ES6

http://jspang.com/2017/06/03/es6/

1. es6的开发环境

现在的Chrome浏览器已经支持ES6了，但是有些低版本的浏览器还是不支持ES6的语法，这就需要我们把ES6的语法自动的转变成ES5的语法，大家知道webpack等工具可以自动编译，我们也可以用babel-cli来完成ES6编译成ES5

（做项目之前先初始化下package.json配置包

Npm init –y//给一个项目生成一个package的包 –y的意思就一遍过不用手动去敲）

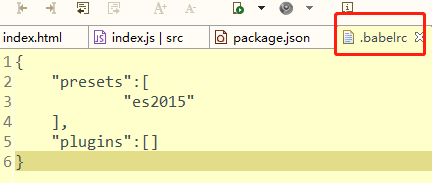
Npm install –g babel-cli//全局安装下

安装完这个后还是不能直接编译的还需要安装如下

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

(--save是运行依赖 –save-dev是开发时依赖（上线就不需要的那种） 都会加到package.json里面)

接下来需要在根目录下建立一个.babelrc的文件 里面的内容如下

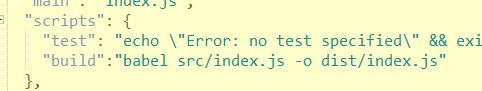


完成以上几步之后 只需要想把哪个es6 文件转成哪个位置的es5就可以了



这是最后的执行代码

（每次都需要执行上边那行代码也是比较麻烦 vue中直接可以npm run build就编译打包了 这里 我们可以把package.json改一下也能用npm运行）



Npm run build 相当于直接运行babel src/index.js –o dist/index.js

1. 变量的声明

// var：它是variable的简写，可以理解成变量的意思。

// let：它在英文中是“让”的意思，也可以理解为一种声明的意思。

// 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。

var a = 'jspang';

{

let a = 'qiufuli'; // let 是一个局部声明 在一段代码块结束后就消失了 循环中常用

}

console.log(a)

const b= '123'; // const 是一个常量声明 后期如果变得话 会报错 如果不想被重复引用的话可以用这个

var b= '222';

console.log(b)

1. 变量的解构赋值

从数组和对象中提取值，对变量进行赋值，这被称为解构

//数组的解构赋值

let [a,b,c] = [0,1,2];

console.log(a,b,c);

//对象解构赋值

let {ao,bo} = {ao:'qiufuli',bo:'邱福利'};

let {a,b=’qiu’} ={a:’web’} //可以有默认值

console.log(ao,bo);

// 如果先声明了 那么解构赋值的时候需要加上（） 否则会报错

let co;

({co} = {co:’qiufuli’})

//字符串解构

const [q,w,e,r] = 'querqwer';

console.log(q,w,e,r)

//对象解构与数组解构的区别是 对象按照键值解构 数组需要按照顺序解构

1. …扩展运算符和rest运算符

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。

Function fun(…arg){

Console.log(arg[0],arg[1],arg[2]); // 1,2,3

}

Fun(1,2,3)

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。

扩展运算符就可以解决这个问题，记住只要带...xxx就是要xxx里面的东西

let arr1 = ['www', 'jspang', 'com'];

let arr2 = […arr1];

arr2.push('shengHongYu');

console.log(arr2);// ['www', 'jspang', 'com',’shengHongYu’];

console.log(arr1);// ['www', 'jspang', 'com'];

//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

function qiufuli(f,...arg){

//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

//…arg 是表示未知的参数 已有的不算在里面

for(let val of arg){

console.log(val) //1,2,3,4,5,6

}

}

qiufuli(0,1,2,3,4,5,6);

1. 字符串模版

波浪按键` ${变量在模板中的引用形势用这个}`

//查找字符串

let qiufuli='邱福利';

let blog = `我的名字叫${qiufuli}`;

console.log(blog.indexOf(qiufuli)); //es5中查找返回索引

console.log(blog.includes(qiufuli)); //es6中查找返回布尔

console.log(blog.startsWith(qiufuli)); // starts 多个s es6中查找开头有没有返回布尔

console.log(blog.endsWith(qiufuli)); // ends 多个s es6中查找结尾有没有返回布尔

1. 剪头函数

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

console.log(add(1,1))

or

var add = (a,b)=>{

return a+b;

}

console.log(add(1,1))

1. ES6中的函数和数组补漏

let arr = ['jspang','技术胖','免费教程'];

function fun(a,b,c){

console.log(a,b,c)

}

Fun(…arr) //利用了扩展运算传递参数 直接把arr里面的内容传过来了 很实用

In 的用法

Let obj={

a:’web’,

b:’qiufuli ’

