Assign和concat均能实现对对象的拷贝，但是都只是对字段、属性直接copy，如果有深层次的object或array嵌套，会导致浅拷贝，需遍历对象，做特殊处理。

Assign做拷贝时，会覆盖同属性的值

var o1 = { a: 1, b: 1, c: 1 };

var o2 = { b: 2, c: 2 };

var o3 = { c: 3 };

var obj = Object.assign({}, o1, o2, o3);

console.log(obj); // { a: 1, b: 2, c: 3 }

只有可遍历的对象能赋值

var v1 = 'abc';

var v2 = true;

var v3 = 10;

var v4 = Symbol('foo');

var obj = Object.assign({}, v1, null, v2, undefined, v3, v4);

// Primitives will be wrapped, null and undefined will be ignored.

// Note, only string wrappers can have own enumerable properties.

console.log(obj); // { "0": "a", "1": "b", "2": "c" }

字符串可遍历。

Assign拷贝时，若报错或中断，后续值的拷贝会被强制终止

var target = Object.defineProperty({}, 'foo', {

value: 1,

writable: false

}); // target.foo is a read-only property

Object.assign(target, { bar: 2 }, { foo2: 3, foo: 3, foo3: 3 }, { baz: 4 });

// TypeError: "foo" is read-only

// The Exception is thrown when assigning target.foo

console.log(target.bar); // 2, the first source was copied successfully.

console.log(target.foo2); // 3, the first property of the second source was copied successfully.

console.log(target.foo); // 1, exception is thrown here.

console.log(target.foo3); // undefined, assign method has finished, foo3 will not be copied.

console.log(target.baz); // undefined, the third source will not be copied either.

Assign拷贝只考虑跟属性，只会完全覆盖根属性的值

const defaultOpt = {

title: {

text: 'hello world',

subtext: 'It\'s my world.'

}

};

const opt = Object.assign({}, defaultOpt, {

title: {

subtext: 'Yes, your world.'

}

});

console.log(opt);

// 预期结果

{

title: {

text: 'hello world',

subtext: 'Yes, your world.'

}

}

// 实际结果

{

title: {

subtext: 'Yes, your world.'

}

}

Concat只是把数组连接起来，用空数组可以做浅复制

连接数组的特例，带非数组的值，把值转成数组了

var alpha = ['a', 'b', 'c'];

var alphaNumeric = alpha.concat(1, [2, 3]);

console.log(alphaNumeric);

// results in ['a', 'b', 'c', 1, 2, 3]