【继承的前置课程】

创建对象的方法

对象字面量

```
let obj = {}
```

```
let obj = new Object()
```

工厂模式

```
let createPerson = function (name, age){
    let o = new Object()
    o.name = name;
    o.age = age;
    o.sayName = function(){
        console.log('nihao')
    }
    return o
}
let instance = createPerson('lwh', 18)
```

解决了创建多个对象
缺点：创建的对象类型不明确
没有对象标识

特点：显式的构造了新对象

构造函数模式

```
function Person(name,age){
    this.name = name;
    this.age = age;
    this.sayName = function(){
        console.log('lwh')
    }
}
let instance = new Person('lwh',18)
```

解决了对象标识问题

缺点：每个实例都要赋值一遍构造函数的属性
实例上会有多余的不需要的属性

特点：没有显示的构造新对象
没有return
方法和属性直接赋值给了this

原型模式

```
// 第一版
function Person(){}
Person.prototype.name = 'lihua';
Person.prototype.age = '23';
Person.prototype.sayName = function(){
    console.log(this.name)
}
let instance = new Person()
instance.name = 'lwh'
```

```
// 第二版
// 一个个的设置protoype过于麻烦
function Person() {}
Person.prototype = {
    constructor: Person, // 为了让Person原型指向从Object变为Person
    name: 'lihua',
    age: 18,
    sayName: function(){
        console.log(this.name)
    }
}
```

```
// 第三版
// 直接令constructor会让属性默认可以枚举，原生的不可以枚举
// 最后使用Object.defineProperty()设置constructor属性
Object.defineProperty(Person.prototype, 'constructor', {
    enmuerable: flase,
    value: Person
})
```

备注：