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

* 合并对象
* 返回第一个参数 默认修改第一个参数
* 只能合并可枚举属性

**let** obj1 = {

name: 'zs',

age: 12,

sex: 'man'

}

**let** obj2 = {

name: 'ww',

age: 20,

class: '1610'

}

**function** **extend** (obj, ...arg) {

arg.forEach(**function** (file) {

**for** (**let** i **in** file) {

obj[i] = file[i];

}

})

**return** obj;

}

**let** obj = extend({}, obj1, obj2);

console.log(obj);

**let** objs = Object.assign({}, obj1, obj2);

console.log(objs);

**Object.getOwnPropertyDescriptor(obj, 属性名, [配置项]);**

* 获取当前对象的属性的描述对象
* 返回值为对象

**let** obj = {

name: 'zs',

age: 12

}

**let** o = Object.getOwnPropertyDescriptor(obj, 'name');

console.log(o);

*<!--{ value: 'zs', 属性值-->*

*<!-- writable: true, 是否可编辑-->*

*<!-- enumerable: true, 是否可枚举-->*

*<!-- configurable: true 是否可配置（删除）-->*

*<!--}-->*

**Object.defineProperty(obj, 属性名, [配置项]);**

* 定义属性

**let** obj = {

name: 'zs'

}

Object.defineProperty(obj, 'age', {

value: 12,

configurable: true, *// 默认false*

enumerable: true, *// 默认为false*

writable: true *// 默认false*

})

obj.age = 22;

console.log(obj)

console.log(Object.keys(obj)) *// 打印不出来的为不可枚举属性*

Object.defineProperty(obj, 'aa', {

**get** () { *// 当使用该键值对时函数被触发*

console.log(0)

**return** 's'

},

**set**(val) { *// 当设置该变量时函数触发*

console.log(val);

}

})

*// 使用了get和set函数之后 aa被称之为访问器属性，不适用时属性称之为数据属性*

obj.aa = 'ss';

console.log(obj.aa)

**Object.keys(obj);**

**Object.values(obj);**

**Object.entries(obj);**

这三个返回的也是一个遍历器对象，可以使用for of 进行遍历

[**Object.is**](http://Object.is)**(v1, v2) 比较两个值是否完全相等**

NaN === NaN *// false NaN和任何数据类型都不想等包括自身*

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