1.

箭头函数=>的语法，实现在ES6标准中。

箭头函数和一般的函数不同，**箭头函数不会绑定this。 或者说箭头函数不会改变this本来的绑定. this还是指向原来的对象。**

函数需要参数：

(参数1，参数2) => {函数体}

函数不需要参数:

() => {函数体}

另外箭头函数还可以更加的简洁：

函数只需要一个参数， 那么参数地方不需要括号：

参数1=> {函数体}

箭头函数如果有花括号则默认没有return， 如果没有花括号则默认有return，没有花括号也可以使用多条表达式。

2.

#### Javascript的三元运算符

语法：条件?代码块1：代码块2

如果条件为真则执行代码块1，反之执行代码块2.

例如：let test = 1>3?1:3;

则test为3

3.

**Javascript的foreach方法：**

**语法**

array.forEach(function(currentValue, index, arr), thisValue)

**参数**

|  |  |
| --- | --- |
| **参数** | **描述** |
| *function (currentValue, index, arr)* | 必需。 数组中每个元素需要调用的函数。 函数参数:   |  |  | | --- | --- | | **参数** | **描述** | | *currentValue* | 必需。当前元素 | | *index* | 可选。当前元素的索引值。可以在定义的回调函数中不定义 | | *arr* | 可选。当前元素所属的数组对象。可以在定义的回调函数中不定义 | |
| *thisValue* | 可选。传递给函数的值一般用 "this" 值。 如果这个参数为空， "undefined" 会传递给 "this" 值 |

**Filter函数**

在Javascript中，我们可以用数组的filter函数从数组中取出某些对象，例如在list of dict中取出某个dict元素。

注意的是，filter方法将提供一个回掉函数作为参数，该回调函数包括三个参数，元素自身，元素的index以及原数组。如果在回调函数中为true则将元素将会保留，如果为false该元素将会被去除。结果返回到一个新的数组中，而不影响原数组。

例如：

var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0];

var filtered = array.filter(function(value, index, arr){

return value > 5;

});

//filtered => [6, 7, 8, 9]

//array => [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]

**Map函数**

在Javascript中，我们可以用数组的map函数对数组中的每个元素执行一次函数，并且将函数的返回值放入新的数组当中。

例如：

var numbers = [1, 2, 3, 4];  
newArray= numbers.map(x => {

return x\*x

});

// newArray = [1, 4, 6, 9]

**Assign函数**

在Javascript中，我们可以用assign函数将源目标中的属性拷贝到目标对象当中。需要注意的是assign执行的是浅拷贝，也就是如果源目标的属性对应的是引用类型时，那么拷贝到目标中的是同一个对象，而不是一个新的对象。

例如：

const target = {a: 1, b: 2};

const source = {b: 4, c: 5};

const returnedTarget = Object.assign(target, source1， source2….);

// returnedTarget为{a:1, b:4, c:5}

**Find函数**

find() 方法返回通过测试（函数内判断）的数组的第一个元素的值。

find() 方法为数组中的每个元素都调用一次函数执行：

* 当数组中的元素在测试条件时返回 true 时, find() 返回符合条件的元素，之后的值不会再调用执行函数。
* 如果没有符合条件的元素返回 undefined

注意: find() 对于空数组，函数是不会执行的。

注意: find() 并没有改变数组的原始值。

测试函数的语法为：

用法：

array.find(function(currentValue, index, arr),thisValue)  
参数：  
  
*currentValue 必需。当前元素*

*index 可选。当前元素的索引值*  
*arr 可选。当前元素所属的数组对象*  
*thisValue 可选。 传递给函数的值一般用 "this" 值。*  
*如果这个参数为空， "undefined" 会传递给 "this" 值*

例子:

*var ages = [4, 12, 16, 20];  
 function checkAdult(age) {   
　　return age >= document.getElementById("ageToCheck").value;   
}   
function myFunction() {   
　　document.getElementById("demo").innerHTML = ages.find(checkAdult);   
}*

这里checkAdult函数便是测试函数， 他有一个参数也就是必须参数age, 表示每一个ages中的元素， 如果大于id=”ageToCheck”的数字则返回true, **那么find函数返回第一个符合条件的元素的值**。

4.

