
WORK
TRUE
千锋教育
QIAN FENG EDUCATION
做真实的自己，用良心做教育
ES5和ES6
H5教学部

目录

1 ES5

2 ES6介绍

3 ES6新增内容

4 练习

做真实的自己，用良心做教育

千锋教育
QIAN FENG EDUCATION

ES5

1. JavaScript 的版本

JavaScript这种语言的基本语法结构是由ECMAScript来标准化的, 所以我们说的JavaScript版本一般指的是ECMAScript版本.

2009年12月，ECMAScript 5.0版正式发布

2011年6月，ECMAscript 5.1版发布

2015年6月，ECMAScript 6正式发布，并且更名为“ECMAScript 2015”。

2016年6月，ECMAScript6.1发布， 与ECMAScript6差异较小

ES5 : ECMAScript 5

ES5

ES5的严格模式：

所谓严格模式，从字面上就很好理解，即更严格的模式 在这种模式下执行，浏览器会对JS的要求更苛刻，语法格式要求更细致，更符合逻辑。

怪异模式：就是我们之前一直使用的开发模式，就叫怪异模式。因为很多时候出来的结果是非常怪异的，所以才称之为怪异模式。

使用严格（全局）模式

```
"use strict";
 n = 10;
 console.log(n);
```

使用严格（局部）模式

```
function fn(){
    "use strict";
    n = 10
    console.log(n);
}
```

ES5

Javascript 为什么要设立严格模式：

1.消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;
2.代码运行的一些不安全之处，保证代码运行的安全；
3.提高编译器效率，增加运行速度；

例如以下几点:

1. 不可以省略var声明变量
2. 禁止使用this关键字指向全局变量
3. 禁止使用八进制方法
4. 不允许在非函数的代码块内声明函数

ES5

ES5-bind

绑定一个新对象， 让函数中的this指向该对象， 一般在定时器中使用较多。

例如：

```
var name = "哈哈";
var obj = {name:"张三"};
var obj2 = {
    name: "李四",
    show: function(){
        console.log(this.name);
    }
}
setInterval(obj2.show.bind(obj), 2000);
```

ES5

ES5-Array新增

indexOf()：判断数组中是否包含某个元素， 和字符串的indexOf用法类似

forEach()：用来遍历数组中的每一项；这个方法执行是没有返回值的，对原来数组也没有影响

```
var arr = [1,2,3,4,5];
arr.forEach(function(value, index, array){
    console.log("值：" + value + ", 下标：" + index + ", 数组：" + array);
});
```

ES5

ES5-Array新增

map(): 和forEach非常相似，都是用来遍历数组中的每一项的，区别是map的回调函数中支持return返回

不管是forEach还是map 都支持第二个参数值，第二个参数的意思是把匿名回调函数中的this进行修改。

```
var arr = [1,2,3,4,5];
var newArr = arr.map(function(value, index, array){
        return value * 2;
});
console.log(arr);  //[1,2,3,4,5]
console.log(newArr);  //[2,4,6,8,10]
```

注意：forEach和map不支持IE8及以下

ES5

ES5-Array新增

reduce()用法： 接收一个函数作为累加器，数组中的每个值（从左到右）开始缩减，最终为一个值，是ES5中新增的一个数组逐项处理方法

```
var result = arr.reduce(function(pre, cur, index, array){
        console.log("前一个值：" + pre + "当前值：" + cur + ", 下标：" + index + ", 数组：" + array);
        return cur;
}, 10);
```

reduce接收四个两个参数， 第二个参数可选（为pre的初始值）

第一个参数是回调函数，该回调函数中又包含四个参数， 分别为：

pre：上一次调用回调函数时的返回值，或者初始值

cur：当前正在处理的数组元素

index：当前正在处理的数组元素下标

array：调用reduce()方法的数组

result: 遍历完数组后的最终结果

ES5

reduce()用法：

使用reduce()求和

```
var arr = [1,2,3,4,5];
var sum = arr.reduce(function(pre, cur){
        return pre + cur;
});
console.log(sum); //25
```

ES5

forEach、map以及reduce的相同点：

都是用于处理数组

都不兼容IE8及以下

forEach、map以及reduce的不同点：

forEach 方法是将数组中的每一个值取出做一些程序员想让他们做的事情

map 方法是将数组中的每一个值放入一个方法中做一些程序员想让他们做的事情后可以返回一个新的数组

reduce 方法 将数组中的每一个值与前面的值处理后得到的最终值

ES5

ES5-Array新增

filter()： 过滤出数组中你想要的元素， 不改变原数组

```
var arr = [1,2,3,4,5];
function fn(n) {
    return n >= 3;
}
var newArr = arr.filter(fn);
console.log(arr);  //[1,2,3,4,5]
console.log(newArr); //[3,4,5]
```

