IT视频学习网Web前端复习题

**1.请按自己的理解简述http 2与http 1.0、http 1.1的区别。**

**答**：**HTTP1.0**：浏览器与web服务器的连接过程是短暂的，每次连接只处理一个请求和响应,客户端与web服务器建立连接后，只能获得一个web资源。

**HTTP1.1**：在一个tcp连接上可以传送多个http请求和响应，多个请求和响应过程可以重叠进行，增加了更多的请求头和响应头；允许客户端与web服务器建立连接后，在一个连接上获取多个web资源。

关于**HTTP2**，因为浏览器会有并发请求限制，在 HTTP / 1.1 时代，每个请求都需要建立和断开， 消耗了好几个 RTT 时间，并且由于 TCP 慢启动的原因，加载体积大的文件会需要更多 的时间。 在 HTTP / 2.0 中引入了多路复用，能够让多个请求使用同一个 TCP 链接，极大 的加快了网页的加载速度。并且还支持 Header 压缩，进一步的减少了请求的数据大小。



**2.请列举三种禁止浏览器缓存的头字段，并写出相应的设置值。**

**答**：expires: 0

cache-control: no-cache

pragma: no-cache

****

**3. 除了 Etag，还有哪些缓存手段，哪些是 HTTP1.0 的？**

**答**：Expires、Cache-Control、Last-Modified、Etag这是前端缓存常用4种手段。

**Expires**首部主要是针对**HTTP 1.0**版本，是响应头里的一个头部，是以日期时间的形式返回给客户端，指示可以缓存这个资源（响应实体）直到指定的日期时间。

**Cache-Control**首部是在**HTTP 1.1**版本以后加入的，提供了细粒度的缓存策略

**Last-Modified 在HTTP 1.0**推出的，指服务器文件的最后修改时间，浏览器会带上If-Modified-Since向服务器发送请求，与服务器文件修改时间Last-Modified做对比，如果时间不同，则获取数据返回200，否则返回304后调用浏览器本地硬盘的缓存。

**ETag** 类似于文件指纹，在**HTTP 1.1**推出，该版本号是由服务端随机生成的，浏览器会带上If-None-Match向服务器发送请求，与服务器文件修改版本ETag做对比，如果版本号不同，则获取数据返回200，否则返回304后调用浏览器本地硬盘的缓存，这种方式比Last-Modified靠谱。

****

**4.常用检测数据的方法有哪些，为什么typeof null 输出object，如何正确输出Null的类型？**

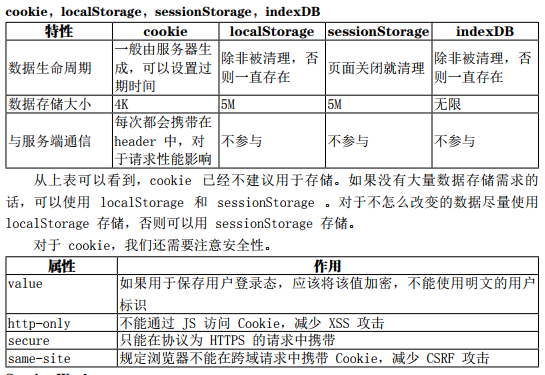
**答：**instanceof 可以正确的判断对象的类型，因为内部机制是通过判断对象的原型链 中是不是能找到类型的 prototype。此外还有typeof。

typeof null 输出object是因为在JS 的最初版本中，使用的是32位系统。为了性能考虑使用低位存储了变量的类型信息，000开头代表是对象，然而null 表示为全零，所以将它错误的判断为object。

如果我们想获得一个变量的正确类型，可以通过0bject. prototype. toString.call(xx)。这样我们就可以获得类似[object Type]的字符串。

****

**5.前端存储有哪些，有什么区别？**



****

**6.看你简历上写研究过源码，谈谈设计模式有哪些原则吧，你在工作中用过哪些设计模式？**

**答：**设计模式6大原则有单一职责原则、里氏替换原则、依赖倒置原则、接口隔离原则、迪米特原则、开闭原则。

设计模式根据情况，可谈常用的工厂模式、组合模式、观察者模式，混合开发常用的桥接模式等等。

****

**7.用过哪些前端框架，谈谈你对目前所使用的框架的用法和理解？**

**答：**当下前端领域流行的框架有Vue、React、Jquery等等。Jquery属于以选择器为导向的框架，整个框架或库主体是一个特殊类数组对象，方便集化操作，因为选择器通常是一下子选择到 N 个元素节点，于是便一并处理了但是Vue、React、Angular是具有明确分层构架的 MV\*框架，而且在 MVVM 框架中，原有 DOM 操作被声明式绑定取代了，由框架自行处理，用户只需专注于业务代码，典型就是Vue、Angular。

