

用new调用函数的四步走

上下文规则总结

规则	上下文
对象.函数()	对象
函数()	window
数组[下标]()	数组
IIFE	window
定时器	window
DOM事件处理函数	绑定DOM的元素
call和apply	任意指定

用new操作符调用函数

- ◆ 现在，我们学习一种新的函数调用方式：

new 函数()

- ◆ 你可能知道new操作符和“面向对象”息息相关，但是现在，我们先不探讨它的“面向对象”意义，而是先把用new调用函数的执行步骤和它上下文弄清楚

用new操作符调用函数

- ◆ JS规定，使用new操作符调用函数会进行“四步走”：
 - 1) 函数体内会自动创建一个空白对象
 - 2) 函数的上下文 (this) 会指向这个对象
 - 3) 函数体内的语句会执行
 - 4) 函数会自动返回上下文对象，即使函数没有return语句


四步走详解

```
function fun() {  
    this.a = 3;  
    this.b = 5;  
}  
  
var obj = new fun();  
console.log(obj);
```

四步走详解 - 第1步

- ◆ 第1步：函数体内会自动创建一个空白对象

```
function fun() {  
    this.a = 3;  
    this.b = 5;  
}
```

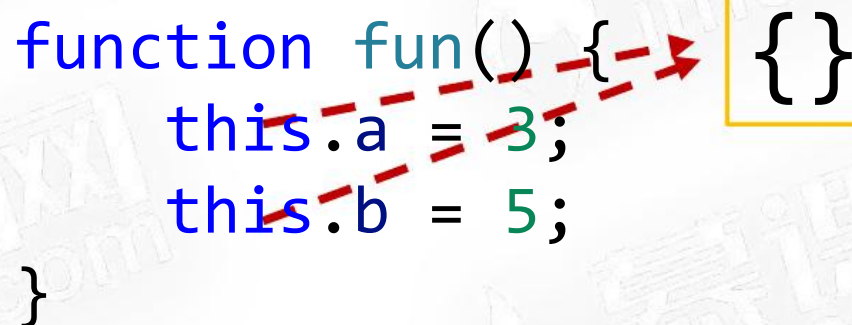


```
var obj = new fun();  
console.log(obj);
```

四步走详解 - 第2步

- ◆ 第2步：函数的上下文（this）会指向这个对象

```
function fun() {  
  this.a = 3;  
  this.b = 5;  
}
```

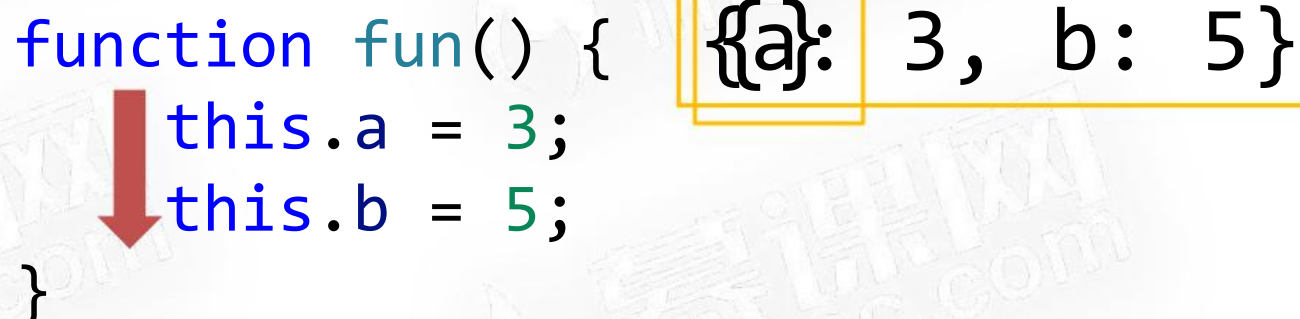


```
var obj = new fun();  
console.log(obj);
```

四步走详解 - 第3步

- ◆ 第3步：执行函数体中的语句

```
function fun() {  
  this.a = 3;  
  this.b = 5;  
}
```



```
var obj = new fun();  
console.log(obj);
```

四步走详解 - 第4步

- ◆ 第4步：函数会自动返回上下文对象，即使函数没有return语句

```
function fun() {  
    this.a = 3;  
    this.b = 5;  
}
```

`{a: 3, b: 5}`

`return this;`

返回的上下文对象被
变量obj接收

`{a: 3, b: 5}`

```
var obj = new fun();  
console.log(obj);
```

四步走详解

```
function fun() {  
    this.a = 3;  
    this.b = 5;  
}
```

```
var obj = new fun();  
console.log(obj);
```