ES6介绍

什么是ES6？

ECMAScript 6（简称ES6）是JavaScript语言的下一代标准，已经在2015年6月正式发布了。

ES6的目标，是使得JavaScript语言可以用来编写大型的复杂的应用程序，成为企业级开发语言。

支持ES6的浏览器:

虽说ES6已经作为新一代标准发布了，但是各大浏览器对新功能实现支持的还需要一段时间，那么我们怎么知道自己使用的浏览器是否支持ES6的相应功能呢？

不用紧张，对ES6的支持可以查看kangax.github.io/es5-compat-table/es6/，在这里可以清晰的了解到不同版本的浏览器对ES6功能的支持情况。随着时间的推移，支持度会越来越高了。

由于浏览器对ES6的支持度还不完整， 所以在实际开发过程中我们通常需要将ES6的代码转换成ES5。

ES6介绍

ES6环境支持:

1，可以使用HBuilder最新版， HBuilder8.0版本支持ES6语法

2，也可以使用其他ES6编译器： **Traceur**

Traceur允许将ES6代码直接插入网页。首先，必须在网页头部加载Traceur库文件.

注意: script标签的type属性的值是module(或者traceur)，而不是text/javascript。这是Traceur编译器识别ES6代码的标识，编译器会自动将所有type=module的代码编译为ES5，然后再交给浏览器执行。

```
<!-- 加载Traceur编译器 -->
<script src="http://google.github.io/traceur-compiler/bin/traceur.js" type="text/javascript"></script>
<!-- 将Traceur编译器用于网页 -->
<script type="text/javascript">
        new traceur.WebPageTranscoder(document.location.href).run();
</script>
<script type= ‘module’ > 自己代码 </script>
```

ES6

ES6-let

let 是ES6中新增关键字。它的作用类似于var，用来声明变量，但是所声明的变量，只在let命令所在的代码块内有效。

```
if(true){
    var a = 1;
    let b = 2;
}
console.log(a);
console.log(b);  // 报错：ReferenceError: b is not defined
```

ES6

ES6-let

let关键字涉及到一个概念：块级作用域

ES6以前，只有全局作用域和函数局部作用域，ES6之后加入块级作用域，一个大括号{}我们称之为一个代码块，一个大括号{}的内部就是块级作用域

利用块级作用域可以有效解决以下问题

1.防止全局变量泄露

2.防止变量提升带来的覆盖问题

ES6

ES6-const

const 声明的是常量，一旦声明，值将是不可变的, const 也具有块级作用域， const 不可重复声明

```
const PI = 3.1415;
console.log(PI);
PI = 3;
console.log(PI);  //报错
```

ES6

ES6-String新增方法:

传统上，JavaScript只有 indexOf 方法，可以用来确定一个字符串是否包含在另一个字符串中。ES6又提供了三种新方法。

includes()：返回布尔值，表示是否找到了参数字符串。

startsWith()：返回布尔值，表示参数字符串是否在源字符串的头部。

endsWith()：返回布尔值，表示参数字符串是否在源字符串的尾部。

repeat()： 返回一个新字符串，表示将原字符串重复n次.

模板字符中，支持字符串插值

```
document.write(`Hello ${first} ${last}!`);   //注意引号
```

ES6

ES6-Array新增

Array.from方法用于将两类对象转为真正的数组：类似数组的对象（array-like object）和可遍历（iterable）的对象（包括ES6新增的数据结构Set和Map）

Array.from()还可以接受第二个参数，作用类似于数组的map方法，用来对每个元素进行处理.

