# ECMScript6

官方文档(英)：<http://www.ecma-international.org/ecma-262/6.0/index.html>

阮一峰ES6入门：<http://es6.ruanyifeng.com/>

视频教程：[http://DeLyQiang.com/2017/06/03/es6/](http://jspang.com/2017/06/03/es6/)

# **第1节：ES6的开发环境搭建**

### 1.1**建立工程目录：**

先建立一个项目的工程目录，并在目录下边建立两个文件夹：****src****和****dist****

* src：书写ES6代码的文件夹，写的js程序都放在这里。
* dist：利用Babel编译成的ES5代码的文件夹，在HTML页面需要引入的时这里的js文件。
* 需要注意的是在引入js文件时，引入的是***dist***目录下的文件。

<script src="dist/index.js"></script>

### 1.2**初始化项目**

在安装Babel之前，需要用npm init先初始化我们的项目。打开终端或者通过cmd打开命令行工具，进入项目目录，输入下边的命令：

npm init -y

-y代表全部默认同意，就不用一次次按回车了。命令执行完成后，会在项目根目录下生产package.json文件。

{

"name": "10es6",

"version": "1.0.0",

"description": "这是es6",

"main": "index.js",

"scripts": {

"build": "babel src/index.js -o dist/index.js"

},

"keywords": [],

"author": "DeLyQiang",

"license": "ISC",

}

可以根据自己的需要进行修改，比如我们修改name的值。

### **1.3全局安装Babel-cli**

npm install -g babel-cli

虽然已经安装了babel-cli，只是这样还不能成功进行转换，如果你不相信可以输入下边的命令试一下。

babel src/index.js -o dist/index.js

你会发现，在dist目录下确实生产了index.js文件，但是文件并没有变化，还是使用了ES6的语法。因为我们还需要安装转换包才能成功转换，继续往下看吧。

### **1.4本地安装babel-preset-es2015 和 babel-cli**

npm install --save-dev babel-preset-es2015 babel-cli

安装完成后，我们可以看一下我们的package.json文件，已经多了devDependencies选项。

"devDependencies": {

"babel-cli": "^6.26.0",

"babel-preset-es2015": "^6.24.1"

}

### **1.5新建.babelrc**

在根目录下新建.babelrc文件，并打开录入下面的代码

{

"presets":[

"es2015"

],

"plugins": []

}

这个文件我们建立完成后，现在可以在终端输入的转换命令了，这次ES6成功转化为ES5的语法。

babel src/index.js -o dist/index.js

### **1.6简化转化命令：**

在学习vue 的时候，可以使用npm run build 直接利用webpack进行打包，在这里也希望利用这种方式完成转换。打开package.json文件，把文件修改成下面的样子。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | {  "name": "10es6",  "version": "1.0.0",  "description": "这是es6",  "main": "index.js",  "scripts": {  "build": "babel src/index.js -o dist/index.js"  },  "keywords": [],  "author": "dely",  "license": "ISC",  "devDependencies": {  "babel-cli": "^6.26.0",  "babel-preset-es2015": "^6.24.1"  }  } |

修改好后，以后我们就可以使用 npm run build 来进行转换了。

## **第2节：新的声明方式**

以前我们在声明时只有一种方法，就是使用****var****来进行声明，ES6对声明的进行了扩展，现在可以有三种声明方式了。

字面理解ES6的三种声明方式：

1. var：它是variable的简写，可以理解成变量的意思。
2. let：它在英文中是“让”的意思，也可以理解为一种声明的意思。
3. const：它在英文中也是常量的意思，在ES6也是用来声明常量的，常量你可以简单理解为不变的量。

### **2.1 var声明：**

var在ES6里是用来升级全局变量的，我们可以先作一个最简单的实例，用var声明一个变量a,然后用console.log进行输出。

var a='DeLyQiang';

console.log(a);

我们可以看到DeLyQiang在控制台已经被打印出来了。那如何理解它的作用是声明全局变量那？我们用匿名函数给他进行一个包裹，然后在匿名函数中调用这个a变量，看看能不能调用到。

var a="DeLyQiang";

window.onload= function(){

    console.log(a);

}

可以看到控制台输出了DeLyQiang，这证明var确实是全局的。如果你觉的这个不够直观说明var是全局声明，还可以用区块的方式进行调用测试，先看下面的代码。

var a=2;

{

   var a=3;

}

console.log(a);

这时打印出来的值是多少那？对，应该是3，因为var是全局声明的。

### **2.2 let局部声明**

通过两个简单的例子，我们对var的全局声明有了一定了解。那跟var向对应的是let，它是局部变量声明。还是上面的例子，我们试着在区块里用let声明。

var a=2;

{

   let a=3;

}

console.log(a);

这时候控制台打印出来的值就是2了。如果我们只在区块里声明，不再外部声明，我们打印a时就会报错，显示找不到变量。

{

   let a=3;

}

console.log(a);

上面两个例子说明了let是局部变量声明，let声明只在区块内起作用，外部是不可以调用的。

我感觉let还没有var好用，其实let是防止你的数据污染的，在大型项目中是非常有用处的。现在看一个循环的例子，我们来看一下let的好处。

### 2.3用var声明的循环

for(var i=0;i<10;i++){

console.log('循环体中:'+i);

}

console.log('循环体外:'+i);

你会发现循环体外的i变量被污染了，如果在外部再使用i时就会出现问题，这是开发者不想看到的。我们再利用let声明，就可以解决这个问题。

用let声明的循环

for(let i=0;i<10;i++){

console.log('循环体中:'+i);

}

console.log('循环体外:'+i);

你执行时会发现控制台报错了，找不到循环体外的i变量。通过两种声明的比较，可以明白let在防止程序数据污染上还是很有用处的。我们要努力去习惯用let声明，减少var声明去污染全局空间，在vue的使用中也要注意这点。

### **2.4const声明常量**

在程序开发中，有些变量是希望声明后在业务层就不再发生变化了，简单来说就是从声明开始，这个变量始终不变，就需要用const进行声明。

我们来一段用const声明错误的代码，在错误中学习const的特性也是非常好的。

const a="DeLyQiang";

var a='戴利强';

console.log(a);

