SHANXI UNIQUE TECHNOLOGY CO.,LTD.

优逸客
SHANXI UNIQUE TECHNOLOGY

JavaScript对象

使用对象时，只关注对象提供的功能，不关注其内部细节

JavaScript对象

一、名词解释:

1.基于对象

一切皆对象，以对象的概念来编程。

2.面向对象编程(oop Object oriented programming)

A.对象

就是人们要研究的任何事物，不仅能表示具体事物，还能表示抽象的规则，计划或事件。

属性的无序集合，每个属性可以存一个值(原始值，对象，函数)

B.对象的属性和行为

属性:用数据值来描述他的状态

行为:用来改变对象行为的方法

C.类

具有相同或相似的性质的对象的抽象就是类。对象的抽象，就是类，类的具体化(实例化)就叫做对象

创建(声明)对象

1.json方法(javascript object notation) 原生格式

```
var obj={};
```

2.构造函数方法

```
function fun1 () {
}
var obj=new fun1();
```

3.Object方法

```
var obj=new Object();
```

对象组成

- 对象的组成

方法(行为、操作)：函数 过程、动态的

对象下的函数：对象的方法

属性——变量：状态

对象下的变量：对象的属性

添加属性和方法

如果属性的值是函数，我们叫做他是对象的方法，否则是属性。

1.构造方法

A.声明的时候添加

B.声明以后再添加

2.json方法

A.声明的时候添加

var obj={属性名:属性值，属性名2:属性值2，属性名3:属性值3，.....};

B.声明以后再添加

访问对象的属性和方法

- 访问属性:

  引用值.属性

  引用值["属性"]

- 访问方法:

  引用值.属性();

  引用值["属性"]()

销毁对象

- 对象=null;

JavaScript垃圾回收机制

- 局部变量回收

当函数执行完成以后，局部变量会被javascript垃圾回收机制，立即清除。

- 对象回收

当对象在页面中没有引用时，javascript会将他视作垃圾，在某个时刻将他销毁。

删除对象的属性

- delete 对象.属性

对象属性的遍历

- 我们可以通过for in遍历对象

```
for (var i in Objct){
        Object[i];
}
```

对象的存储方式

- 变量保存的仅仅是对象的引用地址
- 对象保存在堆当中，每创建一个对象，就开辟一块内存。
- 当javascript引擎检测到对象没有引用的时候，将他当作垃圾，等待回收。在某一时刻回收垃圾对象。

instanceof

- instanceof 用来检测某个对象是否是某个构造函数的实例。

什么是面向对象

- 以对象的思想去编程，就是面向对象编程
- 所谓"面向对象编程",其实指的是一种编程思想模式。

  原先我们所用的就是面向过程编程(函数式编程)
- 我们把系统自带的对象，叫做系统对象

系统对象，无法满足我们的需求。我们需要自己写对象完成功能。

- 使用对象时，只关注对象提供的功能，不关注其内部细节。

面向过程编程

比如制作一部手机：

1.列出配置单

品牌：小米

size:5.7寸屏

color:red green blue

打电话

发短信

基于android

2.买配件

cpu 主板 内存 天线 屏幕 话筒 功放...

3.组装

cpu 内存 天线 话筒 功放 安装到主板上

屏幕跟主板连接

安装手机壳

4.成品

面向对象编程

你有一条 手机自动生产线：

只需要按下按钮 => 出来1部手机

但是实现的过程，我们不需要了解。

了解面向对象的特点

1.只要一部手机,自己可以手动组装，使用面向过程方式最合适。
但要300部，量产 只需要输入数量300(面向对象 大规模使用)
2.面向对象要构思生产线的制造方法 (封装)

在内部实现：

1)如何去 购买配件 需要函数实现

2)如何去 组装主板 需要函数实现

3)如何给 主板安装手机壳 需要函数实现

在外部提供：

1)只提供 输入数量 (不提供 操作按钮)

2)只提供 出手机的接口

（这就是构建类函数并封装，封装后那些功能函数只能调用。）

3.通过自动化生产线 生产出的是一部一部的手机(类的实例化就是对象)
7.生产出来的手机 具有相同的特征 (类的对象有相同属性)
5.修改了机器的配置单,生产出的手机配置发生变化（这就是类和对象的关系）
6.觉得生产出来手机屏小，修改配置 6寸 （面向对象易于维护和改进）
7.有一天我要需要一个子品牌手机，在原先生产线添加一个品牌 (面向对象继承)

面向对象编程(oop)的基本概念

- 对象 是通过属性和方法组成
- 类 一个共享相同结构和行为的对象的集合。
- 抽象 抽取问题核心 (程序好坏,关键在此)
- 封装 不考虑内部实现，只考虑功能使用
- 继承 从已有对象上，继承出新的对象 (复用代码)
- 多态 由继承而产生相关的不同的类

