**ES6面试题**

1、ES5 和 ES6 的区别，说几个 ES6 的新增方法（必会）

**ES5 和 ES6 的区别**

ECMAScript5，即 ES5，是 ECMAScript 的第五次修订，于 2009 年完成标准化

ECMAScript6，即 ES6，是 ECMAScript 的第六次修订，于 2015 年完成，也称 ES2015

ES6 是继 ES5 之后的一次改进，相对于 ES5 更加简洁，提高了开发效率

**ES6 的新增方法**

1、新增声明命令 let 和 const

在 ES6 中通常用 let 和 const 来声明，let 表示变量、const 表示常量

1.1）特点

let 和 const 都是块级作用域。以{}代码块作为作用域范围 只能在代码块里面使用不存在变量提升，只能先声明再使用，否则会报错。在代码块内，在声明变量之前， 该变量 都是不可用的。这在语法上，称为“暂时性死区”（temporal dead zone，简称 TDZ），在同一个代码块内，不允许重复声明

const 声明的是一个只读常量，在声明时就需要赋值。（如果 const 的是一个对象，对 象所包含的值是可以被修改的。抽象一点儿说，就是对象所指向的地址不能改变，而变量成员 是可以修改的。）

2、模板字符串（Template String）

用一对反引号(`)标识，它可以当作普通字符串使用，也可以用来定义多行字符串，也可以在字符串中嵌入变量，js 表达式或函数，变量、js 表达式或函数需要写在${ }中。

3、函数的扩展

3.1）函数的默认参数

ES6 为参数提供了默认值。在定义函数时便初始化了这个参数，以便在参数没有被传递进去时使用。

3.2）箭头函数

在 ES6 中，提供了一种简洁的函数写法，我们称作“箭头函数”。

3.2.1）写法

函数名=(形参)=>{……} 当函数体中只有一个表达式时，{}和 return 可以省 略当函数体中形参只有一个时，()可以省略。

3.2.2）特点

箭头函数中的 this 始终指向箭头函数定义时的离 this 最近的一个函数，如果没有最近的函数就指向 window。

4、对象的扩展

4.1）属性的简写

ES6 允许在对象之中，直接写变量。这时，属性名为变量名, 属性值为变量 的值。

var foo = 'bar';

var baz = {foo}; //等同于 var baz = {foo: foo};

方法的简写。省略冒号与 function 关键字。

var o = {

method() {

return "Hello!";

}

};

// 等同于

var o = {

method: function() {

return "Hello!";

}

};

4.2）Object.keys()方法

获取对象的所有属性名或方法名（不包括原形的内容），返回一个数组。

var obj={name: "john", age: "21", getName: function () { alert(this.name)}};

console.log(Object.keys(obj)); // ["name", "age", "getName"]

console.log(Object.keys(obj).length); //3

console.log(Object.keys(["aa", "bb", "cc"])); //["0", "1", "2"]

console.log(Object.keys("abcdef")); //["0", "1", "2", "3", "4", "5"]

4.3）Object.assign ()

assign 方法将多个原对象的属性和方法都合并到了目标对象上面。可以接收多个参数，第一个参数是目标对象，后面的都是源对象

var target = {}; //目标对象

var source1 = {name : 'ming', age: '19'}; //源对象 1

var source2 = {sex : '女'}; //源对象 2

var source3 = {sex : '男'}; //源对象 3，和 source2 中的对象有同名属性 sex

Object.assign(target,source1,source2,source3);

console.log(target); //{name : 'ming', age: '19', sex: '男'}

5、for...of 循环

var arr=["小林","小吴","小佳"];

for(var v of arr){

console.log(v);

}

//小林 //小吴 //小佳

6、import 和 export

ES6 标准中，JavaScript 原生支持模块(module)。这种将 JS 代码分割成不同功能的小块进行 模块化，将不同功能的代码分别写在不同文件中，各模块只需导出公共接口部分，然后通过模块的导入的方式可以在其他地方使用

export 用于对外输出本模块（一个文件可以理解为一个模块）变量的接口

import 用于在一个模块中加载另一个含有 export 接口的模块

import 和 export 命令只能在模块的顶部，不能在代码块之中

7、Promise 对象

Promise 是异步编程的一种解决方案，将异步操作以同步操作的流程表达出来，避免了层层 嵌套的回调函数，要是为了解决异步处理回调地狱(也就是循环嵌套的问题)而产生的

