对象的理解

目录：

1.什么是对象

2.创建对象的方法

3.对象属性的访问和添加

4.对象值的获取(对象是无序的，不能通过下标来获取)

5. 对象的方法

Object.assign(目标对象，源对象)

Object.is()

Object.create()

Object.toString()

Object.definePropety(obj,key,propety)

Object.keys()

Object.values()

1.什么是对象

        对象是无序的，可以存储不同类型的值。

        {key:value}

        有属性和方法

2.创建对象的方法

        字面量          var obj={}

        new方法       var obj=new Object(可)

        create方法    var obj=Object.create(fn,d可)

        new的方法参数问题：

               有参数，则对象就是参数所对应的数据类型；

                无参数，或参数是null或undefined，返回空对象。

        new的方法注意对象内部this的指向问题，this是在执行该对象的对象，不是定义的时候决定的。

3.对象属性的访问和添加

        .  或 ['不规则名称']或[变量]

        遇到['变量']。[]会解析内部变量，

**var *obj***={

**x**:1

}

***obj***.**name**=**'shao'**;

***console***.log(***obj***.**name**) *//shao*

***console***.log(***obj***.**x**) *//1*

4.对象值的获取(对象是无序的，不能通过下标来获取)

        **[key]**，将key值存入一个数组中，

**var *obj***={**x**:1,**y**:2,**z**:3};

**var *arr***=[**'x'**,**'y'**,**'z'**];

***arr***.forEach(**function** (key) {

***console***.log(***obj***[value]); *//1 2 3*

})

        **Object.keys()方法：**

**var *obj***={**x**:1,**y**:2,**z**:3};

***Object***.keys(***obj***).forEach(**function** (key) {

***console***.log(***obj***[key])

})

        **for  in方法：**

**var *obj***={**x**:1,**y**:2,**z**:3};

**for**(**var *key*** in ***obj***){

***console***.log(***obj***[***key***])

}

        for  of方法：

5. 对象的方法

**Object.assign(目标对象，源对象)**

        对象拼接。目标对象可以使{}。将源对象拼接到目标对象上，目标对象改变，源对象不变。

**var *obj1***={**x**:1,**y**:2,**z**:3}

**var *obj2***={**a**:**'vv'**}

***Object***.assign(***obj1***,***obj2***);

***console***.log(***obj1***); *//{x: 1, y: 2, z: 3, a: "vv"}*

**Object.is()**

        判断两个值是否相等

**var *obj1***={**x**:1,**y**:2,**z**:3}

**var *obj2***={**a**:**'aaa'**}

***console***.log(***Object***.is(***obj1***,***obj2***)) *//false*

***console***.log(***Object***.is(***NaN***,***NaN***)) *//true*

**Object.create()**

        创建对象，不改变原来对象

**var *obj1***={**x**:1,**y**:2,**z**:3}

***obj1***.**name**=**'aaa'**;

**var *obj2***=***Object***.create(***obj1***,{**y**:{**value**:100}})

***console***.log(***obj2***); *//{y:100}*

***console***.log(***obj2***.**name**) *//aaa*

**Object.toString()**

        转成字符串

**var *obj1***={**x**:1,**y**:2,**z**:3}

***console***.log(***obj1***.toString()) *//[object Object]*

**var *arr1***=[**'a'**,**'v'**,**'c'**]

***console***.log(***arr1***.toString()) *//a,v,c与arr.join(',')功能相同*

**var *str1***=**'miaov'**

***console***.log(***str1***.toString()) *//miaov*

Object.definePropety(obj,key,propety)

        操作对象的属性，新增属性

        一个独享的属性背后有很多隐藏细节

        value: 该属性对应的值。可以是任何有效的 JavaScript 值（数值，对象，函数等）。默认为 undefined；

        writable: 当且仅当该属性的writable为true时，value才能被赋值运算符改变。默认为 false；

        configurable: 当且仅当该属性的 configurable 为 true 时，该属性描述符才能够被改变，同时该属性也能从对应的对象上被删除。默认为 false；

        enumerable: 当且仅当该属性的enumerable为true时，该属性才能够出现在对象的枚举属性中。默认为 false。

    注意：通过 . 或 [] 增加的属性，默认都为true，通过defineProperty增加的属性，默认都为false

**var *obj***={**x**:1,**y**:2,**z**:3}

***Object***.defineProperty(***obj***,**'y'**,{

**value**:100,

**writable**:**true**,

**configurable**:**true**,

**enumerable**:**true**

})

***console***.log(***obj***)