一．Let:

1. let定义变量没有预解析（不存在变量提升）
2. let不能重复定义（指的是在同个{}中不能重复let定义）
3. for(){}循环中，for循环（let i=0）中i的定义是父级定义，{let i =1}花括号中的i定义是另一个定义。

二．Const:(特性与let一样)

1. const定义的变量不能修改
2. const定义常量时必须赋值，不能后面赋值，且不能修改该 赋值

三．解构赋值：

数组：

Let [a,b,c]=[1,12,3]

Json:

Let {name,age,job}={name:manting,age:18,job:aaa}

四．‘’字符串模板

优点：可以随意换行，${name}可以表示变量

五．关于字符串

1.字符串查找：

includes（） str.indexOf(要找的单词) 返回索引（位置），没找到返回-1

str.includes(要找的单词) 返回true，没找到false

2.字符串是否以谁开头：str.startsWith(检测的东西) 返回true，没找到false

3.字符串是否以谁结尾：str.endsWith(检测的东西，可用于检测文件后缀.png)返回true/false

4.重复字符串：str.repeat(3重复的次数)

5.字符串填充：往前填充：str.padStart(字符串总长度，填充的东西)

往后填充：Str.padEnd(字符串总长度，填充的东西)

Str. padStart (str.length+padStr.length，padStr)

六．函数

1.函数默认参数

Function show(a=”name”,b=”age”){}其中name，age就是当函数没有传参时的默认参数

2.函数的参数默认已经被定义，不能重复let,const声明

function show(a=18){

let a=15//错误，因为默认参数里面a已经被定义

console.log(a)

}

Show()

七．扩展运算符，reset运算符,剩余运算符

…展开数组[1,2,3,4,5] …[1,2,3,4,5] ----🡪 1,2,3,4,5

可以收缩数组a=1,2,3,4,5 …a----🡪 [1,2,3,4,5]

剩余参数：必须放到 最后

八．箭头函数

()=>{

语句

return

}

1. let show =(a,b)=>{return a+b}
2. this指向问题，当使用箭头函数时，this指向当前定义函数所在的对象，不再试运行时所在的函数对象
3. 箭头函数里面没有arguments,用’…’
4. 箭头函数不能当构造函数

九．数组

1.arr.fotEach( //没有返回值，代替for循环使用

function(var,index,arr){

console.log(this,val,index,arr)

},123

)

没有返回值，代替for循环使用

2.arr.map(循环的回调函数，this指向谁 ){}// 返回是一个新的数组（有返回值）

正常情况下，需要配合return，返回是一个新的数组

若是没有return,相当于使用forEach

重新整理数据结构：

[{title:’aa’}] 🡪[{t:’a’}]

Eg: let arr=[

{title:"aaaa",hot:true},

{title:"bbb",hot:false},

{title:"ccc",hot:false},

{title:"ddd",hot:true}

]

let newarr=arr.map((item)=>{

let jisons=[]

jisons.tts=item.title

jisons.hots=item.hot

return jisons

})

console.log(newarr)

1. arr.filter(循环的回调函数，this指向谁){} //过滤效果，返回一个数组（有返回值）

//回调函数里面返回结果为true的时候，该item就留下

过滤一些不合格的“元素”，返回值如果为true就留下

Eh: let arr=[

{title:"aaaa",hot:true},

{title:"bbb",hot:false},

{title:"ccc",hot:false},

{title:"ddd",hot:true}

]

let newarr=arr.filter((item)=>{

return item.hot==true

})

console.log(newarr)

1. arr.some(循环的回调函数，this指向谁){}//返回true和false

类似查找，数值里面某一个元素符合条件，返回true

let arr=["apple","banana","orange"]

function findarry(arr,item){

return arr.some((val,index,arr)=>{

return val== item

})

}

console.log(findarry(arr,"banana"))//查找这个arr数组里面是否有banana,有就返回true

1. arr.every(循环的回调函数，this指向谁){} //返回true和false

数值里面所有元素都符合条件，才会返回true

let arr1=[1,3,5,7,9,10]

let arr2=[1,3,5,7,9]

function aeeevery(arr){

return arr.every((item,index,arr)=>{//every要循环每一项，必须所有项回调函数的返回值都为true才会返回true

return item%2 ==1 //判断这个数组是否全部的数字都为奇数（就是余数为1）

})

