# 面试题

## JS的运行机制？

相似问题：

### 什么是事件循环（Event Loop）?

### 谈一谈你对JS中同步和异步的理解？

### 怎么理解js是单线程的，但又可以实现异步编程

### 简述JS单线程异步实现原理

js是单线程的语言，它本身不可能是异步的，但js的宿主环境（比如浏览器，Node）是多线程的，宿主环境通过某种方式（事件驱动）使得js具备了异步的属性。

js是单线程语言，浏览器只分配给js一个主线程，用来执行任务（函数），但一次只能执行一个任务，这些任务形成一个任务队列排队等候执行，但前端的某些任务是非常耗时的，比如网络请求，定时器和事件监听，如果让他们和别的任务一样，都老老实实的排队等待执行的话，执行效率会非常的低，甚至导致页面的假死。所以，浏览器为这些耗时任务开辟了另外的线程，这些任务是异步的。

JS明明单线程，但又同时可进行异步操作，这两者不是完全相反的嘛？ 没错，JS是单线程，但是JS是在浏览器中运行的脚本语言，它的宿主，浏览器可不是单线程的，这时候，大家又会产生疑问，这两者之间有什么关系呢？下面我们就来讲讲：

关于任务队列的理解：

单线程只有前一个任务结束，才能执行下一个任务。如果排队是因为计算量大，CPU忙不过来，倒也算了，但是很多时候CPU是闲着的，因为IO设备（输入输出设备）很慢（比如Ajax操作从网络读取数据），不得不等着结果出来，再往下执行。 JS语言的设计者意识到，这时主线程完全可以不管IO设备，挂起处于等待中的任务，先运行排在后面的任务。等到IO设备返回了结果，再回过头，把挂起的任务继续执行下去。 于是呢，任务就被分成两种，一种是同步任务，一种是异步任务 同步任务：只有前一个任务执行完成后，才可执行下一个任务，在主线程中 异步任务：这个队列的所有任务都是不进入主线程执行，而是被浏览提供的线程执行，当执行完毕后就会产生一个回调函数，并且通知主线程，在主线程执行完当前所执行的任务后，就会调取最早通知自己的回调函数，使其进入主线程中执行，比如ajax请求，在主线程中呈现的就是请求结果~

js的异步事件通常包括以下几类：

网络请求：ajax请求，服务端响应（随时发生）

定时器：setTimeOut，setTimeInterval（定时发生）

用户事件：鼠标，键盘，控件（随时发生）

浏览器事件：页面加载完毕，页面加载失败（随时发生） 以上情况下，代码的执行事件都是不确定的，或者不是立刻执行的，不可能由单线程来完成

## 作用域和作用域链

相似问题：

### 什么是作用域？

### 作用域分为哪两种？

### 谈一谈你对作用域链的理解

声明在任何函数以外的数据，拥有全局作用域。  
全局作用域的数据，在这条数据声明之后的任何地方都能访问和修改，且只有在页面关闭后才被删除。

在函数里面使用var,let声明的变量，拥有局部作用域  
局部作用域的数据，只能在函数内部进行访问和修改，且在函数运行以后被立即删除

局部作用域的数据：  
在函数内部使用var或let声明的变量（函数内未使用var或let声明而直接赋值的变量拥有全局作用域）  
函数的参数；  
函数内的声明函数和函数表达式

当我们调用一条数据的时候，会先在本作用域进行查找，如果找不到就向上查找父作用域，如果还是没有找到，就继续向上，一直找到全局作用域，还是找不到就报错。

## 闭包

相似问题：

### 什么闭包

在一个函数内部创建另一个函数，通过另一个函数访问这个函数的局部变量（参数和变量不会被垃圾回收机制所收回）  
是指有权访问另外一个函数作用域中的变量的函数。

### 闭包的优缺点，什么时候使用闭包？

闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。过度使用闭包会导致性能下降，尽量少使用闭包，只在必要时使用。

### 怎么解决闭包的变量长驻内存

解决方法：在退出函数时，将不使用的变量全部删除

### 什么使用使用闭包

闭包可以将一些不希望暴露在全局的变量封装成“私有变量”。

## ES6新增了哪些方法或属性

let const

箭头函数

解构赋值

剩余参数，3个点

扩展运算符

set()对象， map对象

数组的迭代方法（高阶函数）

模板字符串

promise对象

async和await (ES7)

Array.from() 和 Array.of()

对象简写，当key,value一样时，可以只写一个

函数可以设置默认参数

for...of循环

创建多个对象以前使用构造函数，ES6新增了class类

模块化  
exprot default // 导出  
import modA from './a.js' // 引入模块

## 浏览器兼容问题

浏览器分为标准浏览器（火狐，谷歌，IE9+）和非标准浏览器(IE6,7,8)

兼容问题产生原因在于浏览器对HTML5和CSS3,ES6标准支持的程度不同

CSS3动画 低版本浏览器不支持， 解决：用JS, GIF动画

低版本浏览器不支持ES6,ES7一些新特性，解决：使用babel转为ES5

低版本浏览器不支持vue, 使用jquery或原始JS

<https://www.jianshu.com/p/6afd596440bb>

## JS的数据类型

JS的数据类型分为两类：原始类型（基本数据类型，简单数据类型）和对象类型 （引用数据类型，复杂数据类型）

### 原始类型

数字（Number)  
字符串 (String)  
布尔值 (Boolean)  
空 (Null)  
未定义 (Undefined )

