

1、es5 和 es6 的区别，说一下你所知道的 es6

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 新增的一些特性：

1) let 声明变量和 const 声明常量，两个都有块级作用域

ES5 中是没有块级作用域的，并且 var 有变量提升，在 let 中，使用的变量一定要进行声明

2) 箭头函数

ES6 中的函数定义不再使用关键字 function()，而是利用了()=>来进行定义

3) 模板字符串

模板字符串是增强版的字符串，用反引号(`)标识，可以当作普通字符串使用，也可以用来定义多行字符串

4) 解构赋值

ES6 允许按照一定模式，从数组和对象中提取值，对变量进行赋值

5) for of 循环

for...of 循环可以遍历数组、Set 和 Map 结构、某些类似数组的对象、对象，以及字符串

6) import、export 导入导出

ES6 标准中，Js 原生支持模块(module)。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

7) 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

8) ... 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

9) 修饰器 @

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

10) 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

11) 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

12) 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、强大

13) 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

14) 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、var、let、const 之间的区别

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

3、使用箭头函数应注意什么？

（1）用了箭头函数，this 就不是指向 window，而是父级（指向是可变的）

（2）不能够使用 arguments 对象

（3）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

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

4、ES6 的模板字符串有哪些新特性？并实现一个类模板字符串的功能

基本的字符串格式化。将表达式嵌入字符串中进行拼接。用\${}来界定

在 ES5 时我们通过反斜杠()来做多行字符串或者字符串一行行拼接。ES6 反引号(`)就能解决

类模板字符串的功能



```
let name = 'web';  
  
let age = 10;  
  
let str = `你好， ${name} 已经 ${age}岁了`  
  
str = str.replace(/\$\{([^\}]*)\}/g,function(){  
  
    return eval(arguments[1]);  
  
})  
  
console.log(str);//你好， web 已经 10 岁了
```



5、介绍下 Set、Map 的区别？

应用场景 Set 用于数据重组，Map 用于数据储存

Set:

- (1) 成员不能重复
- (2) 只有键值没有键名，类似数组
- (3) 可以遍历，方法有 add, delete, has

Map:

- (1) 本质上是键值对的集合，类似集合
- (2) 可以遍历，可以跟各种数据格式转换

6、ECMAScript 6 怎么写 class ，为何会出现 class？

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



```
//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 + ')';

    }

}

}
```



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

promise 构造函数是同步执行的，then 方法是异步执行的

8、setTimeout、Promise、Async/Await 的区别

事件循环中分为宏任务队列和微任务队列

其中 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?

三个状态：pending、fulfilled、reject

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

当 pending 为 rejectd 时，会进入 catch

10、下面的输出结果是多少



```
const promise = new Promise((resolve, reject) => {  
  console.log(1);  
  resolve();  
  console.log(2);  
})
```



```
promise.then(() => {  
  console.log(3);  
})
```

```
console.log(4);
```



1 2 4 3

Promise 新建后立即执行，所以会先输出 1，2，
而 Promise.then() 内部的代码在 当次 事件循环的 结尾 立刻执行，所以会继续输出 4，最后输出 3

11、使用结构赋值，实现两个变量的值的交换

```
let a = 1;let b = 2;
```

```
[a,b] = [b,a];
```

12、设计一个对象，键名的类型至少包含一个 symbol 类型， 并且实现遍历所有 key

```
let name = Symbol('name');
```

```
let product = {
```

```
[name]:"洗衣机",  
"price":799  
};  
Reflect.ownKeys(product);
```

13、下面 Set 结构，打印出的 size 值是多少

```
let s = new Set();  
s.add([1]);  
s.add([1]);console.log(s.size);
```