}

console.log(aeeevery(arr1))

console.log(aeeevery(arr2))

--------------------------------------------------区分线--------------------------------------------------------------

1. arr.reduce(prev,cur,index,arr)//求数组的和,阶乘,（从左往右）
2. // arr.reduce()求数组的和
3. let arr=[1,2,3,4,5,6,7,8,9,10]
4. let aa=arr.reduce((prev,cur,index,arr)=>{//（之前的数，现在当前的值，当前的所引，当前的数组）
5. console.log(prev)
6. return prev+cur
7. })
8. console.log(aa)

执行顺序：

1. prev：0 cur：1 return 1
2. prev：1 cur：2 return 1+2
3. prev：3 cur：3 return 3+3
4. prev：6 cur：4 return 6+4

…

let arr=[2,2,4]

let aa=arr.reduce((prev,cur,index,arr)=>{//（之前的数，现在当前的值，当前的所引，当前的数组）

console.log(prev,cur)

console.log()

return Math.pow(prev,cur)

})

console.log(aa)

执行顺序：

1. prev：0 cur：2 return 2
2. prev：2 cur：2 return 2\*2
3. prev：4 cur：4 return 256(4\*4\*4\*4)

7. arr.reduceRight()//求数组的和,阶乘,（从右往左）

let arr=[2,2,3]

let aa=arr.reduceRight((prev,cur,index,arr)=>{//（之前的cur数，现在当前的值，当前的所引，当前的数组）

console.log(Math.pow(prev,cur))

return Math.pow(prev,cur)

})

console.log(aa)

执行顺序：

1. prev：0 cur：3 return 3
2. prev：3 cur：2 return 3\*3
3. prev：9 cur：2 return 9\*9

8.for of for (let aa of arr){}

for..of适用遍历数/数组对象/字符串/map/set等拥有迭代器对象的集合.但是不能遍历对象,因为没有迭代器对象

arr.keys()数组的下标

arr.entries()数组的某一项

let arr=["apple","banana","orange"]

for(let val of arr){//arr循环每一项的值

console.log(val)

}

for(let val of arr.keys()){//arr.keys()循环每一项索引的值

console.log(val)

}

for(let val of arr.entries()){//arr.entries()循环每一项索引和值

console.log(val)

}

补充：for in

遍历对象 通常用for in来遍历对象的键名

var myObject={

　　a:1,

　　b:2,

　　c:3

}

for (let val in myObject) {

console.log(val);

}

for of遍历的只是数组内的元素，而不包括数组的原型属性method和索引name

9.Array.from()//返回一个数组（Array.from()适用于所有具备length属性的数据类型的转换为数组）

window.onload=function(){

let ali=document.querySelectorAll("ul li")

let allt=Array.from(ali)

console.log(allt)

}

10.Array.of():把一组值转换为数组

let arr=Array.of('apple','banaba','orange')

console.log(arr)

11.arr.find()//找出数组里面第一个符合的值返回,如果没有就返回undefined

let arr=[2,10,15,19,30,42,48]

let sd=arr.find(function(val,index,arr){

return val>15//找出数组里面第一个符合的值返回,如果没有就返回undefined

})

console.log(sd)

12. arr. findIndex()//找出数组里面第一个符合的值返回该值的索引,如果没有就返回-1

let arr=[2,10,15,19,30,42,48]

let sd=arr.findIndex(function(val,index,arr){

return val>115//找出数组里面第一个符合的值返回,如果没有就返回undefined

})

console.log(sd)

13.arr.fill()填充

arr.fill(填充的东西，开始位置（包含），结束位置（不包含）)

let arr=new Array(10)

arr.fill('填充的内容','1','4')

console.log(arr)

14.arr.includes()//includes() 方法用来判断一个数组是否包含一个指定的值，如果是返回 true，否则false

let site = ['runoob', 'google', 'taobao'];

console.log(site.includes('runoob'))

补充：indexOf() 方法可返回某个指定的字符串值在字符串中首次出现的索引

let site = ['runoob', 'google', 'taobao','google'];

console.log(site.indexOf('google'))

十．对象json

1.jison定义对象

let name="aa"

let age="18"

let jison={

name,

age,

show(){

return this.name

},

nones(){

return this.age

}

}

console.log(jison.show())