创建第一个面向对象


```
var obj={};
  obj.name="张三";  //属性
  obj.show=function (){   //方法
      alert(obj.name);
  }
  obj.show(); //调用

  面向对象中有一个 this关键字指向
  var obj={};
  obj.name="张三";  //属性
  obj.show=function (){   //方法
      alert(this.name);
  }

  obj.show(); //调用

  var obj2={};
  obj2.name="李四";  //属性
  obj2.show=function (){   //方法
      alert(this.name);
  }

  obj2.show(); //调用
```

代码是重复的，不够优化，在JavaScript中我们遇到重复的代码就会通过函数包含起来

面向对象之封装

- 把对象所有的组成部分组合起来，尽可能的隐藏对象的部分细节，使其受到保护，只保留有限的接口和外部发生联系。

封装：工厂函数方式


```
function tv (color,size,brand){
        var TV={};
        TV.color=color;
        TV.size=size;
        TV.brand=brand;
        TV.look=function(){
        alert("看电视");
        }
        TV.DVD=function(){
        alert("看 DVD");
        }
        TV.play=function(){
        alert("玩游戏");
        }
        return TV;
}

var dianshi=tv("red",42,"海信");
var dianshi2=tv("blue",52,"haier");
```

封装：构造函数方式

函数功能：

1.封装一段功能代码

2.充当类的函数 我们叫做构造函数

封装：构造函数方式


```
function tv(color,size,brand){
        this.color=color;
        this.size=size;
        this.brand=brand;
        this.look=function(){
        alert("看电视");
        }
        this.DVD=function(){
        alert("看 DVD");
        }
        this.play=function(){
        alert("玩游戏");
        }
}
var TV=new tv("red",42,"Haier");
```

工厂方式、构造函数方式缺点

- 实例化出代码重复

原型方式

利用了对象的 prototype 属性，可以把它看成创建新对象所依赖的原型。

```
function tv(){
}
tv.prototype.color=red;
tv.prototype.DVD=function(){
alert("看 DVD");
}
tv.prototype.play=function(){
alert("玩游戏");
}
var TV=new tv();
```

混合模式

综合上述集中方式的，总结出较好的创建对象的方法；

```
function tv(color,size,brand) {
        this.color=color;
        this.size=size;
        this.brand=brand;
        this.play=function() {
        alert("玩游戏");
        }
}
tv.prototype.look=function() {
        alert("看电视");
}
tv.prototype.dvd=function() {
        alert("DVD");
}
var tv1=new tv("red",42,"Haier")
```

面向对象之继承

对象的一个类可以从现有的类中派生， 并且拥有现有的类的方法或是属性，这和过程叫做继承。被继承的类叫做父类或是基类，继承的类叫做子类。

通俗来讲就是一个对象拥有另一个对象的属性和方法，就叫做继承

优点：

- 提高代码重用性
- 提高代码可维护性
- 提高代码逻辑性

继承方式：

1. 原型继承
2. 对象冒充的形式(改变 this 指针)

原型继承


```
function person(){
        this.name="张三";
        this.say=function(){
        alert(this.name)
        }
}
function student(){
}
student.prototype=new person();
var zhangsan=new student()
zhangsan.say();
```

对象冒充的形式(改变 this 指针)

- call

obj1.fun.call(obj2,参数 1.....)

call 方法的第一个参数的值赋值给类(即方法)中出现的 this

call 方法的第二个参数开始依次赋值给类(即方法)所接受的参数

- obj1.fun.apply(obj2,[参数 1,.....])

apply 方法接受 2 个参数，

A、第一个参数与 call 方法的第一个参数一样，即赋值给类(即方法)中出现的 this

B、第二个参数为数组类型，这个数组中的每个元素依次赋值给类(即方法)所接受的参数

call


```
function person (name) {
        this.name=name;
        this.say=function() {
                alert(this.name)
        }
}
function student (name) {
        person.call(this);
        this.name=name;
}
var zhangsan=new person('张三');//实例化 人 张三
zhangsan.say();
var lisi=new student ("李四");   //实例化李四学生
lisi.say();
```

apply


```
function person (name,age) {
        this.name=name;
        this.age=age
        this.say=function() {
                alert(this.name)
        }
}
function student () {
        this.role="学生";
        person.apply(this,["zhangsan",18])
}
var zhangsan=new student ();
alert(zhangsan.name)
alert(zhangsan.age)
alert(zhangsan.role)
zhangsan.say();
```

让对象 1 的方法冒充成对象 2 的方法

对象的继承顺序

本身方法→本身原型→父对象方法→父对象原型→顶层对象原型

```
Object.prototype.say=function() {
      alert("我是顶层的方法");
}
function person () {
      this.say=function() {
            alert("我是父类的方法");
      }
}
person.prototype.say=function() {
      alert("我是父类原型的方法");
}
```

```
function study () {
      this.say=function() {
            alert("本身的方法");
      }
}
study.prototype=new person;
study.prototype.say=function() {
      alert("本身原型的方法");
}
var zhangsan=new study ();
alert(zhangsan.say)
```

谢谢观看...

SHANXI **UNIQUE**
TECHNOLOGY CO.,LTD.