Promise 构造函数包含一个参数和一个带有 resolve（解析）和 reject（拒绝）两个参数的回 调。在回调中执行一些操作（例如异步），如果一切都正常，则调用 resolve，否则调用reject。 对于已经实例化过的 Promise 对象可以调用 Promise.then() 方法，传递 resolve 和reject方法作为回调。then()方法接收两个参数：onResolve 和 onReject，分别代表当前Promise 对象在成功或失败时

**Promise 的 3 种状态**

Fulfilled 为成功的状态，Rejected 为失败的状态，Pending 既不是 Fulfilld 也不是 Rejected 的状态，可以理解为 Promise 对象实例创建时候的初始状态

8、解构赋值

8.1）数组的解构赋值

解构赋值是对赋值运算符的扩展。

是一种针对数组或者对象进行模式匹配，然后对其中的变量进行赋值。

在代码书写上简洁且易读，语义更加清晰明了；也方便了复杂对象中数据字段获取。

数组中的值会自动被解析到对应接收该值的变量中，数组的解构赋值要一一对应如果有对应不上的就是 undefined

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

// a = 1 // b = 2 // c = 3

8.2）对象的解构赋值

对象的解构赋值和数组的解构赋值其实类似，但是数组的数组成员是有序的而对象的属性则是无序的，所以对象的解构赋值简单理解是等号的左边和右边的结构相同

let { foo, bar } = { foo: 'aaa', bar: 'bbb' }; // foo = 'aaa' // bar = 'bbb'

let { baz : foo } = { baz : 'ddd' }; // foo = 'ddd'

9、Set 数据结构

Set 数据结构，类似数组。所有的数据都是唯一的，没有重复的值。它本身是一个构造函数。

9.1）Set 属性和方法

Size（） 数据的长度

Add（） 添加某个值，返回 Set 结构本身。

Delete（） 删除某个值，返回一个布尔值，表示删除是否成功。

Has（） 查找某条数据，返回一个布尔值。

Clear（）清除所有成员，没有返回值。

9.2）主要应用场景：数组去重

10、class

class 类的继承 ES6 中不再像 ES5 一样使用原型链实现继承，而是引入 Class 这个概念

**ES6 所写的类相比于 ES5 的优点：**

区别于函数，更加专业化（类似于 JAVA 中的类）写法更加简便，更加容易实现类的继承

11、…

展开运算符可以将数组或对象里面的值展开；还可以将多个值收集为一个变量

12、async、await

使用 async/await, 搭配 Promise,可以通过编写形似同步的代码来处理异步流程, 提高代码的简洁性和可读性 async 用于申明一个 function 是异步的，而 await 用于等待一个异步方法执行完成

13、修饰器

@decorator 是一个函数，用来修改类甚至于是方法的行为。修饰器本质就是编译时执行的函数

14、Symbol

Symbol 是一种基本类型。Symbol 通过调用 symbol 函数产生，它接收一个可选的名字参数，该函数返回的 symbol 是唯一的

15、Proxy

Proxy 代理使用代理（Proxy）监听对象的操作，然后可以做一些相应事情

2、ES6 的继承和 ES5 的继承有什么区别（必会）

**ES6 的继承和 ES5 的继承的区别**

ES5 的继承是通过原型或者是构造函数机制来实现

ES6 用过 class 关键字定义类，里面有构造方法，类之间通过 extends 关键字实现，子类必须在 constructor 方法中调用 super 方法

3、var、let、const 之间的区别（必会）

**区别**

var 声明变量可以重复声明，而 let 不可以重复声明

var 是不受限于块级的，而 let 是受限于块级

var 会与 window 相映射（会挂一个属性），而 let 不与 window 相映射

var 可以在声明的上面访问变量，而 let 有暂存死区，在声明的上面访问变量会报错

const 声明之后必须赋值，否则会报错

const 定义不可变的量，改变了就会报错

const 和 let 一样不会与 window 相映射、支持块级作用域、在声明的上面访问变量会报错

4、Class、extends 是什么，有什么作用（必会）

**什么是 Class，Class 的作用**

ES6 的 Class 可以看作只是一个 ES5 生成实例对象的构造函数的语法糖。它参考了 java 语言，定义了一个类的概念，让对象原型写法更加清晰，对象实例化更像是一种面向对象编程。