上下文规则总结

规则	上下文
对象.函数()	对象
函数()	window
数组[下标]()	数组
IIFE	window
定时器	window
DOM事件处理函数	绑定DOM的元素
call和apply	任意指定
用new调用函数	秘密创建出的对象

构造函数

什么是构造函数

- ◆ 我们将之前书写的函数进行一下小改进：

```
function People(name, age, sex) {  
    this.name = name;  
    this.age = age;  
    this.sex = sex;  
}
```

接收三个参数

this上绑定同名属性

什么是构造函数

```
function People(name, age, sex) {  
    this.name = name;  
    this.age = age;  
    this.sex = sex;  
}  
  
var xiaoming = new People('小明', 12, '男');  
var xiaohong = new People('小红', 10, '女');  
var xiaogang = new People('小刚', 13, '男');
```

传入三个参数

什么是构造函数

```
{name: "小明", age: 12, sex: "男"}
```

```
{name: "小红", age: 10, sex: "女"}
```

```
{name: "小刚", age: 13, sex: "男"}
```

什么是构造函数

- ◆ 用new调用一个函数，这个函数就被称为“构造函数”，任何函数都可以是构造函数，只需要用new调用它
- ◆ 顾名思义，构造函数用来“构造新对象”，它内部的语句将为新对象添加若干属性和方法，完成对象的初始化
- ◆ 构造函数必须用new关键字调用，否则不能正常工作，正因如此，开发者约定构造函数命名时首字母要大写

构造函数名称首字母约定大写

构造函数名称首字母大写，暗示它是一个构造函数，调用时必须用new关键字

```
function People(name, age, sex) {  
    this.name = name;  
    this.age = age;  
    this.sex = sex;  
}  
  
var xiaoming = new People('小明', 12, '男');  
var xiaohong = new People('小红', 10, '女');  
var xiaogang = new People('小刚', 13, '男');
```

构造函数名称首字母约定大写

◆ 请判断对错：

一个函数名称首字母大  了，它就是构造函数

◆ 一定要记住：一个函数是不是构造函数，要看它是否用new调用，而至于名称首字母大写，完全是开发者的习惯约定

如果不用new调用构造函数

```
function People(name, age, sex) {  
    this.name = name;  
    this.age = age;  
    this.sex = sex;  
}  
  
People('小明', 12, '男');  
People('小红', 10, '女');  
People('小刚', 13, '男');
```

构造函数中的this不是函数本身

```
function People(name, age, sex) {  
    this.name = name;  
    this.age = age;  
    this.sex = sex;  
}  
  
var xiaoming = new People('小明', 12, '男');  
var xiaohong = new People('小红', 10, '女');  
var xiaogang = new People('小刚', 13, '男');
```

尝试为对象添加方法

```
function People(name, age, sex) {  
  this.name = name;  
  this.age = age;  
  this.sex = sex;  
  this.sayHello = function () {  
    console.log('我是' + this.name + ', 我' + this.age + '岁了');  
  };  
}  
var xiaoming = new People('小明', 12, '男');  
var xiaohong = new People('小红', 10, '女');  
var xiaogang = new People('小刚', 13, '男');  
xiaoming.sayHello();  
xiaohong.sayHello();  
xiaogang.sayHello();
```