在编译这段代码的过程中，你就会发现已经报错，无法编译了，原因就是我们const声明的变量是不可以改变的。const是很好理解的，我就不作过多的解释说明了。

这节课学了ES6的3种声明方式，var、let、const，这三种方式各有所长，既然已经学习了新技术，就要拥抱它，试着在项目中根据情况用let和const进行声明吧，不要再只使用var了。

# **第3节：变量的解构赋值**

ES6允许按照一定模式，从数组和对象中提取值，对变量进行赋值，这被称为解构。解构赋值在实际开发中可以大量减少我们的代码量，并且让我们的程序结构更清晰。也许你还是不太明白，那先来一个最简单的****数组解构赋值****来进行赋值。

### **3.1数组的解构赋值：**

简单的数组解构：

以前，为变量赋值，我们只能直接指定值。比如下面的代码：

|  |
| --- |
| let a=0;  let b=1;  let c=2; |

而现在我们可以用数组解构的方式来进行赋值。

|  |
| --- |
| let [a,b,c]=[1,2,3]; |

上面的代码表示，可以从数组中提取值，按照位置的对象关系对变量赋值。

****数组模式和赋值模式统一：****

可以简单的理解为等号左边和等号右边的形式要统一，如果不统一解构将失败。

|  |
| --- |
| let [a,[b,c],d]=[1,[2,3],4]; |

如果等号两边形式不一样，很可能获得undefined或者直接报错。

****解构的默认值****：

解构赋值是允许你使用默认值的，先看一个最简单的默认是的例子。

|  |
| --- |
| let [foo = true] =[];  console.log(foo); //控制台打印出true |

上边的例子数组中只有一个值，可能你会多少有些疑惑，我们就来个多个值的数组，并给他一些默认值。

|  |
| --- |
| llet [a,b="DeLyQiang"]=['戴利强']  console.log(a+b); //控制台显示“戴利强DeLyQiang” |

现在我们对默认值有所了解，需要注意的是****undefined****和****null****的区别。

|  |
| --- |
| let [a,b="DeLyQiang"]=['戴利强',undefined];  console.log(a+b); //控制台显示“戴利强DeLyQiang” |

undefined相当于什么都没有，b是默认值。

|  |
| --- |
| let [a,b="DeLyQiang"]=['戴利强',null];  console.log(a+b); //控制台显示“戴利强null” |

null相当于有值，但值为null。所以b并没有取默认值，而是解构成了null。

### **3.2对象的解构赋值**

解构不仅可以用于数组，还可以用于对象。

|  |
| --- |
| let {foo,bar} = {foo:DeLyQiang,bar:'戴利强'};  console.log(foo+bar); //控制台打印出了“DeLyQiang戴利强” |

****注意：****对象的解构与数组有一个重要的不同。数组的元素是按次序排列的，变量的取值由它的位置决定；而对象的属性没有次序，变量必须与属性同名，才能取到正确的值。

****圆括号的使用****

如果在解构之前就定义了变量，这时候你再解构会出现问题。下面是错误的代码，编译会报错。

|  |
| --- |
| let foo;  {foo} ={foo:DeLyQiang};  console.log(foo); |

要解决报错，使程序正常，我们这时候只要在解构的语句外边加一个圆括号就可以了。

|  |
| --- |
| let foo;  ({foo} ={foo:DeLyQiang});  console.log(foo); //控制台输出DeLyQiang |

### **3.3字符串解构**

字符串也可以解构，这是因为，此时字符串被转换成了一个类似数组的对象。

|  |
| --- |
| const [a,b,c,d,e,f]="DeLyQiang";  console.log(a);  console.log(b);  console.log(c);  console.log(d);  console.log(e);  console.log(f); |

总结：这节内容很简单，但是也有一些细节需要注意，在实战项目中解构Json数据格式还是很普遍的，有了ES6，提高了不少工作效率。

# **第4节：扩展运算符和rest运算符**

上节课学的解构已经可以大大增加我们的开发效率，这节课我们学习扩展运算符和rest运算符，它们都是…（三个点）。它们可以很好的为我们解决参数和对象数组未知情况下的编程，让我们的代码更健壮和简洁。

### **4.1对象扩展运算符（…）：**

当编写一个方法时，我们允许它传入的参数是不确定的。这时候可以使用对象扩展运算符来作参数，看一个简单的列子：

|  |
| --- |
| function DeLyQiang(...arg){      console.log(arg[0]);      console.log(arg[1]);      console.log(arg[2]);      console.log(arg[3]);    }  DeLyQiang(1,2,3); |

这时我们看到控制台输出了 1,2,3，undefined，这说明是可以传入多个值，并且就算方法中引用多了也不会报错。

****扩展运算符的用处：****

我们先用一个例子说明，我们声明两个数组arr1和arr2，然后我们把arr1赋值给arr2，然后我们改变arr2的值，你会发现arr1的值也改变了，因为我们这是对内存堆栈的引用，而不是真正的赋值。

|  |
| --- |
| let arr1=['www','DeLyQiang','com'];  let arr2=arr1;  console.log(arr2);  arr2.push('shengHongYu');  console.log(arr1); |

****控制台输出：****

|  |
| --- |
| ["www", "DeLyQiang", "com"]  ["www", "DeLyQiang", "com", "shengHongYu"] |

这是我们不想看到的，可以利用对象扩展运算符简单的解决这个问题，现在我们对代码进行改造。

|  |
| --- |
| let arr1=['www','DeLyQiang','com'];  //let arr2=arr1;  let arr2=[...arr1];  console.log(arr2);  arr2.push('shengHongYu');  console.log(arr2);  console.log(arr1); |

现在控制台预览时，你可以看到我们的arr1并没有改变，简单的扩展运算符就解决了这个问题。

### **4.2rest运算符**

如果你已经很好的掌握了对象扩展运算符，那么理解rest运算符并不困难，它们有很多相似之处，甚至很多时候你不用特意去区分。它也用…（三个点）来表示，我们先来看一个例子。

|  |
| --- |
| function DeLyQiang(first,...arg){      console.log(arg.length);  }    DeLyQiang(0,1,2,3,4,5,6,7); |