}

Console.log(‘a’in obj) // true

let arr3=['jspang','技术胖'];

console.log(0 in arr3); // true 对于数组来说 可以是索引

数组的遍历

let arr4=['jspang','技术胖','前端教程'];

arr4.forEach((val,index)=>console.log(index,val)) // 这里 val index是反着的 切记

//filter

arr4.filter(x=>console.log(x));// 'jspang','技术胖','前端教程'

//some

arr4.some(x=>console.log(x));// 'jspang','技术胖','前端教程'

//map

console.log(arr4.map(x=>'web'));// web,web,web 相当于替换

1. es6中的对象

let name="jspang";

let skill = "web";

let obj={name,skill} //可以不用{name:name,skill:skill}的形式了

let key=’skill’;

可以对对象的key值构建

var obj2={

[key]:’web’ // skill:web

}

//Object.is()

console.log(obj3.name === obj4.name);//true

console.log(Object.is(obj3.name,obj4.name)); //true

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

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

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

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

===为同值相等，Object.is()为严格相等(就是说连表前上看都是一样的)。

Object.assign( )合并对象

var a={a:'jspang'};

var b={b:'技术胖'};

var c={c:'web'};

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

console.log(d);//输入a,b,c的合并对象



1. Set和WeakSet数据结构

Set的数据结构(不是数据类型)是以 数组 的形式构建的

主要的功能是去重

Let setArr = new Set([1,2,2,2,3,4,5]) //[1,2,3,4,5];

//追加add

setArr.add(‘web’);//[1,2,3,4,5,’web’]

//删除delete

setArr.delete(‘web’);

//查找has

Console.log(setArr.has(‘1’)); //true

//删除clear

setArr.clear();

//forEach循环

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

//size

console.log(setArr.size);

WeakSet的声明 因为set只能放数组 weakset能放对象

let weakObj=new WeakSet();

let obj={a:'jspang',b:'技术胖'}

weakObj.add(obj);

console.log(weakObj);

WeakSet也遵循Set的增删改查

//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，我们灵活应用还是有一定的用处的。

1. map数据结构

map的效率和灵活性更好 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:'jspang',

skill:'web'

}

Var map = new Map();

//增 set（）

Map.set(json,’iam’) // json是key iam是value 中间是=》 连接的 {name;’jspang’,skill:’web’} =>’iam’

//取值get()

console.log(map.get(json));

//删除delete的特定值：

map.delete(json);

console.log(map)

//清楚所有元素clear

console.log(map.clear())

//size属性

console.log(map.size);

//查找是否存在has

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

1. 用Proxy进行预处理

//proxy 代理 es6 增强 对象和函数（方法） 声明周期 预处理

var obj={

add:function(val){

return val+10;

},

name:'I am Jspang'

};

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

console.log(obj.name);

//我们用new的方法对Proxy进行声明。

var pro = new Proxy({

add:function(val){

return val+10;

},

name:'I am Jspang'

},{

//get属性是在你得到某对象属性值时预处理的方法，

get:function(target,key,property){

console.log('target:',target);

console.log('key',key);

console.log('property',property);

return target[key]

},

//set属性是值你要改变Proxy属性值时，进行的预先处理

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

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

return target[key] = value;

}

})

console.log(pro.name)

pro.name='技术胖';//改变值时触发set

console.log(pro.name)

//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

let target = function(){

return 'i am jspang';

}

let hander = {

apply(target,ctx,args){

console.log('do apply');

return Reflect.apply(...arguments);

}

}

let pro = new Proxy(target,hander);

console.log(pro())

11, 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

//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

let state=1;

//resolve成功 ,reject失败

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');

}

}

Step2, step3我就不写了 跟上面的一样内容换成 坐下吃饭 收拾桌子

// val 是Promise对象执行成功或者失败返回的内容

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;

});

1. 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。

Class Coder{

Name(val){

Console.log(val);

Return val;//要是被下面的方法用this调用的话 不return就没有返回值

}

// 在class 声明的时候 多个方法中间不用加分号

skill(val){

//还有this.name('jspang') 要注意return 的用法。

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

}

//constructor 传的是class类的参数

constructor(a,b){

this.a = a;

this.b = b;

}

// 类的参数调用的方法

add(){

return this.a + this.b;

}

}

Let jspang = new Code(1,2);

Jspang.add() //3

// 类的继承

class htmler extends Coder{

}

let pang = new htmler;

pang.name('bbb'); //因为继承了 直接有值

1. 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。

export var name="jspang";

import {name} from './temp';

console.log(name);

//export 的东西 在引入的时候是需要 import {xxx} from 'xxxxx'

//export default 是默认输出(只能有一个) 在引入的时候是需要 import 自定义名字 from 'xxxx'