2.Object.is(a,b)比较两个东西是否相等,返回true,false

console.log(Object.is(+0,-0))

console.log(Object.is(+0,-0))

3.Object.assign()用来：1)复制对象，2)合并数组，对象

Object.assign(目标对象，放进目标对象的参数1，放进目标对象的参数2，放进目标对象的参数3)

目标对象可以是数组也可以是对象，放进目标对象的参数里面的属性名相同的话，放进目标对象的参数位置在后面的会覆盖前面的值

let json={a:1}

let json2={b:2,a:2}

let json3={c:3}

let ss={}//ss=[]

let obj=Object.assign(ss,json,json2,json3)

console.log(obj)

console.log(ss)

4. Object.keys()，返回一个数组，成员是参数对象自身的（不含继承的）所有可遍历（enumerable）属性的键名

Object.values()，返回一个数组，成员是参数对象自身的（不含继承的）所有可遍历（enumerable）属性的键值

Object.entries()，返回一个数组，成员是参数对象自身的（不含继承的）所有可遍历（enumerable）属性的键值对数组。

1. 对象身上的拓展…

let json={a:1,b:2};

let json2= {...json}

console.log(json2)

十一.promise

1. new Promise((resolve,reject)=>{

// resolve调用成功

// reject失败的时候调用

}).then(res=>{//执行回调成功

}).catch(err=>{ //捕获错误

})

Promise. resolve:将现有的东西，转成一个Promise对象，resolve状态，成功状态

new Promise(resolve=>{ resolve(‘aaa’)})

Promise. reject:将现有的东西，转成一个Promise对象，reject状态，成功状态

new Promise(reject =>{ reject (‘aaa’)})

等价于：new Promise((resolve ,reject) =>{

resolve(‘aaa’);

reject(‘bbb’);

})

2. Promise.all([p1,p2,p3]):把Promise打包，扔到一个数组里面，打包完之后还是一个Promise对象。

Promise.all必须确保所有的Promise对象，都是resolve状态，才会执行then

let p1=Promise.resolve('aaa');

let p2=Promise.resolve('bbb');

let p3=Promise.resolve("ccx");

Promise.all([p1,p2,p3]).then(res=>{

// console.log(res)

let [res1,res2,res3]=res

console.log(res1,res2,res3)

}).catch(err=>{

console.log(err)

})

3.Promise.race([p1,p2,p3])只要参数中有一个率先得到结果就开始执行

let p1=Promise.reject('aaa');

let p2=Promise.resolve('bbb');

let p3=Promise.resolve("ccx");

Promise.race([p1,p2,p3]).then(res=>{

console.log(res)

}).catch(err=>{

console.log(err)//执行，以为p1先获取到结果

})

十二。模块化

Js不支持模块化

Ruby require

Puthon import

在ES6之前，社区制定一套模块规范

Commonjs 主要服务端nodejs require(‘http’)

AMD requires,CURIJS

CMD seaJs

ES6出来，统一服务端和客户端模块规范

imputer{xx} ddd

Math.pow()

Math.abs()

1. 如何让定义模块

export a

export let a=12

export {a as aa} 引用时使用aa

1. 如何使用

<script type="module"> import './js/mkh.js'</script>

Import 特点：

1. import可以是相对路径，也可以是绝对路径
2. import只会导入一次（无论引入多少次）
3. import {a as aa} from ‘./js/mkh.js’
4. import \* as mod from './js/mkh.js';
5. 有提升效果，import会自动提升到顶部，首先执行
6. 导出去模块内容，如果里面有定时器或者其他更改到值，引用的页面页会跟着改变

补充:Import()类似node里面require，可以动态引入，默认Import语法不能写到if里面

Import的返回值是个Promise对象，所以可以使用Promise的一切属性

import('./js/mkh.js').then(res=>{

console.log(res)

})

优点： 1.按需加载

2．可以写在if中

3.路径页可以动态

3．async await:异步加载

// 异步加载、

async function mani(){

const mod1=await import('./js/mkh.js')

const mod2=await import('./js/copy.js')

console.log(mod1,mod2)

const [m1,m2]=await Promise.all([

import('./js/mkh.js'),

import('./js/copy.js')

])

console.log(m1,m2)

}

mani()

4.使用严格模式’use strict’