这时候控制台打印出了7，说明我们arg里有7个数组元素，这就是rest运算符的最简单用法。

如何循环输出rest运算符

这里我们用for…of循环来进行打印出arg的值，我们这里只是简单使用一下，以后我们会专门讲解for…of循环。

|  |
| --- |
| function DeLyQiang(first,...arg){      for(let val of arg){          console.log(val);      }  }  DeLyQiang(0,1,2,3,4,5,6,7); |

for…of的循环可以避免我们开拓内存空间，增加代码运行效率，所以建议大家在以后的工作中使用for…of循环。有的小伙伴会说了，反正最后要转换成ES5，没有什么差别，但是至少从代码量上我们少打了一些单词，这就是开发效率的提高。

总结：我们这节课学习了对象扩展运算符和reet运算符，它们两个还是非常类似的，但是你要自己区分，这样才能在工作中运用自如。在以后的课程中还会有很多关于扩展运算符和rset运算符的妙用，让我们一起期待吧。

# **第5节：字符串模版**

这节是ES6对字符串新增的操作，最重要的就是字符串模版，字符串模版的出现让我们再也不用拼接变量了，而且支持在模板里有简单计算操作

### **5.1字符串模版**

先来看一个在ES5下我们的字符串拼接案例:

|  |
| --- |
| let DeLyQiang='戴利强';  let blog = '非常高兴你能看到这篇文章，我是你的老朋友'+DeLyQiang+'。这节课我们学习字符串模版。';  document.write(blog); |

ES5下必须用****+DeLyQiang+****这样的形式进行拼接，这样很麻烦而且很容易出错。ES6新增了字符串模版，可以很好的解决这个问题。字符串模版不再使用****‘xxx’****这样的单引号，而是换成了**xxx**这种形式，也叫连接号。这时我们再引用DeLyQiang变量就需要用${DeLyQiang}这种形式了，我们对上边的代码进行改造。

|  |
| --- |
| let DeLyQiang='戴利强';  let blog = `非常高兴你能看到这篇文章，我是你的老朋友${DeLyQiang}。这节课我们学习字符串模版。`;  document.write(blog); |

可以看到浏览器出现了和上边代码一样的结果。而且这里边支持****html标签****，可以试着输入一些。

|  |
| --- |
| let DeLyQiang='戴利强';  let blog = `<b>非常高兴你能看到这篇文章</b>，我是你的老朋友${DeLyQiang}。<br/>这节课我们学习字符串模版。`;  document.write(blog); |

****对运算的支持：****

|  |
| --- |
| let a=1;  let b=2;  let result=`${a+b}`;  document.write(result); |

强大的字符串模版，在实际开发中，我们可以让后台写一个活动页面，然后轻松的输出给用户。

### **5.2字符串查找**

ES6还增加了字符串的查找功能，而且支持中文哦，

****查找是否存在:****

先来看一下ES5的写法，其实这种方法并不实用，给我们的索引位置，我们自己还要确定位置。

|  |
| --- |
| let DeLyQiang='戴利强';  let blog = '非常高兴你能看到这篇文章，我是你的老朋友戴利强。这节课我们学习字符串模版。';  document.write(blog.indexOf(DeLyQiang)); |

这是网页中输出了20，我们还要自己判断。

ES6直接用includes就可以判断，不再返回索引值，这样的结果我们更喜欢，更直接。

|  |
| --- |
| let DeLyQiang='戴利强';  let blog = '非常高兴你能看到这篇文章，我是你的老朋友戴利强。这节课我们学习字符串模版。';  document.write(blog.includes(DeLyQiang)); |

****判断开头是否存在：****

|  |
| --- |
| blog.startsWith(DeLyQiang); |

****判断结尾是否存在：****

|  |
| --- |
| blog.endsWith(DeLyQiang); |

需要注意的是：starts和ends 后边都要加s，我开始时经常写错，希望小伙伴们不要采坑。

### **5.3复制字符串**

我们有时候是需要字符串重复的，比如分隔符和特殊符号，这时候复制字符串就派上用场了，语法很简单。

|  |
| --- |
| document.write('DeLyQiang|'.repeat(3)); |

当然ES6对字符串还有一些其它操作，因为实际工作中不太使用，这里就不作太多的介绍了。希望你能动手练习一下，并把这些新特性应用到工作中，否则可能很快就忘记了。

# **第6节：ES6数字操作**

### **6.1二进制和八进制**

二进制和八进制数字的声明并不是ES6的特性，我们只是做一个常识性的回顾，因为很多新人小伙伴会把他们当成字符串或者不知道是什么，所以这算是赠送的知识点。

****二进制声明：****

二进制的英文单词是Binary,二进制的开始是0（零），然后第二个位置是b（注意这里大小写都可以实现），然后跟上二进制的值就可以了。

let binary = 0B10101;

console.log(binary)

这时候浏览器的控制台显示出了21。

****八进制声明：****

八进制的英文单词是Octal，也是以0（零）开始的，然后第二个位置是O（欧），然后跟上八进制的值就可以了。

let octal=0o666

console.log(octal)

这时候浏览器的控制台显示出了438。

### **6.2数字判断和转换**

****数字验证Number.isFinite( xx )****

可以使用Number.isFinite( )来进行数字验证，只要是数字，不论是浮点型还是整形都会返回true，其他时候会返回false。

let a=11/4

console.log(Number.isFinite(a)) //true

console.log(Number.isFinite('DeLyQiang')) //false

console.log(Number.isFinite(NaN)) //false

console.log(Number.isFinite(undefined)) //false

****NaN验证****

NaN是特殊的非数字，可以使用Number.isNaN()来进行验证。下边的代码控制台返回了true。

console.log(Number.isNaN(NaN))

****判断是否为整数Number.isInteger(xx)****

let a=985.1

console.log(Number.isInteger(a))

