# ES2015

ECMAScript 6（简称ES6）是于2015年6月正式发布的JavaScript语言的标准，正式名为ECMAScript 2015（ES2015）。它的目标是使得JavaScript语言可以用来编写复杂的大型应用程序，成为企业级开发语言 [1] 。

## 扩展运算符

扩展运算符，也就是展开语法(Spread syntax), 可以在函数调用/数组构造时, 将数组表达式或者string在语法层面展开；还可以在构造字面量对象时, 将对象表达式按key-value的方式展开。(译者注: 字面量一般指 [1, 2, 3] 或者 {name: "mdn"} 这种简洁的构造方式)

<**script**>  
 **var *arr1*** = [1, 2, 3];  
 **var *arr2*** = [4, 5, 6];  
  
 *// var arr3 = arr.concat(arr2);* **var *arr3*** = [...***arr1***, ...***arr2***, 5, 98];  
 ***console***.log(***arr3***);  
  
 **var *str*** = **'123456'** *// var str2 = str.split('');* **var *str2*** = [...***str***, **'hello'**];  
 ***console***.log(***str2***);  
  
 **var *obj*** = {  
 **a**: 1,  
 **b**: 2,  
 **c**: 3  
 }  
 **var *obj2*** = { ...***obj***, **d**: **'4'** };  
 ***console***.log(***obj2***);  
</**script**>

求一组数字的最大值：不支持直接把数组作为Math.max的参数，所以使用扩展运算符：

<**script**>  
 **var *arr*** = [1,22,31,5,56]  
 ***console***.log(***Math***.max(...***arr***));  
</**script**>

剩余语法(Rest syntax) 看起来和展开语法完全相同，不同点在于, 剩余参数用于解构数组和对象。从某种意义上说，剩余语法与展开语法是相反的：展开语法将数组展开为其中的各个元素，而剩余语法则是将多个元素收集起来并“凝聚”为单个元素。

## 解构赋值

ES6解构赋值语法：用来将数组中的值或对象中的属性取出来区分为不同变量

### 对象的解构赋值

<**script**>  
 **let *obj*** = {  
 **name**: **'小明'**,  
 **age**: 33  
 };  
  
 ***console***.log(***obj***.**name**, ***obj***.**age**);  
  
 **let** { ***name***, ***age*** } = ***obj***;  
 ***console***.log(***name***, ***age***);  
</**script**>

注意花括号中必须是属性名，不能是自己命名的变量，不能是：let {a,b}=obj  
也就是说键名和变量名是一致的

如果需要重命名，也就是设置别名，使用冒号：

<**script**>  
 **let *obj*** = {  
 **name**: **'小明'**,  
 **age**: 33  
 };  
  
 **let** { **name**: ***str*** } = ***obj***;  
 ***console***.log(***str***);  
</**script**>

设置默认值，使用=号：

<**script**>  
 **let *obj*** = { **name**: **'张三'**, **age**: 19 , **c**: **undefined** }  
 **let** { ***c*** } = ***obj***;  
 ***console***.log(***c***); *// undefined*</**script**>

<**script**>  
 **let *obj*** = { **name**: **'张三'**, **age**: 19 , **c**: 100 }  
 **let** { ***c*** } = ***obj***;  
 ***console***.log(***c***); *// 100*</**script**>

<**script**>  
 **let *obj*** = { **name**: **'张三'**, **age**: 19 , **c**: **undefined** }  
 **let** { ***c*** = 100 } = ***obj***;  
 ***console***.log(***c***); *// 100*</**script**>

<**script**>**let *obj2*** = {  
 **name**: **'小明'**,  
 **age**: 33,  
 **d**: **false** };  
 **let** { ***d*** = 100 } = ***obj2***;  
 ***console***.log(***d***); *// false*</**script**>

------------

同时使用别名和默认值：

<**script**>  
 **let *obj*** = { **name**: **'张三'**, **age**: 19 , }  
 **let** { **gender**:***g***=**'girl'** } = ***obj***;  
 ***console***.log(***g***); *// girl*</**script**>

<**script**>  
 **let *obj*** = { **name**: **'张三'**, **age**: 19 , **gender**:**'boy'** }  
 **let** { **gender**:***g***=**'girl'** } = ***obj***;  
 ***console***.log(***g***); *// 'boy'*</**script**>

支持剩余参数：

<**script**>  
 **let *obj*** = {  
 **name**: **'xiaoming'**,  
 **age**: 25,  
 **sex**: **'man'**,  
 **height**: 170  
 }  
 **let** { ***name***, ***age***, ...***r*** } = ***obj***;  
 ***console***.log(***name***, ***age***, ***r***);  
</**script**>