Array.of(): 方法用于将一组值，转换为数组

Array.of(3, 11, 8) // [3,11,8]

find()和findIndex()

数组实例的find方法，用于找出第一个符合条件的数组成员。它的参数是一个回调函数，所有数组成员依次执行该回调函数，直到找出第一个返回值为true的成员，然后返回该成员。如果没有符合条件的成员，则返回undefined。

ES6

for-of 遍历集合：

这是目前遍历数组最简洁和直接的语法；

它避免了for-in的所有缺陷；

与forEach()不一样，它支持break，continue和return

entries() keys() values()

可以用for-of循环进行遍历，唯一的区别是keys()是对键名的遍历、values()是对键值的遍历，entries()是对键值对的遍历

ES6

ES6-Function新增

函数默认参数

```
function sayHello2(name='qianfeng'){
    document.write(`Hello ${name}`);  //注意引号
}
```

扩展运算符(spread)是三个点（...）,将一个数组转为用逗号分隔的参数序列。该运算符主要用于函数调用

```
var arr=['张三','李四','王五'];
function fn(a,b,c){
    console.log(`${a},${b},${c}`); //张三,李四,王五
}
fn(...arr);
//还可用于合并数组
[...arr1, ...arr2, ...arr3]
```

ES6

箭头函数：

ES6允许使用“箭头”（=>）定义函数

```
var fn = v => v;
```

上面的箭头函数等同于：

```
var fn = function(v) {
    return v;
}
```

如果箭头函数不需要参数或需要多个参数，就使用一个圆括号代表参数部分。

```
var fn = () => 5;  等同于：  var fn = function () { return 5 };
var sum = (a, b) => a+b; 等同于：
var sum = function(a, b) {
    return a+b;
};
```

ES6

如果箭头函数的代码块部分多于一条语句，就要使用大括号将它们括起来，使用return语句返回。

```
var sum = (a, b) => { let c = a+b; return c; }
```

由于大括号被解释为代码块，所以如果箭头函数直接返回一个对象，必须在对象外面加上括号。

```
var fn = id => ({ id: 11, name: "zhangsan" });
```

箭头函数使得表达更加简洁。

```
const isEven = n => n % 2 == 0;
const square = n => n * n;
```

ES6

箭头函数可以简化回调函数的写法：

```
// 正常函数写法
[1,2,3].map(function (x) {
   return x * x;
});

// 箭头函数写法
[1,2,3].map(x => x * x);
```

ES6

ES6-Object新增:

属性的简写:

```
var foo = 'bar';
var baz = {foo};    // 等同于 var baz = {foo: foo};
```

方法的简写:

```
var o = {
    method() {
        return "Hello!";
    }
};
```

ES6

属性名表达式:

```
let obj = {};
obj['a'+'bc'] = 123;
console.log(obj);
```

方法名表达式:

```
let obj = {
    ['h'+'ello']() {
        return 'hi';
    }
};
```

ES6

Object.is()用来比较两个值是否严格相等。它与严格比较运算符（===）的行为基本一致， 不同之处只有两个：一是+0不等于-0，二是NaN等于自身

Object.assign() 方法用来将源对象（source）的所有可枚举属性，复制到目标对象（target）。它至少需要两个对象作为参数，第一个参数是目标对象，后面的参数都是源对象。只要有一个参数不是对象，就会抛出TypeError错误。

```
var target = { a: 1 };
var source1 = { b: 2 };
var source2 = { c: 3 };
Object.assign(target, source1, source2);
console.log(target) // {a:1, b:2, c:3}
```

ES6

ES6-Set :

数据结构Set类似于数组，但是成员的值都是唯一的，没有重复的值。

```
var set = new Set([1,2,3,4,5,5,5,5]);
console.log(set.size); // 5
```

Set的属性和方法:

size : 数量

add(value)：添加某个值，返回Set结构本身

delete(value)：删除某个值，返回一个布尔值，表示删除是否成功

has(value)：返回一个布尔值，表示该值是否为Set的成员

clear()：清除所有成员，没有返回值

ES6

数组的去重问题

```
var arr = [1,2,1,2,3,4,3,4,6,6,2];
var set = new Set(arr);
var newArr = new Array(...set);
console.log(newArr);
```

ES6

ES6-WeakSet:

WeakSet和Set一样都不存储重复的元素, 用法基本类似，但有一些不同点, WeakSet的成员只能是对象，而不能是其他类型的值。

ES6

ES6-Map:

Map 是一个“超对象”，其 key 除了可以是 String 类型之外，还可以为其他类型（如：对象）

```
let map = new Map([[1, 'one'],[2, 'two'],[3, 'three']]);
```