****整数转换Number.parseInt(xxx)和浮点型转换Number.parseFloat(xxx)****

let a=985.1

console.log(Number.isInteger(a))

console.log(Number.parseInt(a))

console.log(Number.parseFloat(a))

### **6.3整数取值范围操作**

整数的操作是有一个取值范围的，它的取值范围就是2的53次方。我们先用程序来看一下这个数字是什么.

let DeLyQiang = Math.pow(2,53)-1;

console.log(DeLyQiang) //9007199254740991

在我们计算时会经常超出这个值，所以我们要进行判断，ES6提供了一个常数，叫做最大安全整数，以后就不需要我们计算了。

****最大安全整数****

console.log(Number.MAX\_SAFE\_INTEGER)

****最小安全整数****

console.log(Number.MIN\_SAFE\_INTEGER)

****安全整数判断isSafeInteger( )****

let DeLyQiang = Math.pow(2,53)-1;  
console.log(Number.isSafeInteger(DeLyQiang))

总结：这节课我们学习了ES6数字的操作，方法很多，很零散，需要经常复习或者实战中慢慢熟悉。

# **第7节：ES6中新增的数组知识（1）**

**7.1JSON数组格式转换**

JSON的数组格式就是为了前端快速的把JSON转换成数组的一种格式，我们先来看一下JSON的数组格式怎么写。

let json = {

'0': 'DeLyQiang',

'1': '戴利强',

'2': '大胖逼逼叨',

length:3

}

这就是一个标准的JSON数组格式，跟普通的JSON对比是在最后多了一个length属性。只要是这种特殊的json格式都可以轻松使用ES6的语法转变成数组。在ES6中绝大部分的Array操作都存在于Array对象里。我们就用Array.from(xxx)来进行转换。我们把上边的JSON代码转换成数组，并打印在控制台。

let json = {

'0': 'DeLyQiang',

'1': '戴利强',

'2': '大胖逼逼叨',

length:3

}

let arr=Array.from(json);

console.log(arr)

实际开发中这种方法还是比较常用的，毕竟节省了我们代码行数，也让我们的程序更清晰。

### **7.2Array.of()方法：**

它负责把一堆文本或者变量转换成数组。在开发中我们经常拿到了一个类似数组的字符串，需要使用eval来进行转换，如果你一个老手程序员都知道eval的效率是很低的，它会拖慢我们的程序。这时候我们就可以使用Array.of方法。我们看下边的代码把一堆数字转换成数组并打印在控制台上：

let arr =Array.of(3,4,2,5)

console.log(arr)

当然它不仅可以转换数字，字符串也是可以转换的，看下边的代码：

let a='sdf'

let arr =Array.of(3,4,a,'戴利强')

console.log(arr)

### **7.3find( )实例方法:**

所谓的实例方法就是并不是以Array对象开始的，而是必须有一个已经存在的数组，然后使用的方法，这就是实例方法（不理解请看下边的代码，再和上边的代码进行比对，你会有所顿悟）。这里的find方法是从数组中查找。在find方法中我们需要传入一个匿名函数，函数需要传入三个参数：

* value：表示当前查找的值。
* index：表示当前查找的数组索引。
* arr：表示当前数组。

在函数中如果找到符合条件的数组元素就进行return，并停止查找。你可以拷贝下边的代码进行测试，就会知道find作用。

let arr=[1,2,3,4,5,6,7,8,9];

console.log(arr.find(function(value,index,arr){

return value>5;

}))

控制台输出了6，说明找到了符合条件的值，并进行返回了，如果找不到会显示undefined。

# **第8节：ES6中新增的数组知识（2）**

### **8.1fill( )实例方法：**

fill()也是一个实例方法，它的作用是把数组进行填充，它接收三个参数，第一个参数是填充的变量，第二个是开始填充的位置，第三个是填充到的位置。

let arr=[0,1,2,3,4,5,6,7,8,9];

arr.fill('DeLyQiang',2,5);

console.log(arr);

上边的代码是把数组从第二位到第五位用DeLyQiang进行填充。

浏览器console打印如下：

**[0, 1, "DeLyQiang", "DeLyQiang", "DeLyQiang", 5, 6, 7, 8, 9]**

### **8.2数组的遍历**

****for…of循环：****

这种形式比ES5的for循环要简单而且高效。先来看一个最简单的for…of循环。

let arr=['DeLyQiang','戴利强','大胖逼逼叨']

for (let item of arr){

console.log(item);

}

****for…of数组索引****:有时候开发中是需要数组的索引的，那我们可以使用下面的代码输出数组索引。

let arr=['DeLyQiang','戴利强','大胖逼逼叨']

for (let index of arr.keys){

console.log(index);

}

可以看到这时的控制台就输出了0,1,2，也就是数组的索引。

****同时输出数组的内容和索引****：我们用entries()这个实例方法，配合我们的for…of循环就可以同时输出内容和索引了。

let arr=['DeLyQiang','戴利强','大胖逼逼叨']

for (let [index,val] of arr.entries()){

console.log(index+':'+val);

}

### **8.3entries( )实例方法：**

entries()实例方式生成的是Iterator形式的数组，那这种形式的好处就是可以让我们在需要时用next()手动跳转到下一个值。我们来看下面的代码：

let arr=['DeLyQiang','戴利强','大胖逼逼叨']

let list=arr.entries();

console.log(list.next().value);

console.log(list.next().value);

console.log(list.next().value);

总结：这两节是ES6对数组的扩展，数组在我们的实际开发中是特别重要的，几乎每天都要编写数组的操作代码

# **第9节：ES6中的箭头函数和扩展**

这节开始，先不着急看ES6中的函数，而是回顾一下ES5中的函数写法。写一个函数，进行一个加法计算。

function add(a,b){

    return a+b;

}

console.log(add(1,2));

我们声明了一个add函数，然后传入a和b两个值，返回a+b的值。 然后我们在控制台打印了这个函数的返回结果，这里是3.