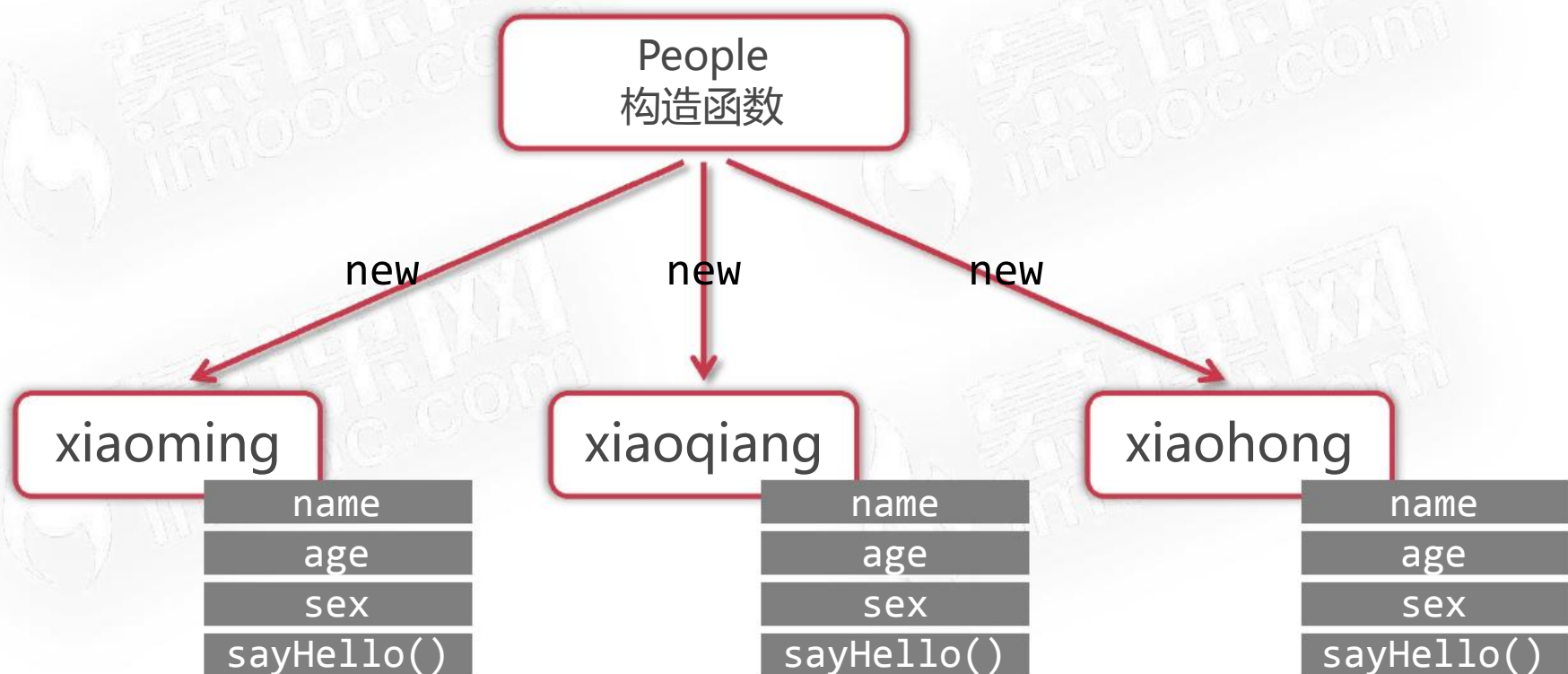
我是小明，我12岁了

我是小红，我10岁了

我是小刚，我13岁了

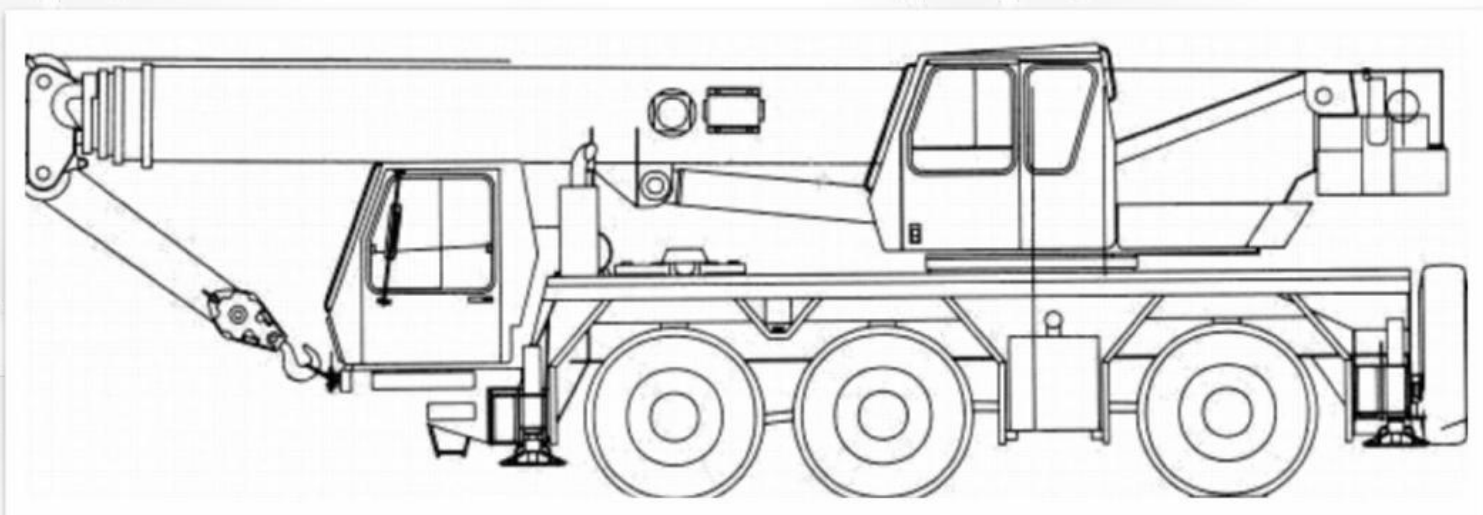
类与实例

类与实例



类好比是“蓝图”