他的方法和 Set 差不多:

size：返回成员总数。

set(key, value)：设置一个键值对。

get(key)：读取一个键。

has(key)：返回一个布尔值，表示某个键是否在Map数据结构中。

delete(key)：删除某个键。

clear()：清除所有成员。

keys()：返回键名的遍历器。

values()：返回键值的遍历器。

entries()：返回所有成员的遍历器。

ES6

ES6-解构赋值

数组（完全对称，不完全对称）

```
var [a,[b,c],d]  = [1,[2,3],4];
```

对象 (属性名相同，属性名不同)

```
var {name,age} = {name:'zhangsan',age:19}
```

函数

```
function func([x,y]){return x - y} ;
console.log( func([19,11]) );
```

解构赋值可以有效解决以下2个问题

1.快速提取JSON字符串

2.交换数组中的2个元素

ES6

ES6-Class:

ES6提供了更接近传统语言的写法，引入了Class（类）这个概念，作为对象的模板。通过class关键字，可以定义类。

```
class Point {
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }
    toString() {
         return '('+this.x+', '+this.y+')';
    }
}
```

注意： constructor方法是类的默认方法，通过new命令生成对象实例时，自动调用该方法。

ES6

Class的继承

Class之间可以通过extends关键字，实现继承

```
class Point {
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }
}
class ColorPoint extends Point {
    constructor(x, y, color) {
        super(x, y);
        this.color = color;
    }
    getData(){
        console.log(this.x+','+this.y+','+this.color);
        return this.x+','+this.y+','+this.color;
    }
}
```

ES6

ES6-Symbol

Symbol 是ES6引入的一种新的数据类型，来表示独一无二的值

数据类型有6种：Undefined、Null、布尔值（Boolean）、字符串（String）、数值（Number）、对象（Object） 现在有7种了， 加上Symbol

创建Symbol值 ，不能添加new

```
var s1 = Symbol();
var s2 = Symbol();
s1 === s2; // false
```

有参数的情况

```
var s1 = Symbol("foo");
var s2 = Symbol("foo");
s1 === s2; // false
```

ES6

Symbol用法：

作为属性名, 防止属性的值被更改

```
var obj = {};
var name = Symbol();
obj[name] = 'hello';
obj[name]; // "hello"
```

Symbol值作为对象属性名时，不能用点运算符

```
var obj = {};
var name = Symbol();
obj.name = 'zhangsan';
obj[name] = 'lisi';
console.log(obj.name, obj[name]);    //zhangsan,lisi
```

ES6

ES6-Generator

Generator Function: 生成器函数, 最大特点就是可以交出函数的执行权（即暂停执行）

Generator函数的写法： *号是为了区分于普通函数，

```
function* gen(x){
  var y = yield x + 2;
  return y;
}
var g = gen(1);
g.next() // { value: 3, done: false }
g.next() // { value: undefined, done: true }
```

整个Generator函数就是一个封装的异步任务，异步操作需要暂停的地方，都用 yield 语句注明，

next 方法的作用是分阶段执行 Generator 函数。每次调用 next 方法，会返回一个对象，表示当前阶段的信息（ value 属性和 done 属性）。value 属性是 yield 语句后面表达式的值，表示当前阶段的值；done 属性是一个布尔值，表示 Generator 函数是否执行完毕，即是否还有下一个阶段

ES6

JS中各种循环的区别：

for : JS语言诞生就有的循环， 也是使用最多的循环方式， 很多计算机语言都有的循环方式， 可用于遍历数组，字符串；

while : 很多计算机语言也都有的循环方式， 和for循环在使用上的区别是： for循环一般用于已知遍历次数的情况，while一般用在不知道遍历次数的情况；

do-while: 和while有点类似， 使用较少

for-in : 专门用于遍历对象的循环方式 ， 也可以遍历数组，但不推荐遍历数组

forEach: ES5新增的遍历方式，支持IE9+, 遍历数组的写法比较简洁，快速， 但是不能中断循环，不能使用break和continue；

for-of : ES6新增的遍历方式， 和for-in用法类似， 但比for-in强大很多， for-of可以遍历数组， 字符串， Set ， Map等.

练习

练习

1, 使用reduce求数组中所有元素的乘积， var arr = [2,3,4,5,6,7]

2, 使用reduce求数组中的最大数, var arr = [2,4,6,3,7,8,5]

3, 使用indexOf完成数组去重.

4. 使用for-of求5! (1*2*3*4*5)

5, 使用for-of求1!+2!+3!+4!+5!

6, 使用for-of求数组的最大数和最小数

7, 掌握数组去重的多种方法.

8, 掌握各种循环的使用

THANK YOU

千锋教育
QIAN FENG EDUCATION

做真实的自己，用良心做教育