### **9.1默认值**

在ES6中给我们增加了默认值的操作，我们修改上边的代码，可以看到现在只需要传递一个参数也是可以正常运行的。

function add(a,b=1){

    return a+b;

}

console.log(add(1));

### **9.2主动抛出错误**

在使用Vue的框架中，可以经常看到框架主动抛出一些错误，比如v-for必须有:key值。那尤大神是如何做到的那？其实很简单，ES6中我们直接用throw new Error( xxxx ),就可以抛出错误。

function add(a,b=1){

    if(a == 0){

        throw new Error('This is error')

    }

     return a+b;

}

console.log(add(0));

### **9.3函数中的严谨模式**

我们在ES中就经常使用严谨模式来进行编程，但是必须写在代码最上边，相当于全局使用。在ES6中我们可以写在函数体中，相当于针对函数来使用。

function add(a,b=1){

    'use strict'

    if(a == 0){

        throw new Error('This is error');

    }

     return a+b;

}

console.log(add(1));

上边的代码如果运行的话，你会发现浏览器控制台报错，这是ES6中的一个坑，如果没人指导的话，可能你会陷进去一会。这个错误的原因就是如果你使用了默认值，再使用严谨模式的话，就会有冲突，所以我们要取消默认值的操作，这时候你在运行就正常了。

function add(a,b){

    'use strict'

    if(a == 0){

        throw new Error('This is error');

    }

     return a+b;

}

console.log(add(1,2));

### **9.4获得需要传递的参数个数**

如果你在使用别人的框架时，不知道别人的函数需要传递几个参数怎么办？ES6为我们提供了得到参数的方法(xxx.length).我们用上边的代码看一下需要传递的参数个数。

function add(a,b){

'use strict'

if(a == 0){

throw new Error('This is error');

}

return a+b;

}

console.log(add.length);

这时控制台打印出了2，但是如果去掉严谨模式，并给第二个参数加上默认值的话，这时候add.length的值就变成了1， 也就是说它得到的是必须传入的参数。

### **9.5箭头函数**

在学习Vue的时候，大量的使用了箭头函数，因为箭头函数真的很好用，我们来看一个最简单的箭头函数。也就是上边我们写的add函数，进行一个改变，写成箭头函数。

var add =(a,b=1) => a+b;

console.log(add(1))

****{}的使用****

在箭头函数中，方法体内如果是两句话，那就需要在方法体外边加上{}括号。例如下边的代码就必须使用{}.

var add= (a,b=1) => {

console.log('DeLyQiang')

return a+b;

};

console.log(add(1));

箭头函数中不可加new，也就是说箭头函数不能当构造函数进行使用。

总结： 写新项目的时候尽量使用ES6带来的便利和特点，这样才能快速成长。

# **第10节：ES6中的函数和数组补漏**

### **10.1对象的函数解构**

我们在前后端分离时，后端经常返回来JSON格式的数据，前端的美好愿望是直接把这个JSON格式数据当作参数，传递到函数内部进行处理。ES6就为我们提供了这样的解构赋值。

let json = {

a:'DeLyQiang',

b:'戴利强'

}

function fun({a,b='DeLyQiang'}){

console.log(a,b);

}

fun(json)

是不是感觉方便了很多，我们再也不用一个个传递参数了。

****数组的函数解构****

函数能解构JSON，那解构我们的数组就更不在话下了，我们看下边的代码。我们声明一个数组，然后写一个方法，最后用…进行解构赋值。

let arr = ['DeLyQiang','戴利强','ECMScript'];

function fun(a,b,c){

console.log(a,b,c);

}

fun(...arr)

这种方法其实在前面的课程中已经学过了，这里我们就当复习了。

### **10.2in的用法**

in是用来判断对象或者数组中是否存在某个值的。我们先来看一下用in如何判断对象里是否有某个值。

****对象判断****

let obj={

a:'DeLyQiang',

b:'戴利强'

}

console.log(‘a’ in obj)

****数组判断****

先来看一下ES5判断的弊端，以前会使用length属性进行判断，为0表示没有数组元素。但是这并不准确，或者说真实开发中有弊端。

let arr=[,,,,,];

console.log(arr.length); //5

上边的代码输出了5，但是数组中其实全是空值，这就是一个坑啊。那用ES6的in就可以解决这个问题。

let arr=[,,,,,];

console.log(0 in arr); //false

let arr1 =['DeLyQiang','戴利强'];

console.log(0 in arr1) //true

注意：这里的0指的是数组下标位置是否为空。

### **10.3数组的遍历方法**

1.forEach

let arr1 =['DeLyQiang','戴利强','ECMScript',,];

arr1.forEach((val,index)=>console.log(index,val));

forEach循环的特点是会自动省略为空的数组元素，相当于直接给我们筛空了。当是有时候也会给我们帮倒忙。

1. Filter

let arr1 =['DeLyQiang','戴利强','ECMScript',,];

arr1.filter(x=>console.log(x))

这种方法在Vue用过，他其实也有循环的功能

3.some

let arr1 =['DeLyQiang','戴利强','ECMScript',,];

arr1.some(x=>console.log(x))

4.map

let arr1 =['DeLyQiang','戴利强','ECMScript',,];

console.log(arr1.map(x=>'web'))



map在这里起到一个替换的作用，这个后续会详细讲解。

### **10.4数组转换字符串**

在开发中我们经常会碰到把数组输出成字符串的形式，我们今天学两种方法，你要注意两种方法的区别。

join()方法

let arr1 =['DeLyQiang','戴利强','ECMScript',,];

console.log(arr1.join('|'));

join()方法就是在数组元素中间，加了一些间隔，开发中很有用处。

toString()方法

let arr1 =['DeLyQiang','戴利强','ECMScript',,];

console.log(arr1.toString());

