ES6 PPT1511131: 变量解构赋值

Thursday, November 5, 2015 10:15 AM

可迭代/遍历对象（eg. []）与一般对象（eg. {}）之解构模式：

1. 可迭代模式（例 - 数组/[]）：

   典型应用：let [v0, v1, v2] = [10, 11, 12]; // let v0=10, v1=11, v2=12;

   简单解析：let [var0, var1, var2] = aIterable; // let var0=aIterable[0], var1=aIterable[1], var2=aIterable[2];

   深度模型：let [expressionA, expressionB] = aIterable; // let expressionA=aIterable[0], expressionB=aIterable[1];

2. 一般对象模式（例 - 不可迭代对象/{}）：

   典型应用：let {a, b, c} = {a:101, b:102, c:103}; // let a=101, b=102, c=103;

   简单解析：let {varA, varB, varC} = oHasAttr; // let varA=oHasAttr.varA, varB=oHasAttr.varB, varC=oHasAttr.varC;

   深度模型：let {attrA:exprA, attrB:exprB } = oHasAttr; // let exprA=oHasAttr.attrA, exprB=oHasAttr.attrB;

可迭代对象/数组解构赋值：

1. 赋值表达式右侧必须为可迭代对象：eg. [], "string is also iterable"

```
let [a, [[b], c]] = [1, [[2], 3]]; // a=1, b=2, c=3
let [s0, s1, s2, s3] = "我也可以被解构"; // s0=我, s1=也, s2=可, s3=以
(function(){ let [va, [vb, vc], vd] = arguments; })('a', ['b', 'c'], 'd'); // va='a', vb='b', vc='c', vd='d'
let [v1, v2, v3] = (function* gen(){ yield '1st'; yield '2nd'; yield '3rd'; })(); // v1='1st', v2='2nd', v3='3rd'
let [v4, ...vmany] = [4, 5, 6, 7]; // v4=4, vmany=[5,6,7]
var [foo] = 1; // TypeError: Invalid attempt to destructure non-iterable instance
var [foo] = false; // TypeError: Invalid attempt to destructure non-iterable instance
var [foo] = NaN; // TypeError: Invalid attempt to destructure non-iterable instance
var [foo] = undefined; // TypeError: Invalid attempt to destructure non-iterable instance
var [foo] = null; // TypeError: Invalid attempt to destructure non-iterable instance
var [foo] = {}; // TypeError: Invalid attempt to destructure non-iterable instance
```

2. 无值元素被赋予 undefined 值：

```
let [d, , f] = [4, 5, 6]; // d=4, f=6
const [g, h] = [7, 8, 9]; // g=7, h=8
var [i, j, ...vlot] = [10]; // i=10, j=undefined, vlot=[]
let [k, l, [m, n]] = [undefined, 'love', [, 'net', 'IAmIgnored']]; // k=undefined, l='love', m=undefined, n='net'
let [s4,[s5,s6,s7],s8,s9] = "四五八九十"; // s4=四, s5=五, s6=undefined, s7=undefined, s8=八, s9=九
let [s4,[s5a,s5b,s5c],s6,s7] = "四五六七八"; // s4=四, s5a=五, s5b=undefined, s5c=undefined, s6=六, s7=七
```

3. 应用默认值赋值：先执行右侧表达式取值，无值者自左侧依次尝试取默认值

```
let [o, p='pig', q, [r='run', s='sun']] = ['open', 'PI', , [undefined, null]]; // o='open', p='PI', q=undefined, r='run', s=null
```

```
let [t=100, u=++t, v=t--] = [200, 300,]; // t=199, u=300, v=200
```

一般对象解构赋值：

1. 表达式右侧必须为含有 Attribute 的 对象：eg. {}, [], "", window, 123

```
let {a, b, c} = {b:2, c:3, a:1}; // a=1, b=2, c=3
let {d} = {D:4}; // d=undefined
let {cos, sin} = Math; // cos=Math.cos, sin=Math.sin
let {length} = [11,12,13]; // length=3
var {foo1} = false; // foo1===undefined
var {foo2} = NaN; // foo2===undefined
var {foo3} = undefined; // TypeError: Cannot read property 'foo3' of undefined
var {foo4} = null; // TypeError: Cannot read property 'foo4' of null
```

2. 重定义输出变量名：

```
let {a:a, b:B, c:c} = {b:2, c:3, a:1}; // a=1, B=2, c=3, b: ReferenceError: b is not defined
let {alert:myAlert, confirm:myConfirm} = window; // myAlert=window.alert,
myConfirm=window.confirm
let {attrA:[varStr, varNum]} = {attrA:["IAmString", 456]}; // varStr='IamString', varNum=456
let {attrB:{attrStr, attrObj:varObj}} = {attrB:{attrStr:"IAmString2", attrObj:{attrBool:true}}};
//attrStr=, varObj=
let {body:{innerHTML:theBodyHTML}} = document; // theBodyHTML=document.body.innerHTML
```

3. 应用默认值赋值：

```
let {d=4, e, f, g:myg=7} = {e:null, f:undefined}; //d=4, e=null, f:undefined, myg=7
let {t=100, u=++t, v=t--, w=(t=50)} = {u:300, t:200}; //t=50, u=300, v=200, w=50
```

4. 非声明情况下赋值：可使用 () 将表达式括起来

```
var tan; {tan} = Math; // Expected an assignment or function call and instead saw an expression.
var abs; ({abs} = Math); // abs=Math.abs
```

5. 函数参数的解构赋值：

```
//不一样的输出1
function move({x, y} = { x: 0, y: 0 }) { return [x, y]; }
move({x: 3, y: 8}); // [3, 8]
move({x: 3}); // [3, undefined]
move({}); // [undefined, undefined]
move(); // [0, 0]
//不一样的输出2
function move({x = 0, y = 0} = {}) { return [x, y]; }
move({x: 3, y: 8}); // [3, 8]
move({x: 3}); // [3, 0]
move({}); // [0, 0]
move(); // [0, 0]
```

各类应用：交换变量值[x, y]=[y, x]，快捷提取对象/JSON成员赋值...，import/export