答案：2

两个数组[1]并不是同一个值，它们分别定义的数组，在内存中分别对应着不同的存储地址，因此并不是相同的值

都能存储到 Set 结构中，所以 size 为 2

14、Promise 中 reject 和 catch 处理上有什么区别

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

15、使用 class 手写一个 promise



```
//创建一个 Promise 的类  class Promise{  
  constructor(executer){//构造函数 constructor 里面是个  
    执行器  
    this.status = 'pending';//默认的状态 pending  
    this.value = undefined//成功的值默认 undefined  
    this.reason = undefined//失败的值默认 undefined  
    //状态只有在 pending 时候才能改变  
    let resolveFn = value =>{  
      //判断只有等待时才能 resolve 成功  
      if(this.status == pending){  
        this.status = 'resolve';  
        this.value = value;  
      }  
    }  
  }  
}
```

```
}
```

```
//判断只有等待时才能 reject 失败
```

```
let rejectFn = reason =>{
```

```
  if(this.status == pending){
```

```
    this.status = 'reject';
```

```
    this.reason = reason;
```

```
  }
```

```
}
```

```
try{
```

```
  //把 resolve 和 reject 两个函数传给执行器 executer
```

```
  executer(resolve,reject);
```

```
  }catch(e){
```

```
    reject(e);//失败的话进 catch  }
```

```
}
```

```
then(onFulfilled,onReject){
```

```
  //如果状态成功调用 onFulfilled
```

```
  if(this.status = 'resolve'){
```

```
    onFulfilled(this.value);
```

```
  }
```

```
  //如果状态失败调用 onReject
```

```
if(this.status = 'reject'){  
    onReject(this.reason);  
}  
}  
}
```



16、如何使用 Set 去重

```
let arr = [12,43,23,43,68,12];  
let item = [...new Set(arr)];  
console.log(item);//[12, 43, 23, 68]
```

17、将下面 for 循环改成 for of 形式

```
let arr = [11,22,33,44,55];  
let sum = 0;for(let i=0;i<arr.length;i++){  
    sum += arr[i];  
}
```

答案：

```
let arr = [11,22,33,44,55];
```

```
let sum = 0;for(value of arr){  
    sum += value;  
}
```

18、理解 async/await 以及对 Generator 的优势

async await 是用来解决异步的，async 函数是 Generator 函数的语法糖

使用关键字 async 来表示，在函数内部使用 await 来表示异步

async 函数返回一个 Promise 对象，可以使用 then 方法添加回调函数

当函数执行的时候，一旦遇到 await 就会先返回，等到异步操作完成，再接着执行函数体内后面的语句

async 较 Generator 的优势：

(1) 内置执行器。Generator 函数的执行必须依靠执行器，而 Async 函数自带执行器，调用方式跟普通函数的调用一样

(2) 更好的语义。async 和 await 相较于 * 和 yield 更加语义化

(3) 更广的适用性。yield 命令后面只能是 Thunk 函数或 Promise 对象, async 函数的 await 后面可以是 Promise 也可以是原始类型的值

(4) 返回值是 Promise。async 函数返回的是 Promise 对象, 比 Generator 函数返回的 Iterator 对象方便, 可以直接使用 then() 方法进行调用

19、forEach、for in、for of 三者区别

forEach 更多的用来遍历数组

for in 一般常用来遍历对象或 json

for of 数组对象都可以遍历, 遍历对象需要通过和 Object.keys()

for in 循环出的是 key, for of 循环出的是 value

20、说一下 es6 的导入导出模块

导入通过 import 关键字



// 只导入一个

```
import {sum} from './example.js'
```

// 导入多个

```
import {sum,multiply,time} from "./exportExample.js"
```

// 导入一整个模块

```
import * as example from "./exportExample.js"
```



导出通过 export 关键字



//可以将 export 放在任何变量,函数或类声明的前面

```
export var firstName = 'Michael';
```

```
export var lastName = 'Jackson';
```

```
export var year = 1958;
```

//也可以使用大括号指定所要输出的一组变量 var firstName

```
= 'Michael';var lastName = 'Jackson';var year = 1958;
```

```
export {firstName, lastName, year};
```

//使用 export default 时,对应的 import 语句不需要使用大括号

```
let bosh = function crs(){}
```

```
export default bosh;
```

```
import crc from 'crc';
```


//不使用 export default 时，对应的 import 语句需要使用大括号

```
let bosh = function crs(){}  
  
export bosh;  
  
import {crc} from 'crc';
```