转换时只是是用逗号隔开了。

# **第11节：ES6中对象**

对象对于Javascript是非常重要的。在ES6中对象有了很多新特性，

### **11.1对象赋值**

ES6允许把声明的变量直接赋值给对象，我们看下面的例子。

let name='DeLyQiang';

let skill='web';

var obj={name,skill};

console.log(obj); //Object {name: "DeLyQiang", skill: "web"}

### **11.2对象Key值构建**

有时候我们会在后台取出key值，而不是我们前台定义好的，这时候我们如何构建我们的key值那。比如我们在后台取了一个key值，然后可以用[ ] 的形式，进行对象的构建。

let key='skill';

var obj={

[key]:'web'

};

console.log(obj.skill);

### **11.3自定义对象方法**

对象方法就是把兑现中的属性，用匿名函数的形式编程方法。这个在以前就有应用，这里只是简单的复习一下。

var obj={

add:function(a,b){

return a+b;

}

}

console.log(obj.add(1,2)); //3

### **11.4Object.is(  ) 对象比较**

对象的比较方法,以前进行对象值的比较，经常使用===来判断，比如下面的代码：

var obj1 ={name:'DeLyQiang'};

var obj2 ={name:'DeLyQiang'};

console.log(obj1.name === obj2.name); //true

那ES6为我们提供了is方法进行对比。

var obj1 ={name:'DeLyQiang'};

var obj2 ={name:'DeLyQiang'};

console.log(obj1.name === obj2.name); //true

console.log(Object.is(obj1.name, obj2.name)); //true

区分=== 和 is方法的区别是什么，看下面的代码输出结果。

console.log(+0 === -0); //true

console.log(NaN === NaN); //false

console.log(Object.is(+0, -1)); //false

console.log(Object.is(NaN, NaN)); //true

此处，===为同值相等，is()为严格相等。

### **11.4Object.assign(  )合并对象**

操作数组时，经常使用数组合并，那对象也有合并方法，那就是assgin(  )。看一下啊具体的用法。

var a={a:'DeLyQiang'};

var b={b:'戴利强'};

var c={c:'es6'};

let d=Object.assign(a,b,c);

console.log(d)

基础知识的点是非常杂的，应多练习

# **第12节：Symbol在对象中的作用**

通过场景应用的方式学习Symbol，它的意思是全局标记。先看看它的声明方式。

### **12.1声明Symbol**

我们先来回顾一下我们的数据类型，在最后在看看Symbol如何声明，并进行一个数据类型的判断。

var a = new String;

var b = new Number;

var c = new Boolean;

var d = new Array;

var e = new Object;

var f= Symbol();

console.log(typeof(d));

console.log(typeof(f));

### **12.2Symbol的打印**

我们先声明一个Symbol，然后我们在控制台输出一下。

var g = Symbol('DeLyQiang');

console.log(g);

console.log(g.toString());

这时候我们仔细看控制台是有区别的，没有toString的是红字，toString的是黑字。

### **12.3Symbol在对象中的应用**

看一下如何用Symbol构建对象的Key，并调用和赋值。

var DeLyQiang = Symbol();

var obj={

[DeLyQiang]:'戴利强'

}

console.log(obj[DeLyQiang]);

obj[DeLyQiang] ='web';

console.log(obj[DeLyQiang])

### **12.4Symbol对象元素的保护作用**

在对象中有很多值，但是循环输出时，并不希望全部输出，那我们就可以使用Symbol进行保护。

没有进行保护的写法：

var obj={name:'DeLyQiang',skill:'web',age:18};

let age=Symbol();

obj[age]=18;

for (let item in obj){

console.log(obj[item]);

}

console.log(obj)

现在我不想别人知道我的年龄，这时候我就可以使用Symbol来进行循环保护。

var obj={name:'DeLyQiang',skill:'web'};

let age=Symbol();

obj[age]=18;

for (let item in obj){

console.log(obj[item]);

}

console.log(obj)

# **第13节：Set和WeakSet数据结构**

这节学习Set数据结构，注意这里不是数据类型，而是****数据结构****。它是ES6中新的东西，并且很有用处。Set的数据结构是以数组的形式构建的。

### **13.1Set的声明**

let setArr = new Set(['DeLyQiang','戴利强','web','DeLyQiang']);

console.log(setArr) //{"DeLyQiang", "戴利强", "web"}

Set和Array 的区别是Set不允许内部有重复的值，如果有只显示一个，相当于去重。虽然Set很像数组，但是他不是数组。

### **13.2Set值的增删查**

****追加add：****

在使用Array的时候，可以用push进行追加值，那Set稍有不同，它用更语义化的add进行追加。

let setArr = new Set(['DeLyQiang','戴利强','web','DeLyQiang']);

console.log(setArr) //{"DeLyQiang", "戴利强", "web"}

setArr.add('前端职场')

console.log(setArr)

****删除delete：****

let setArr = new Set(['DeLyQiang','戴利强','web','DeLyQiang']);

console.log(setArr) //{"DeLyQiang", "戴利强", "web"}

setArr.add('前端职场')

console.log(setArr) //{"DeLyQiang", "戴利强", "web","前端职场"}

setArr.delete('前端职场')

console.log(setArr) //{"DeLyQiang", "戴利强", "web"}

****查找has：****

用has进行值的查找，返回的是true或者false。

let setArr = new Set(['DeLyQiang','戴利强','web','DeLyQiang']);

console.log(setArr) //{"DeLyQiang", "戴利强", "web"}

console.log(setArr.has('DeLyQiang')); //true

****删除clear:****

let setArr = new Set(['DeLyQiang','戴利强','web','DeLyQiang']);

console.log(setArr) //{"DeLyQiang", "戴利强", "web"}

setArr.clear()

console.log(setArr); //true

### **13.3set的循环**

****for…of…循环：****

let setArr = new Set(['DeLyQiang','戴利强','web','DeLyQiang']);

for (let item of setArr){

console.log(item)

}

****size属性****

size属性可以获得Set值的数量。

let setArr = new Set(['DeLyQiang','戴利强','web','DeLyQiang']);

for (let item of setArr){

console.log(item)

}

console.log(setArr.size)

****forEach循环****

let setArr = new Set(['DeLyQiang','戴利强','web','DeLyQiang']);

setArr.forEach((value) => console.log(value))

