**ES6重点知识介绍**

## 代码块

在大括号内的都为一个代码块为一个作用域。

## 作用域

for循环中i为全局变量，但是只应用在循环内部有效。for循环会产生两层作用域，父层变量可在子作用域访问，在子作用域中又可声明相同变量，因变量不容所以不冲突。

## const命令

定义常量，固定不变，（约定常量名全部大写），使用方法与变量相同但不可改变。但若定义了对象因其存在堆内在栈上只有一个地址，若地址不改变则不报错，可以通过更改堆内的属性内容或增伤属性，只要地址不改变则无错。

若想数组、对象都不可更改：使用Object.freeze(内容)来冻结内容无妨更改。冻结后不可更改，但是更改后也不报错。

## let命令

用来声明变量。它的用法和var类似，但是所声明的变量，只在let命令所在的代码块中有效。

## 不存在变量提升

Var命令会产生变量提升现象（即变量可以在声明之前使用）， 值为undefined

## 不允许重 复声明变量

let命令let命令不允许重复声明变量

## 变量的结构赋值

这种写法属于“模式匹配”，只要等号两边的模式相同，左边的变量就会被赋予对应的值。

let [foo, [[bar], baz]] = [1, [[2], 3]];

foo // 1

bar // 2

baz // 3

let [ , , third] = ["foo", "bar", "baz"];

third // "baz"

默认值

解构赋值允许指定默认值。默认值生效的条件是传入参数的值为undefined

let [foo = true] = [];

foo // true

对象的解构赋值，同样可以嵌套,但是如果嵌套的对象父级不存在则子级即使存在也无法获取。

let { foo, bar } = { foo: "aaa", bar: "bbb" };

foo // "aaa"

bar // "bbb"

如果想让变量名与属性名不一致必须将属性名赋值给变量名，后者被赋值

var { foo: baz } = { foo: 'aaa', bar: 'bbb' };

baz // "aaa"

let obj = { first: 'hello', last: 'world' };

let { first: f, last: l } = obj;

f // 'hello'

l // 'world'

对象的解构赋值的简写

let { foo: f, bar: b } = { foo: "aaa", bar: "bbb" };

## promise

Promise 是异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理和更强大。它由社区最早提出和实现，ES6 将其写进了语言标准，统一了用法，原生提供了Promise对象。

所谓Promise，简单说就是一个容器，里面保存着某个未来才会结束的事件（通常是一个异步操作）的结果。从语法上说，Promise 是一个对象，从它可以获取异步操作的消息。Promise 提供统一的 API，各种异步操作都可以用同样的方法进行处理。

在数据层实例上创建Promise对象，填入内容为ajax获取，当返回数据时，将数据以resolve方法返回成功函数，以reject返回访问失败函数。在应用层调用该对象传参后可跟then方法中获取成功与失败函数。

数据层实例：

function ajaxmode(m,url,parm){

var promise=new Promise(function(resolve,reject){

var o;

if(window.XMLHttpRequest){

o=new XMLHttpRequest()

}else if(window.ActiveXObject){

o=new ActiveXObject("Microsoft.XMLHTTP")

}

o.open(m,url,true);

o.onreadystatechange=function(){

if(o.readyState==4){

if(o.status==200){

var da=JSON.parse(o.responseText).results[0];

resolve(da); 返回成功的值

}else{

reject("数据获取失败") 返回失败的值

}

}

}

o.send(null);

})

return promise;

应用层实例：ajaxmode("get",url).then(function(data){console.log(data)},function(err){console.log(err)})

## Obj.assign：浅拷贝，对象属性的合并

传送门：<https://blog.csdn.net/zgrkaka/article/details/70792297>

var nObj = Object.assign({},obj,obj1);//花括号叫目标对象，后面的obj、obj1是源对象。对象合并是指：将源对象里面的属性添加到目标对象中去，若两者的属性名有冲突，后面的将会覆盖前面的

## class

相当于提供了系统的构造函数，绑定原型，可继承制定对象功能。方法之间可以不加分号，加了会报错。创建的方法都已经绑定在了原型上。

1. 创建模板：

class Person { 此处为通过class创建的名字。

constructor(name,address){ 初始化值，默认的方法，若没有写则会创建空的方法。

this.name=name; 绑定值

this.address=address;

}

sleep(){ 创建行为

console.log(this.name+"is　sleeping...")

}

}

1. 使用模板：

var p=new Person("小明","河南"); 根据模板创建，必须使用new

p.sleep(); 调用模板行为

**继承**

用法： class 创建名 extends 继承的对象名

class Studen extends Person{

study(){ 若想有不同的功能则需要添加独有的行为或者初始化值。

console.log(this.name+"正在学习")

}

}

## 模块

ES6 模块的设计思想，是尽量的静态化，使得编译时就能确定模块的依赖关系，以及输入和输出的变量。CommonJS 和 AMD 模块，都只能在运行时确定这些东西。比如，CommonJS 模块就是对象，输入时必须查找对象属性。ES6 模块的设计思想，是尽量的静态化，使得编译时就能确定模块的依赖关系，以及输入和输出的变量。CommonJS 和 AMD 模块，都只能在运行时确定这些东西。比如，CommonJS 模块就是对象，输入时必须查找对象属性。

**注意：**可导出任意数据，但是但是不能导出表达式，可用于变量，对象，class，函数。

**配置环境：**当在页面应用时因为页面无法读取该属性所以需要转码后在输出。需要安装几个插件

需要安装的es6转码的包有三个： babel-core, babel-loader, babel-preset-es2015

再装两个包用于打包及静态服务器功能：

webpack webpack-dev-server

为了使用Webpack命令，再装webpack的全局模式

在根目录中加入老师给的两个配置文件，将webpack-config.js的输出文件更名和路径。

在package.json文件中的scripts属性中添加启动命令： "dev":"webpack-dev-server --inline --hot" 静态服务器与页面热刷新。

更改后使用命令 npm run dev 来运行页面的热刷新。

使用webpack –w命令会监控代码文件是否改变，改变后会运行一次程序。

注：在页面的引入页面必须更改为转码后的地址，导入文件常起名为mian.js

**用法一：导入名称与导出名称必须一直**

在输出文件中输出数据

例：let username="豆豆";

let userage="20";

export {username,userage} 输出多个数据。

在接受文件中接收数据，并操作一般起名为main.js。使用命令imp ort 接收指定文件的制定数据。

例：import {username as name,userage as age} from './person.js'; 使用as是为了起别名，导入导出都可以写。

console.log(name,age)

**用法二：默认输出值无名称，导入时随便起个名字。若导出无大括号则导入也不能加入大括号。**

导出数据：export default {

name:"豆",

age:18,

address:"河南"

}

导入数据：import dou from './person.js'; 可以随意起个名字