**什么是 extends，extends 的作用**

extends 是 ES6 引入的关键字，其本质仍然是构造函数+原型链的组合式继承。

Class 类可以通过 extends 实现继承。

**Class 和 ES5 构造函数的不同点**

1、类的内部定义的所有方法，都是不可枚举的。

2、ES6 的 class 类必须用 new 命令操作，而 ES5 的构造函数不用 new 也可以执行。

3、ES6 的 class 类不存在变量提升，必须先定义 class 之后才能实例化，不像 ES5 中可以将构造函数写在实例化之后。

4、ES5 的继承，实质是先创造子类的实例对象 this，然后再将父类的方法添加到 this 上面。ES6 的继承机制完全不同，实质是先将父类实例对象的属性和方法，加到 this 上面（所以必须先调用 super 方法），然后再用子类的构造函数修改 this。

5、module、export、import 有什么作用（必会）

module、export、import 是 ES6 用来统一前端模块化方案的设计思路和实现方案。 export、import 的出现统一了前端模块化的实现方案，整合规范了浏览器/服务端的模块化方法，用来取代传统的 AMD/CMD、requireJS、seaJS、commondJS 等等一系列前端模块不同的实现方案，使前端模块化更加统一规范，JS 也能更加能实现大型的应用程序开发。

import 引入的模块是静态加载（编译阶段加载）而不是动态加载（运行时加载）。

import 引入 export 导出的接口值是动态绑定关系，即通过该接口，可以取到模块内部实时的值。

6、使用箭头函数应注意什么/箭头函数和普通函数的区别 （必会）

**区别**

用了箭头函数，this 就不是指向 window，而是父级（指向是可变的）

不能够使用 arguments 对象

不能用作构造函数，这就是说不能够使用 new 命令，否则会抛出一个错误

不可以使用 yield 命令，因此箭头函数不能用作 Generator 函数

7、介绍下 Set、Map 的区别（必会）

**区别**

应用场景 Set 用于数据重组，Map 用于数据储存

Set：

成员不能重复

只有键值没有键名，类似数组

可以遍历，方法有 add, delete,has

Map:

本质上是健值对的集合，类似集合

可以遍历，可以跟各种数据格式转换

8、setTimeout、Promise、Async/Await 的区别（必会）

**事件循环中分为宏任务队列和微任务队列**

宏任务（macrotask）：在新标准中叫 task

主要包括：script(整体代码)，setTimeout，setInterval，setImmediate，I/O，ui rendering

微任务（microtask）：在新标准中叫 jobs

主要包括：process.nextTick， Promise，MutationObserver（html5 新特性）

**setTimeout、Promise、Async/Await 的区别**

setTimeout 的回调函数放到宏任务队列里，等到执行栈清空以后执行

Promise.then 里的回调函数会放到相应宏任务的微任务队列里，等宏任务里面的同步代码执 行完再执行

async 函数表示函数里面可能会有异步方法，await 后面跟一个表达式async 方法执行时，遇到 await 会立即执行表达式，然后把表达式后面的代码放到微任务队列里，让出执行栈让同步代码先执行

9、Promise 有几种状态，什么时候会进入 catch？（必会）

**Promise 有几种状态**

三个状态：pending、fulfilled、reject

两个过程：padding -> fulfilled、padding -> rejected

**Promise 什么时候会进入 catch**

当 pending 为 rejectd 时，会进入 catch

10、ES6 怎么写 Class ，为何会出现 Class（必会）

**什么是 Class，Calss 的作用**

ES6 的 class 可以看作是一个语法糖，它的绝大部分功能 ES5 都可以做到，新的 class 写法只 是让对象原型的写法更加清晰、更像面向对象编程的语法

**ES6 怎么写 Class**

//定义类

class Point {

constructor(x,y) {

//构造方法

this.x = x; //this 关键字代表实例对象

this.y = y;

} toString() {

return '(' + this.x + ',' + this.y + ')';

}

}

11、Promise 构造函数是同步执行还是异步执行，那么then 方法呢？（必会）

Promise 构造函数是同步执行的，then 方法是异步执行的

12、Promise 只有成功和失败 2 个状态，怎么让一个函数无论 成功还是失败都能被调用？（必会）

**使用 Promise.all()**

Promise.all（）用于将多个 Promise 实例，包装成一个新的 Promise 实例

Promise.all（）接受一个数组作为参数，数组里的元素都是 Promise 对象的实例，如果不是，就会先调用下面讲到的 Promise.resolve（），将参数转为 Promise 实例，再进一步处理。（Promise.all（）方法的参数可以不是数组，但必须具有 Iterator 接口，且返回的每个成员都 是 Promise 实例。）