### **13.4WeakSet的声明**

let weakObj=new WeakSet();

let obj={a:'DeLyQiang',b:'戴利强'}

weakObj.add(obj);

console.log(weakObj)

这里需要注意的是，如果你直接在new 的时候就放入值，将报错。

WeakSet里边的值也是不允许重复的，我们来测试一下。

let weakObj=new WeakSet();

let obj={a:'DeLyQiang',b:'戴利强'}

let obj1=obj;

weakObj.add(obj);

weakObj.add(obj1);

console.log(weakObj)

总结：在实际开发中Set用的比较多，WeakSet用的并不多，但是他对传入值必须是对象作了很好的判断，灵活应用还是有一定的用处的。

# **第14节：map数据结构**

这节主要学习map的这种ES6新加的数据结构。在一些构建工具中是非常喜欢使用map这种数据结构来进行配置的，因为map是一种灵活，简单的适合一对一查找的数据结构。我们知道的数据结构，已经有了json和set。那map有什么特点。

### **14.1Json和map格式的对比**

map的效率和灵活性更好

先来写一个JSON，这里我们用对象进行模拟操作。

let json = {

name:'DeLyQiang',

skill:'web'

}

console.log(json.name)

但是这种反应的速度要低于数组和map结构。而且Map的灵活性要更好，你可以把它看成一种特殊的键值对，但你的key可以设置成数组，值也可以设置成字符串，让它不规律对应起来。

let json = {

name:'DeLyQiang',

skill:'web'

}

console.log(json.name)

var map=new Map();

map.set(json,'iam');

console.log(map)



当然也可key字符串，value是对象。我们调换一下位置，依然是符合map的数据结构规范的。

let json = {

name:'DeLyQiang',

skill:'web'

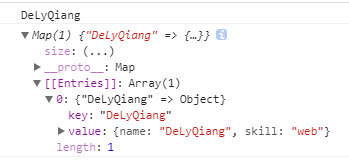
}

console.log(json.name)

var map=new Map();

map.set('DeLyQiang',json);

console.log(map)



### **14.2map的增删查**

上边我们已经会为map增加值了，就是用我们的set方法，这里我就不重复讲解了。直接看如何取出我们的值。

****取值get****

现在取json对应的值。(此处有错误)

console.log(map.get(json))

****删除delete****

删除delete的特定值：

map.delete(json)

console.log(map)

****size属性****

console.log(map.size)

****查找是否存在has****

console.log(map.has('DeLyQiang'))

****清楚所有元素clear****

map.clear();

总结：map在现在开发中已经经常使用，它的灵活性和高效性是我们喜欢的。开发中试着去使用map吧，你一定会喜欢上它的。

# **第15节：用Proxy进行预处理**

如果接触过Vue，一定会知道钩子函数，当我们在操作一个对象或者方法时会有几种动作，比如：在运行函数前初始化一些数据，在改变对象值后做一些善后处理。这些都算钩子函数，Proxy的存在就可以让我们给函数加上这样的钩子函数，你也可以理解为在执行方法前预处理一些代码。你可以简单的理解为他是函数或者对象的生命周期。

Proxy的应用可以使函数更加强大，业务逻辑更加清楚，而且在编写自己的框架或者通用组件时非常好用。Proxy涉及的内容非常多

先来回顾一下定义对象的方法。

var obj={

add:function(val){

return val+10;

},

name: 'I am DeLyQiang'

};

console.log(obj.add(100));

console.log(obj.name);

声明了一个obj对象，增加了一个对象方法add和一个对象属性name，然后在控制台进行了打印。

### **15.1声明Proxy**

我们用new的方法对Proxy进行声明。可以看一下声明Proxy的基本形式。

new Proxy({},{});

需要注意的是这里是两个花括号，第一个花括号就相当于我们方法的主体，后边的花括号就是Proxy代理处理区域，相当于我们写钩子函数的地方。

现在把上边的obj对象改成我们的Proxy形式。

var pro = new Proxy({

add:function(val){

return val+10;

},

name: 'I am DeLyQiang'

},{

get:function(target,key,propert){

console.log('come in Get');

return target[key];

}

});

console.log(pro.name)

可以在控制台看到结果，先输出了come in Get。相当于在方法调用前的钩子函数。

****get属性****

get属性是在你得到某对象属性值时预处理的方法，他接受三个参数

* target：得到的目标值
* key：目标的key值，相当于对象的属性
* property：这个不太常用，用法还在研究中，还请大神指教。

****set属性****

set属性是值你要改变Proxy属性值时，进行的预先处理。它接收四个参数。

* target:目标值。
* key：目标的Key值。
* value：要改变的值。
* receiver：改变前的原始值。

var pro = new Proxy({

add:function(val){

return val+10;

},

name: 'I am DeLyQiang'

},{

get:function(target,key){

console.log('come in Get');

return target[key];

},

set:function(target,key,value,receiver){

console.log(' setting ${key}= ${value}');

return target[key] = value;

}

});

console.log(pro.name)

pro.name='戴利强';

console.log(pro.name)

### **15.2apply的使用**

apply的作用是调用内部的方法，它使用在方法体是一个匿名函数时。看下边的代码。

let target = function () {

return 'I am DeLyQiang';

};

var hander = {

apply(target, ctx, args) {

console.log('do apply');

return Reflect.apply(...arguments);

}

}

var pro = new Proxy(target, hander);

console.log(pro());

其实proxy的知识是非常多的，这里我建议看阮一峰大神的《ES6》。我这里只能算是入门课程，俗话说得好：“师傅领进门，修行靠个人”，那我们下节课见了。

# **第16节：promise对象的使用**

ES6中的promise的出现给我们很好的解决了回调地狱的问题，在使用ES5的时候，在多层嵌套回调时，写完的代码层次过多，很难进行维护和二次开发，ES6认识到了这点问题，现在promise的使用，完美解决了这个问题。那我们如何理解promise这个单词在ES5中的作用那，你可以想象他是一种承诺，当它成功时执行一些代码，当它失败时执行一些代码。它更符合人类的行为思考习惯，而不在是晦涩难懂的冰冷语言。

