

0.回顾

1.JS对象【JS是基于对象，不是面向对象】

- 1.1.Java对象【各编程语言的面向对象的三大特征】
- 1.2.JS对象概念：JS是基于对象
- 1.3.基于Object创建一个对象【优点：容易扩展】
【代码不可重用】
- 1.4.基于字面量赋值方式创建一个对象
【代码不可重用】
- 1.5.常见的内置对象

2.j04_a01_创建person对象

3.JS对象2【JS构造函数=Java的类】

- 3.0.Java的构造函数
- 3.1.定义构造函数，再用构造函数创建对象
【JS构造函数 == Java的类】
优点：【代码可重用】
缺点：【代码不容易扩展——添加属性或方法只能到定义的地方】
- 3.2.原型prototype
【空构造函数，配合prototype属性，定义属性或方法】
优点：【代码可重用】
优点：【代码容易扩展——添加属性或方法可以在另一个js中】

4.j04_a02_创建Person函数

0.回顾

BOM == js将Browser理解成对象 = 浏览器对象模型

DOM == js将HTML文档理解成对象 = 文档对象模型

core-dom ：节点的导航

html-dom ：节点的CRUD 【重点和难点】

css-dom ： style和className

1.JS对象【JS是基于对象，不是面向对象】

1.1.Java对象【各编程语言的面向对象的三大特征】

封装：把属性和方法放在一起，隐藏该隐藏的，暴露该暴露的。

继承：是通过扩展父类来创建新类的方式，新类可重用父类中的属性和方法。

多态：

同一个父类引用【继承】，指向不同子类对象时【替换】，
同一个行为执行不同的操作，表现不同的状态【重写】。

面向过程：没有封装的概念，方法归方法，属性作为全局变量

面向对象：属性和方法放在类里，方法可以直接访问该类中的相关属性

方法访问其它类的属性，需要实例化。

在现实世界中，万物皆对象。

继承：出于蓝，胜于蓝部分（重写）

Son extends Father, 重写biz()方法

Daught extends Father, 重写biz()方法

父.biz() 开个线下便利店

子.biz()重写 开个网上商城

女.biz()重写 开个服装店

多态：多个子类【继承】，子类必须有重写相关方法【重写】，变量指向子类对象

```
Father s1 = null;
```

```
s1 = new Father();
```

```
s1.biz(); // ??
```

```
s1 = new Son();
```

```
s1.biz(); // ??
```

```
s1= new Daught ();
```

```
s1.biz(); // ?
```

1.2.JS对象概念：JS是基于对象

■ 对象是包含相关属性和方法的集合体

◆ 属性

◆ 方法

■ 什么是面向对象

◆ 面向对象仅仅是一个概念或者编程思想

◆ 通过一种叫做原型的方式来实现面向对象编程

JS

1.3.基于Object创建一个对象【优点：容易扩展】

【代码不可重用】

■ 基于Object对象的方式创建对象

语法

```
var 对象名称=new Object( );
```

通过 . 添加属性和方法

示例

```
var flower=new Object();  
flower.name="长春花";  
flower.genera="夹竹桃科 长春花属";  
flower.area="非洲、亚热带、热带以及中国大陆的华东、西南、中南等地";  
flower.uses="观赏或用药等";  
flower.showName=function(){ alert(this.name); }  
flower.showName();
```

l63_lw.js是对163.js框架的一个扩展

```
var flower=new Object();  
flower.name="长春花";  
flower.genera="夹竹桃科 长春花属";  
flower.area="非洲、亚热带、热带以及中国大陆的华东、西南、中南等地";  
flower.uses="观赏或用药等";  
flower.showName=function(){ alert(this.name); }  
flower.showName();
```

163.js

163_lw.js

1.4.基于字面量赋值方式创建一个对象

【代码不可重用】

■ 使用字面量赋值方式创建对象

示例

```
var flower={  
    name:"长春花",  
    genera:"夹竹桃科 长春花属",  
    area:"非洲、亚热带、热带以及中国大陆的华东、西南、中南等地",  
    uses:"观赏或用药等",  
    showName:function(){ alert(this.name); }  
}  
flower.showName();
```

1.5.常见的内置对象

■ 常见的内置对象

- ◆ String（字符串）对象
- ◆ Date（日期）对象
- ◆ Array（数组）对象
- ◆ Boolean（逻辑）对象
- ◆ Math（算数）对象
- ◆ RegExp对象

2.j04_a01_创建person对象

3.JS对象2【JS构造函数=Java的类】

3.0.Java的构造函数

```
public class Father{  
    private int age;
```

```
public Father(int age){
    this.age = age;
}

}

public class Entry{

    public static void main(String[] args){

        Father f1 = new Father(58);
        // (1) new : 代表在heap堆中开辟一个对象空间
        // (2) Father : 对象空间的规格大小, 通过Father类计算
        // (3) (58) : 调用并执行Father类的构造函数, 该构造函数只能一个整型参数

        // 最后, 将该空间的地址引用赋值给f1变量, f1是存放在栈中

    }

}
```