### 数组的解构赋值

<**script**>  
 **var *arr*** = [**'张三'**,**'李四'**,**'王五'**,**'赵六'**];  
 **var** [***a***,***b***,***c***] = ***arr***;  
 ***console***.log(***a***,***b***,***c***); *// 张三 李四 王五*</**script**>

数组的解构赋值是严格按照顺序的，不想取的值，可以留空占位：  
支持设置默认值

<**script**>  
 **var *arr*** = [**'张三'**,**'李四'**,**'王五'**,**'赵六'**];  
 ***arr***[0] = **undefined**;  
 **var** [***a***=**'hello'**,***b***, ,***d***] = ***arr***; *//如果第0个不存在，则取默认值* ***console***.log(***a***,***b***,***d***); *// hello 李四 赵六*</**script**>

支持剩余参数

<**script**>  
 **var *arr*** = [**'张三'**,**'李四'**,**'王五'**,**'赵六'**];  
 **var** [***a***,***b***,...***r***] = ***arr***;  
 ***console***.log(***a***,***b***,***r***); *// 张三 李四 (2) ["王五", "赵六"]*</**script**>

## 对象的新增

### 新写法

<**script**>  
  
 *// ES5对象写法：* **var *a*** = 1;  
 **var *objes5*** = {  
 **a**: ***a***,  
 fn: **function**(){  
 ***console***.log(**'hello es5'**);  
 }  
 };  
  
 **var *objes6*** = {  
 ***a***,  
 fn(){  
 ***console***.log(**'hello es6'**);  
 }  
 }  
  
 ***console***.log(***objes5***);  
 ***objes5***.fn();  
  
 ***console***.log(***objes6***);  
 ***objes6***.fn();  
</**script**>

### 对象的键名可以是变量

<**script**>  
  
 *// ES6对象的键名可以是变量，注意放到中括号里面：* **var *attrname*** = **'width'**;  
 **var *attrobj*** = {};  
 **var *attrfn*** = **'fn'**;  
 **var *obj*** = {  
 [***attrname***]: 100, *// 支持* [***attrobj***]: **"hello"**, *// 不支持对象作为键名* [***attrfn***](){ *// 支持* ***console***.log(**'hello es6'**);  
 }  
 };  
  
 ***console***.log(***obj***);  
 ***obj***.fn();  
</**script**>

### Object.is()方法

Object.is() 方法判断两个值是否为同一个值。

<**script**>  
 ***console***.log(1 == **'1'**); *// true* ***console***.log(***NaN*** === ***NaN***); *// false* ***console***.log(***Object***.is(1,1)); *// true* ***console***.log(***Object***.is(1,**'1'**)); *// false* ***console***.log(***Object***.is(***NaN***, ***NaN***)); *// true* ***console***.log(***Object***.is({},{})); *// false* **var *obj*** = {};  
 ***console***.log(***Object***.is(***obj***,***obj***)); *// true*</**script**>

### Object.assign()

Object.assign() 方法用于将所有可枚举属性的值从一个或多个源对象复制到目标对象。它将返回目标对象。  
如果目标对象中的属性具有相同的键，则属性将被源对象中的属性覆盖。后面的源对象的属性将类似地覆盖前面的源对象的属性。

<**script**>  
 **const *target*** = { **a**: 1, **b**: 2 };  
 **const *source*** = { **b**: 4, **c**: 5 };  
  
 **const *returnedTarget*** = ***Object***.assign(***target***, ***source***);  
  
 ***console***.log(***target***);  
 *// expected output: Object { a: 1, b: 4, c: 5 }* ***console***.log(***returnedTarget***);  
 *// expected output: Object { a: 1, b: 4, c: 5 }*</**script**>

### 示例

传统写法：

<**script**>  
 **function** *move*(obj) {  
 **var** defaultObj = {  
 **ease**: **'linear'**,  
 **duration**: 2000  
 };  
 **var** para = {  
 **ease**: obj.**ease** || defaultObj.**ease**,  
 **duration**: obj.**duration** || defaultObj.**duration** };  
 ***console***.log(para);  
 }  
 *move*({  
 **ease**: **'easeIn'**,  
 *// duration: 1000* })  
</**script**>

ES6写法：

<**script**>  
 **function** *move*(obj) {  
 **var** defaultObj = {  
 **ease**: **'linear'**,  
 **duration**: 2000  
 };  
 **var** para = {};  
 ***Object***.assign(para, defaultObj, obj,)  
 ***console***.log(para);  
 }  
 *move*({  
 **ease**: **'easeIn'**,  
 *// duration: 1000* })  
</**script**>