\_===Array.prototype

Array.prototype.isPrototypeOf(obj)

2. 判断构造函数

obj.constructor===Array

obj instanceof Array

(is) 实例

问题: 不够严谨

3. 判断对象的class属性

每个对象都有一个内部隐藏的class属性

只在创建对象时，保存对象的类型名

之后，不随继承关系改变而改变

问题: 无法用.直接访问class

解决: 只有顶级父对象Object.prototype中的toString()方法才能输出对象的class

问题: 各类型的原型对象中都重写了toString()方法，子对象默认无法调用顶级父对象中的toString()

解决: .call()

函数名.call(对象)

让对象强行调用本调不到的函数

判断class属性是不是数组:

Object.prototype.toString.call(obj)==="[object Array]"

4. Array.isArray(obj)

静态方法 vs 实例方法:

实例方法: 保存在类型的原型对象中，只允许当前类型的子对象使用。其它类型的子对象不能使用。

何时: 只允许当前类型的子对象使用

如何:

1. 存储位置: 所有实例方法都定义在原型对象中

2. 调用: 必须先创建该类型的子对象，再用子对象调用实例方法。

静态(static)方法: 不定义在原型对象中，任何对象都可使用的方法。

何时: 任何对象都可使用

如何:

1. 存储位置: 直接添加到构造函数对象上

2. 调用: 不用创建任何子对象，用构造函数直接.调用

多态: 同一个函数在不同情况下表现出不同的状态

2种:

1. 重载:

2. 重写override:

什么是: 如果子对象觉得从父对象继承来的成员不好用！可在子对象本地定义同名成员，覆盖父对象同名成员

为什么: 子对象觉得从父对象继承来的成员不好用

何时: 如果子对象觉得从父对象继承来的成员不好用

如何: 在子对象本地定义同名成员，覆盖父对象同名成员

自定义继承关系: 3种:

1. 只修改一个子对象的父对象:

child.\_\_proto\_\_=father;

Object.setPrototypeOf(child,father)

2. 修改所有子对象的父对象

构造函数.prototype=father;

强调: 时机: 定义构造函数后，立刻更换！

3. 两种类型间的继承:

问题: 两种类型，拥有部分相同的属性结构和方法定义

解决: 定义一个公共的抽象父类型

如何: 2步:

1. 定义抽象父类型

1. 定义父类型构造函数，包含相同部分的属性结构

2. 定义父类型的原型对象，包含相同部分的方法

2. 让子类型继承抽象父类型

1. 在子类型构造函数中借用父类型构造

问题: 直接调用父类型构造函数，无法为子对象添加父类型中的属性

原因: 直接调用一个函数，其中的this默认都指window

解决: .call

函数.call(替换this的对象, 参数, ...)

将函数内的this替换为指定的对象，并传入参数

总结: 将来，只要函数中的this不是想要的，都可用call替换

2. 让子类型原型对象继承父类型原型对象

2. ES5:

ECMAScript3.1 ECMAScript5 ECMAScript6

ES5:

严格模式: 比普通js运行要求更严格的模式

为什么: js语言本身有广受诟病的缺陷

何时: 今后所有js程序，都要运行在严格模式下

如何: 2种：

1. 新项目: <script>或js文件顶部: "use strict"

2. 旧项目: 逐个函数向严格模式迁移：

在function内部: "use strict"

包括:

1. 禁止给未声明的变量赋值

解决: 内存泄漏，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

3. 普通函数调用中的this不再默认指向window

解决: 内存泄漏，全局污染

4. 不建议使用arguments arguments.callee arguments.caller

作业: 使用递归和循环两种方式，实现计算费波纳茨数列的函数: 1 1 2 3 5 8 13 21 34 55

要求: fib(5) 5

fib(10) 55