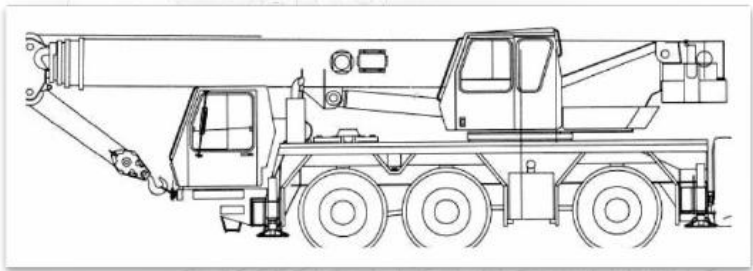
- ◆ 如同“蓝图”一样，类只描述对象会拥有哪些属性和方法，但是并不具体指明属性的值



实例是具体的对象



类和实例



类



实例



实例



实例

类和实例

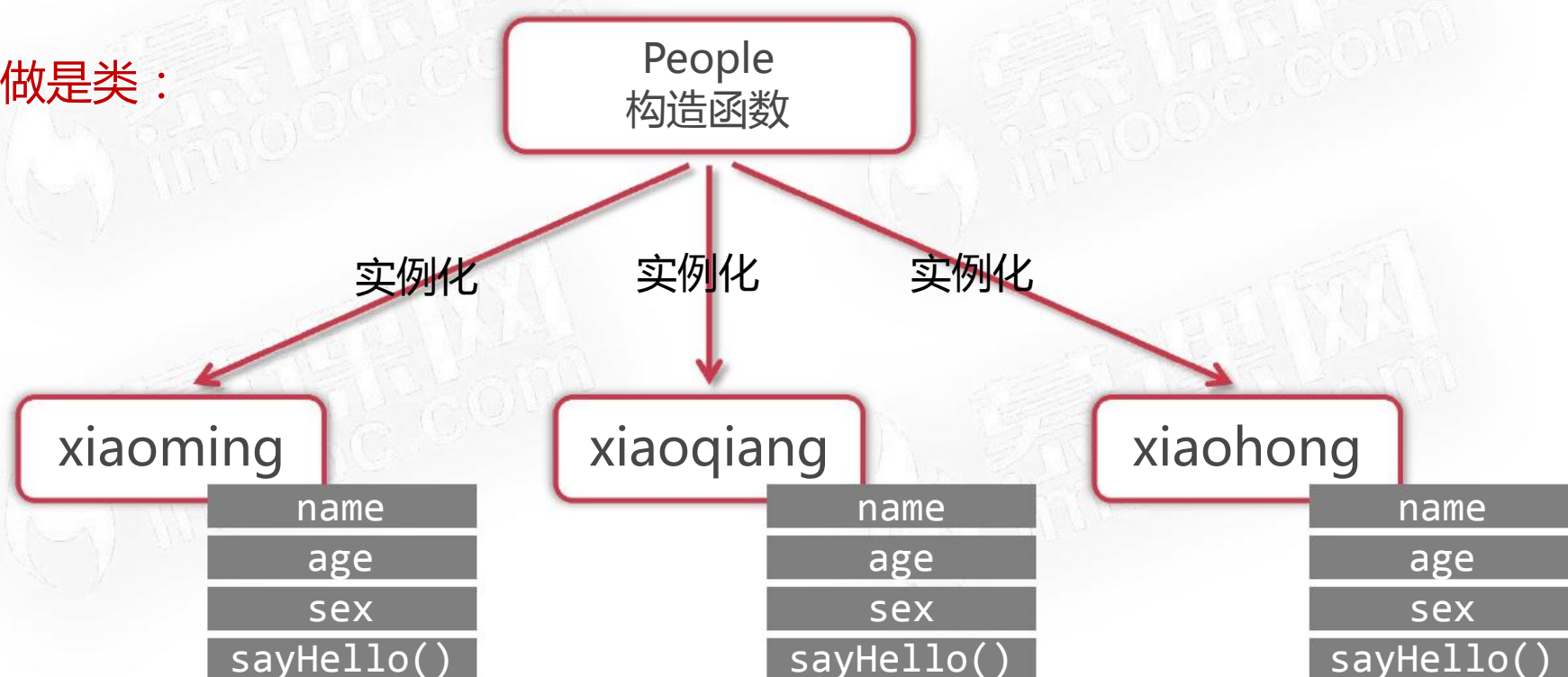
- ◆ “狗” 是类，“史努比” 是实例、“小白” 是实例



构造函数和“类”

可以看做是类：

实例：



构造函数和“类”

- ◆ Java、C++等是“面向对象”（object-oriented）语言
- ◆ JavaScript是“基于对象”（object-based）语言
- ◆ JavaScript中的构造函数可以类比于OO语言中的“类”，写法的确类似，但和真正OO语言还是有本质不同，在后续课程还将看见JS和其他OO语言完全不同的、特有的原型特性