### **promise的基本用法**

promise执行多步操作非常好用，那我们就来模仿一个多步操作的过程，那就以吃饭为例吧。要想在家吃顿饭，是要经过三个步骤的。

1. 洗菜做饭。
2. 坐下来吃饭。
3. 收拾桌子洗碗。

这个过程是有一定的顺序的，你必须保证上一步完成，才能顺利进行下一步。我们可以在脑海里先想想这样一个简单的过程在ES5写起来就要有多层的嵌套。那我们现在用promise来实现。

let state=1;

function step1(resolve,reject){

console.log('1.开始-洗菜做饭');

if(state==1){

resolve('洗菜做饭--完成');

}else{

reject('洗菜做饭--出错');

}

}

function step2(resolve,reject){

console.log('2.开始--坐下吃饭')

if(state==1){

resolve('坐下来吃饭--完成');

}else{

reject('坐下来吃饭--出错');

}

}

// function step3(resolve,reject){

// console.log('3.开始.收拾桌子洗完');

// if(state==1){

// resolve('收拾桌子洗完--完成');

// }else{

// reject('收拾桌子洗完--出错');

// }

// }

let step3=(resolve,reject) =>{

console.log('3.开始.收拾桌子洗完');

if(state==1){

resolve('收拾桌子洗完--完成');

}else{

reject('收拾桌子洗完--出错');

}

}

new Promise(step1).then(function(val){

console.log(val)

return new Promise(step2)

}).then(function(val){

console.log(val)

return new Promise(step3);

}).then(function(val){

console.log(val);

return val;

})

Promis在现在的开发中使用率算是最高的，而且你面试前端都会考这个对象，大家一定要掌握好。

# **第17节：class类的使用**

我们在ES5中经常使用方法或者对象去模拟类的使用，虽然可以实现功能，但是代码并不优雅，ES6为我们提供了类的使用。需要注意的是我们在写类的时候和ES5中的对象和构造函数要区分开来，不要学混了。

### **17.1类的声明**

先声明一个最简单的coder类，类里只有一个name方法，方法中打印出传递的参数。

class Coder{

name(val){

console.log(val)

return val;

}

}

### **17.2类的使用**

我们已经声明了一个类，并在类里声明了name方法，现在要实例化类，并使用类中的方法。

class Coder{

name(val){

console.log(val)

return val;

}

}

let DeLyQiang= new Coder;

DeLyQiang.name('DeLyQiang');

### **17.3类的多方法声明**

class Coder{

name(val){

console.log(val)

return val;

}

skill(val){

console.log(this.name('DeLyQiang')+':'+'Skill'+val);

}

}

let DeLyQiang= new Coder;

DeLyQiang.name('DeLyQiang');

DeLyQiang.skill('web');

这里需要注意的是两个方法中间不要写逗号了，还有这里的this指类本身，还有要注意return 的用法。

### **17.4类的传参**

在类的参数传递中我们用constructor( )进行传参。传递参数后可以直接使用this.xxx进行调用。

class Coder{

name(val){

console.log(val)

return val;

}

skill(val){

console.log(this.name('DeLyQiang')+':'+'Skill'+val);

}

constructor(a,b){

this.a=a;

this.b=b;

}

add(){

return this.a+this.b;

}

}

let DeLyQiang= new Coder(1,2);

console.log(DeLyQiang.add());

我们用constructor来约定了传递参数，然后用作了一个add方法，把参数相加。这和以前我们的传递方法有些不一样，所以需要小伙伴们多注意下。

### **17.5class的继承**

如果你学过java，一定知道类的一大特点就是继承。ES6中也就继承，在这里我们简单的看看继承的用法。

let dai= new htmler;

dai.name('戴利强')

声明一个htmler的新类并继承Coder类，htmler新类里边为空，这时候我们实例化新类，并调用里边的name方法。结果也是可以调用到的。

# **第18节：模块化操作**

在ES5中我们要进行模块华操作需要引入第三方类库，随着前后端分离，前端的业务日渐复杂，ES6为我们增加了模块话操作。模块化操作主要包括两个方面。

* export :负责进行模块化，也是模块的输出。
* import : 负责把模块引，也是模块的引入操作。

### **18.1export的用法：**

export可以让我们把变量，函数，对象进行模块话，提供外部调用接口，让外部进行引用。先来看个最简单的例子，把一个变量模块化。我们新建一个temp.js文件，然后在文件中输出一个模块变量。

export var name = 'DeLyQiang';

然后可以在index.js中以import的形式引入。

import {name} from './temp';

console.log(name);

这就是一个最简单的模块的输出和引入。

控制台：babel-node index.js



### ****18.2多变量的输出****

这里声明了3个变量，需要把这3个变量都进行模块化输出，这时候我们给他们包装成对象就可以了。

import {a,b,c} from './temp';

console.log(a,b,c);

### ****18.3函数的模块化输出****

export function add(a,b){

return a+b

}

### ****18.4as的用法****

有些时候我们并不想暴露模块里边的变量名称，而给模块起一个更语义话的名称，这时候我们就可以使用as来操作。

var a ='DeLyQiang';

var b ='戴利强';

var c = 'web';

export {

    x as a,

    y as b,

    z as c

}

### **18.5export default的使用**

加上default相当是一个默认的入口。在一个文件里export default只能有一个。我们来对比一下export和export   default的区别

****1.export****

var a = 'DeLyQiang';

export function add(a,b){

return a+b

}

对应的导入方式

import {a,add} from './temp'; //也可以分开写

2.export defalut

export default var a='DeLyQiang'

对应的引入方式（报错）

import sss from './temp';

console.log(sss);

ES6的模块化不能直接在浏览器中预览，必须要使用Babel进行编译之后正常看到结果。这节课讲完我们ES6的课程就算结束了，你可能觉的没有书上的内容多，那是因为很多东西都归到了ES7中。甚至连Babel都不能很好的转换，这些知识不讲解了。另外如果你想继续深入学习，可以搜索阮一峰大神的ES6在线图书