27-第二十七章 面向对象 oop 继承(二)

一、对象 与 函数的关系

1、 `Object` 函数是Function 的一个实例

```
1.      Object.constructor  == Function  //true
2.
```

2、 `函数` 即是Function 的实例，也是Object 的实例

```
1.
2.      function fn(){}
3.      fn.constructor  == Function  //true
4.      fn.constructor  == Object    //true
5.
```

3、 `{}` 与 Object 的关系

```
1.  var obj = {};
2.  obj.constructor  === Object   //true
3.
```

总结， `对象是由函数构造出来的`

十一、原型链

> obj.prototype.`isPrototypeOf(` fn.prototype `)` //fn.prototype 是否继承自 obj.prototype

```
function fn(){}
var  obj = new fn();
Object.prototype.isPrototypeOf( fn.prototype ) //  true
Function.prototype.isPrototypeOf( Object.prototype ) //true

```

十二、 对象继承

在上面例子中 可以发现,给对象的constructor.prototype添加方法属性 对象就会继承 如果要实现一个对象继承其他对象我们这样做

> 一、利用`call()`及`for in`继承
> 如

```
function inherit(){

            fn.call(obj3,'旺财');
            for(key  in  fn.prototype){
                if(!obj3[key]){
                    obj3[key] = fn.prototype[key];
                }

            }
        }
```

```
function inherit(constructor,obj,ownAttr){
        constructor.call(obj3,ownAttr);
        for(key  in  constructor.prototype){
            if(!obj[key]){
                obj[key] =constructor.prototype[key];
            }

    }


}
```

一、 构造函数实例方式继承

> 一、 利用obj.constructor.prototype 继承对象 `自身属性` 及 `继承属性`

```
 function fn(name){
  this.name = name;
  }
  fn.prototype.index = '88';
  function fn2(age){
      this.age = age;
  }
  fn2.prototype  = new fn('旺财');
  var obj2 = new fn2(66);
  console.log(obj2.name);//旺财
  console.log(obj2.index);//88
   console.log(obj2.age);//88
```

> 二、利用 prototype 继承对象 `自身属性` 及 `继承属性`

```
function fn(name){
    this.name = name;
}
fn.prototype.index = '88';
function fn2(age){
     this.age = age;
 }
var obj = new fn('二狗');
fn.prototype  = Object.create(fn.prototype);
console.log(obj2.name);//二狗
console.log(obj2.index);//88

```

在上面例子中 可以发现,给对象的constructor.prototype添加方法属性 对象就会继承 如果要实现一个对象继承其他对象我们这样做

二、`obj.constructor.prototype` 与 对象`obj.__proto__` 的关系 ,带__为非标准属性

```
function fn(name){
     this.name = name;
 }
var obj = new fn('二狗');
 console.log( obj.__proto__ === fn.prototype ) //true,
```

三、 对象继承对象

```
var obj = {name : 'hello'}
var  obj2 = {age:30}
obj2.__proto__ = obj;
console.log( obj2.name);//hello
```

继承 构造函数 实例对象

```
function fn(name){
        this.name = name;
    }
    var obj = new fn('hello');
    var obj2 = {
        age:30,
        __proto__:obj
    }
    console.log(obj2.name);//hello;
```

案例：用面向对象改写 限制范围拖拽

案例：用面向对象改写 自动轮播