**示例：var p =Promise.all([p1,p2,p3])**

p 的状态由 p1、p2、p3 决定，分为两种情况。

当该数组里的所有 Promise 实例都进入 Fulfilled 状态：Promise.all\*\*返回的实例才会变成Fulfilled 状态。并将 Promise 实例数组的所有返回值组成一个数组，传递给 Promise.all 返回实例的回调函数\*\*。当该数组里的某个 Promise 实例都进入 Rejected 状态：Promise.all 返回的实例会立即变成Rejected 状态。并将第一个 rejected 的实例返回值传递给 Promise.all 返回实例的回调函数

13、ES6 如何转化为 ES5，为什么要转化（必会）

**ES6 语法为什么要转化 ES5 语法**

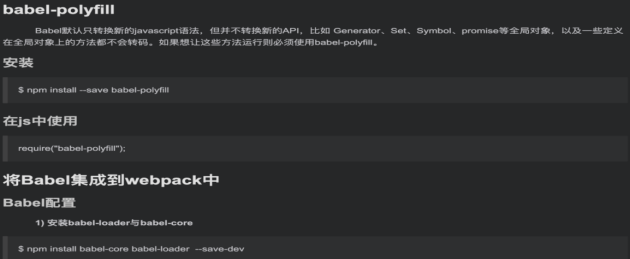
ECMAScript2015，更新语法、规则、功能，浏览器对 ES6 的支持程度并不是很好，如果写了 ES6 的代码，需要运行在浏览器上的时候，需要将 ES6 的代码转成 ES5 的代码去浏览器上运行。

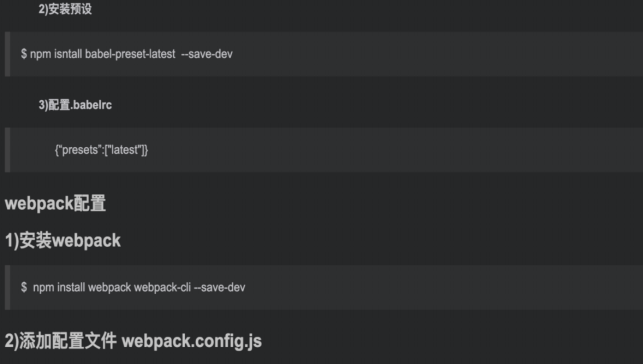
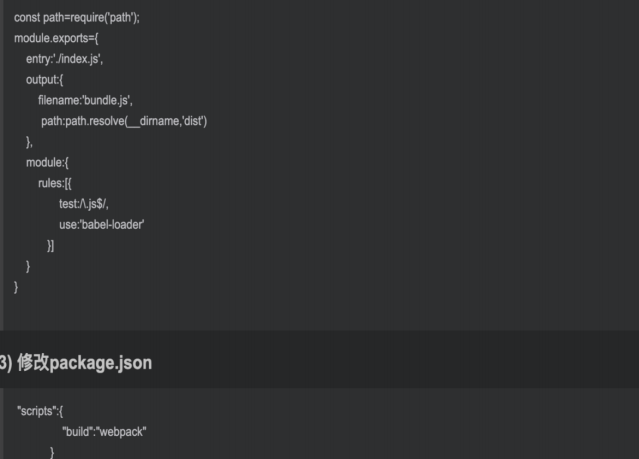
**Babel 是什么**

babel 是一个 ES6 转码器，可以将 ES6 代码转为 ES5 代码，以便兼容那些还没支持 ES6的平台

**ES6 如何转化为 ES5**







14、日常前端代码开发中，有哪些值得用 ES6 去改进的编 程优化或者规范（必会）

1、常用箭头函数来取代 var self = this;的做法。

2、常用 let 取代 var 命令。

3、常用数组/对象的结构赋值来命名变量，结构更清晰，语义更明确，可读性更好。

4、在长字符串多变量组合场合，用模板字符串来取代字符串累加，能取得更好地效果和阅读体验。

5、用 Class 类取代传统的构造函数，来生成实例化对象。

6、在大型应用开发中，要保持 module 模块化开发思维，分清模块之间的关系，常用import、 export 方法。

15、ES6 和 node 的 commonjs 模块化规范的区别（高薪常问）

ES6 是 js 的增强版，是 js 的语法规范，commonjs 都只是为了解决 js 文件之间的依赖和引用问题，所以是一种 js 的包管理规范，其中的代表是 node 遵循 commonjs 规范

16、Promise 中 reject 和 catch 处理上有什么区别（高薪 常问）

reject 是用来抛出异常，catch 是用来处理异常

reject 是 Promise 的方法，而 catch 是 Promise 实例的方法

reject 后的东西，一定会进入 then 中的第二个回调，如果 then 中没有写第二个回调，则

进入catch

网络异常（比如断网），会直接进入 catch 而不会进入 then 的第二个回调