### 对象类型

除了5种原始数据类型之外的都是对象（Object）,对象是属性的集合，Object本质是由一组无序的键值对组成的

## 数据存储机制

相似问题

### 前端存储方法有哪些？

### 本地存储的方式

cookie localStorage sessionStorage

### localStorage sessionStorage区别

localStorage的生命周期是永久性的。假若使用localStorage存储数据，即使关闭浏览器，也不会让数据消失，除非主动的去删除数据

sessionStorage 的生命周期是在浏览器关闭前。也就是说，在整个浏览器未关闭前，其数据一直都是存在的。浏览器一旦关闭，数据就销毁了

## 深拷贝和浅拷贝

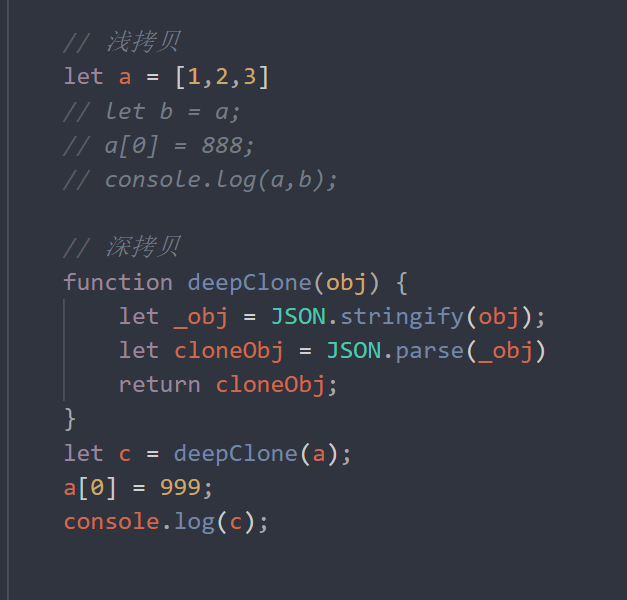
原始数据类型：在内存中，存放在栈中的简单数据段，它们直接存储在变量访问的位置。按照值来操作  
对象数据类型：在内存中，存放在堆中的对象，存储在变量处的值是一个指针，指向存储对象的内存处。按照地址来操作

浅拷贝：把数组或对象的引用地址赋值给另外一个新数组或对象。新对象/数组只是原对象的一个引用

深拷贝：把数组或对象的值赋值给另外一个数组或对象。是值 而不是 地址

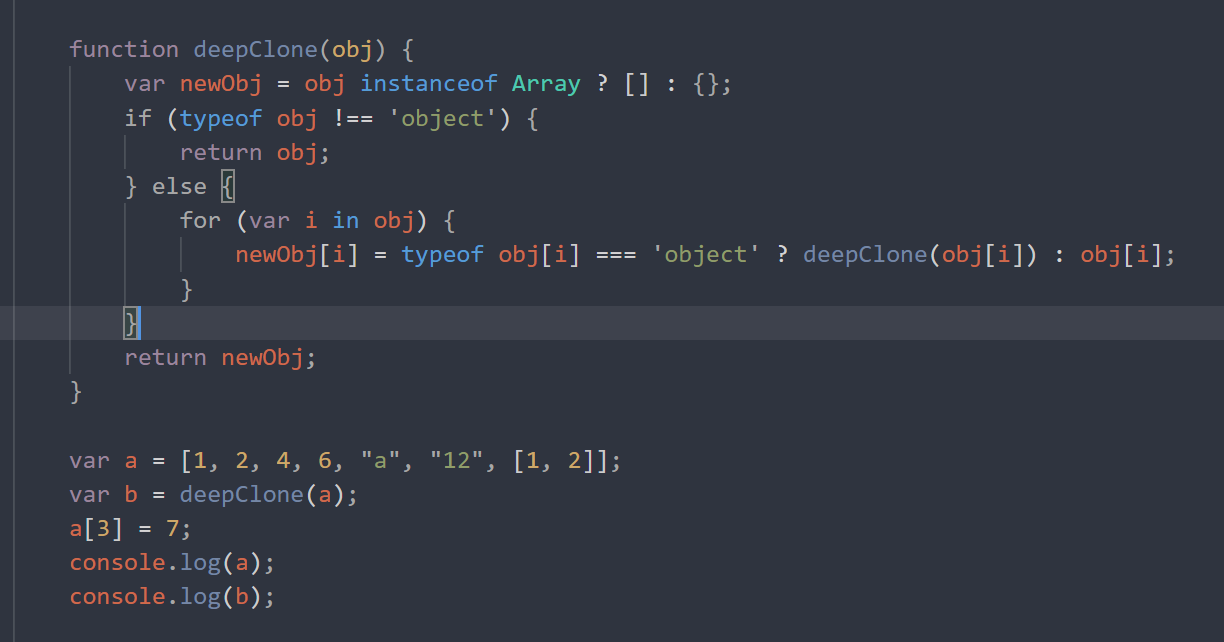
### 深拷贝方法一

使用JSON对象的方法



### 深拷贝方法二

使用递归



## call和apply, bind的区别是什么

哪一个性能更好？ call的性能更好

它们都可以改变函数里面的this指向

call 传参的时候是单个传

apply传参是一个集合或数组

bind只是对原函数的一个拷贝

## 箭头函数和普通函数

区别：

箭头函数表达式的语法比函数表达式更简洁

没有自己的this，arguments，super

箭头函数不能用作构造函数，和 new一起用会抛出错误。

箭头函数没有prototype属性。

箭头函数没有arguments对象。可以用rest参数代替

## 跨域问题

https://segmentfault.com/a/1190000011145364



## JS去重方法

set对象

for循环遍历