Javascript向数组头插入元素使用方法unshift(元素)， 向数组尾部插入元素使用方法push(元素)

5.

Javascript的解构引入语法：

import React, {Component} from ‘react’🡪意思是从react中引入React, 以及**React中的**Component.以后便可以直接在使用Component.

6.

Javascript模块化引入工具requireJs

随着网站功能逐渐丰富，网页中的js也变得越来越复杂和臃肿，原有通过script标签来导入一个个的js文件这种方式已经不能满足现在互联网开发模式，我们需要团队协作、模块复用、单元测试等等一系列复杂的需求。

RequireJS是一个非常小巧的JavaScript模块载入框架，是AMD规范最好的实现者之一。最新版本的RequireJS压缩后只有14K，堪称非常轻量。它还同时可以和其他的框架协同工作，使用RequireJS必将使您的前端代码质量得以提升。

在引入requireJs后，我们便可以在使用require关键字引入javascript模块。

8.

ECMAScript 和 JavaScript 的关系是，**前者是后者的规格，后者是前者的一种实现**。

javascript是netscape创造的并交给了国际标准化组织ECMA来制定标准化过程。

ES6是ECMA的为JavaScript制定的第6个版本的标准，标准委员会最终决定，标准在每年的 6 月份正式发布一次，作为当年的正式版本。**ECMAscript 2015 是在2015年6月份发布的ES6的第一个版本**。依次类推ECMAscript 2016 是ES6的第二个版本、 ECMAscript 2017 是ES6的第三个版本。

ES6增加的例子有：

|  |
| --- |
| 新增关键字Let：Let只在代码块内有效，var在全局范围内有效。Let只能声明一次，var可以声明多次。  **let关键字不存在变量提升，var关键字存在变量提升**。  例如：  console.log(a); //ReferenceError: a is not defined  let a = "apple";  console.log(b); //undefined  var b = "banana";  变量 b 用 var 声明存在变量提升，所以当脚本开始运行的时候，b 已经存在了，但是还没有赋值，所以会输出 undefined。  变量 a 用 let 声明不存在变量提升，在声明变量 a 之前，a 不存在，所以会报错。  Undefined和ReferenceError: x is not defined不一样，undefined是申明了但没有赋值具体的数值，值就是undefined,而没有申明过直接调用则是not defined, 是错误。 |
| const 声明一个只读变量，声明之后不允许改变，不允许赋值或者重新声明。意味着，一旦声明必须初始化，否则会报错。 |
| 使用语法糖class来定义类 |
| 提供箭头函数=》 |
| 模板字符串语法，在**反单引号**中，以${varname}的形式插入变量  例如：  let str=” world”;  console.log(`Hellow ${str}`) // 输出Hellow World |
| 扩展语法  扩展操作符 …  使用扩展操作符…, 用于将数组或者字典打开并且并且复制  例子1:  var mid = [3, 4];  var arr = [1, 2, ...mid, 5, 6]; // 通过扩展操作符，mid数组将被打开成为3,4元素插入到新的数组当中  例子2:  var arr = ['a', 'b', 'c'];  var arr2 = [...arr]; // **通过扩展操作符，复制出arr2和arr有同样内容。 这样就算arr2改变，arr也不会改变。如果通过arr2 = arr,那样则会因为是同一个引用地址而同时改变。**  arr2.push('d');  例子3:  var str = "hello";  var chars = [...str]; // 通过扩展操作符， 将字符串hello拆分为数组 |

9.

onChange={this.toggleTodo} 与onChange={this.toggleTodo()}的异同

因为onChange需要指定一个函数，所以this.toggleTodo是可以的传入一个function引用。但是this.toggleTodo()是不行的，因为这个意思是调用函数。

另外如果toggleTodo函数需要参数的时候，例如需要一个参数item,那么onChange={this.toggleTodo}的形式无法表达需要传入的参数，所以通用的方法是写成onChange = {()=>{this.toggleTodo(item)}, 表示定义一个箭头函数，不需要参数，然后这个箭头函数将调用this.toggleTodo并且传入参数item.

10.