3.1.定义构造函数，再用构造函数创建对象

【JS构造函数 == Java的类】

优点：【代码可重用】

缺点：【代码不容易扩展——添加属性或方法只能到定义的地方】

```
function Flower(name,genera,area,uses){
    this.name=name,
    .....
    this.showName=function(){
        alert(this.name);
    }
}
```

对象
↓
类 =

构造函数始终都应该以一个大写字母开头

```
var flower1=new Flower("长春花","夹竹桃科 长春花属","非洲、亚
热带、热带以及中国大陆的华东、西南、中南等地","观赏或用药等")
flower1.showName();
```

为什么JS中称为 构造函数创建对象？原因：flower1.constructor属性指向构造函数名

■ constructor属性指向Flower

列

```
alert(flower1.constructor===Flower);
alert(flower2.constructor===Flower);
alert(flower3.constructor===Flower);
```

3.2.原型prototype

【空构造函数，配合prototype属性，定义属性或方法】

优点：【代码可重用】

优点：【代码容易扩展——添加属性或方法可以在另一个js中】

- 每个函数都有一个prototype属性，这个属性是一个指针，指向一个对象
- prototype就是通过调用构造函数而创建的那个对象实例的原型对象

- 每个函数都有一个prototype属性，这个属性是一个指针，指向一个对象
- prototype就是通过调用构造函数而创建的那个对象实例的原型对象

```
var flower1 = new Flower(...);
```

```
flower1.prototype == Flower.prototype属性
```

```
function Flower(){  
  
    }  
    Flower.prototype.name="曼陀罗花";  
    Flower.prototype.genera="茄科 曼陀罗属";  
    Flower.prototype.area="印度、中国北部";  
    Flower.prototype.uses="观赏或药用";  
    Flower.prototype.showName=function() {  
        alert(this.name);  
    }  
    var flower1=new Flower();  
    flower1.showName();  
    var flower2=new Flower();  
    flower2.showName();  
    alert(flower1.showName==flower2.showName);  
}
```

prototype方式容易扩展

```
function Flower(){
```

```
}
```

```
Flower.prototype.name="曼陀罗花";
```

```
Flower.prototype.genera="茄科 曼陀罗属";
```

```
Flower.prototype.area="印度、中国北部";
```

```
Flower.prototype.uses="观赏或药用";
```

```
Flower.prototype.showName=function() {
```

```
    alert(this.name);
```

```
}
```

```
var flower1=new Flower();
```

```
flower1.showName();
```

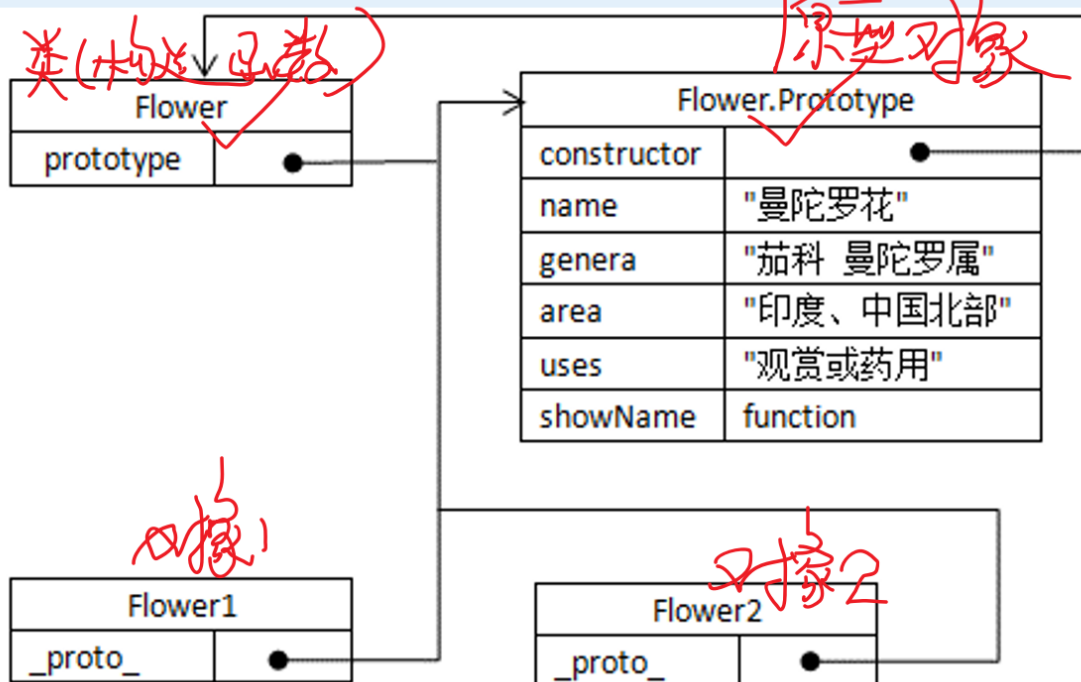
```
var flower2=new Flower();
```

```
flower2.showName();
```

```
alert(flower1.showName==flower2.showName);
```

163.js

163_lw.js



4.j04_a02_创建Person函数

【构造函数和原型方式实现】

参看本文的相关代码