****

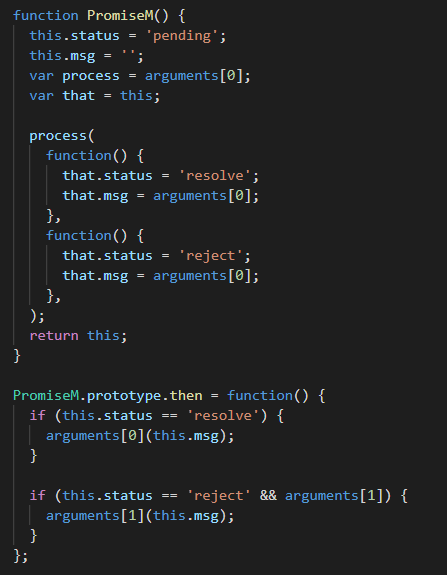
**8.为什么JS需要异步，谈谈你对JS事件循环的理解。（一般会写一堆promise、setTimeout让你判断顺序）**

**答：**因为**没有锁机制**，所以JS 是门非阻塞单线程语言。JS 在执行的过程中会产生执行环境，这些执行环境会被顺序的加入到执行栈中。 如果遇到异步的代码，会被挂起并加入到 Task（有多种 task） 队列中。一旦执行栈 为空，Event Loop 就会从 Task 队列中拿出需要执行的代码并放入执行栈中执行，所 以本质上来说 JS 中的异步还是同步行为。事件循环是协调执行栈顺序执消息队列的一种机制，抛开Node环境不提，**只谈在浏览器环境下**，正确的一次 Event loop 顺序是：**①**执行同步代码（宏任务）；**②**执行栈为空，查询是否有微任务需要执行；**③**执行所有微任务；**④**必要的话渲染 UI；**⑤**然后开始下一轮 Event loop，执行宏任务中的异步代码。

****

**9.手写一个用JS 实现Promise。**

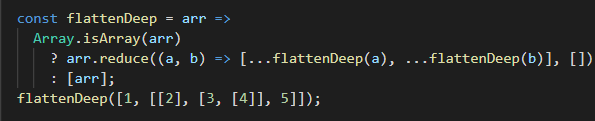
**答：**



****

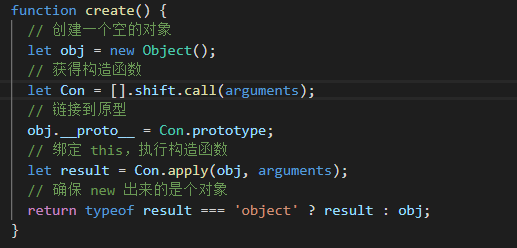
**10.常用的数组方法都知道吧，Map常用吗，如何实现数组降维？**

**答：**Map 作用是生成一个新数组，遍历原数组，将每个元素拿出来做一些变换然后 append 到新的数组中，Map 有三个参数，分别是当前索引元素，索引，原数组，而且FlapMap 方法 和 map 的作用几乎是相同的，但是对于多维数组来说，会将原数组降维。 可以将 FlapMap 看成是 map + flatten ，目前该函数在浏览器中还不支持。如果想将一个多维数组彻底的降维，可以这样实现：



**11.当我们new一个对象时，其过程时什么样的，手写实现一个new。**

**答：**①新生成了一个对象；②链接到原型；③绑定 this；④返回新对象。



**12. js 实现事件分发系统,要求包含三个最基本功能 on (监听事件), off (移出事件), emit (触发事件)**

**答：**封装一个封装 EventEmitter



**13.跨域的含义，常用跨域的解决方案？**

**答：**同域：协议、域名、端口号均相同；跨域是指一个域下的文档或脚本试图去请求另一个域下的资源；同源策略由Netscape公司1995年引入浏览器，协助浏览器免受到XSS、CSFR等攻击。所谓同源是指"协议+域名+端口"三者相同，即便两个不同的域名指向同一个ip地址，也非同源。方案主要有①JSONP（淘宝和天猫通过jsonp跨域共享cookie）；②CORS（主流的跨域解决方案）；③document.domain + iframe跨域（只能用于二级域名相同的情况下）；④window.name + iframe跨域（依赖消息通信机制实现但有兼容问题）；⑤nginx代理跨域；⑥nodejs中间件代理跨域；⑦WebSocket协议跨域；⑧Flash代理跨域。

**14. 如何理解重绘和回流？**

**答：**重绘和回流是渲染步骤中的一小节，但是这两个步骤对于性能影响很大。重绘是当节点需要更改外观而不会影响布局的，比如改变 color 就叫称为重绘；回流是布局或者几何属性需要改变就称为回流。 回流必定会发生重绘，重绘不一定会引发回流。回流所需的成本比重绘高的多，改 变深层次的节点很可能导致父节点的一系列回流。

**15.如何设置DNS 预解析？**

**答：**DNS 解析也是需要时间的，可以通过预解析的方式来预先获得域名所对应的 IP。

比如<link rel="dns-prefetch" href="//baidu.cn">

16.对模块化的理解，对于 CommonJS 和 ES6 中的模块化的两者区别是什么？

答：前者支持动态导入，也就是 require(${path}/xx.js)，后者目前不支持，但是已有提案 • 前者是同步导入，因为用于服务端，文件都在本地，同步导入即使卡住主线程 影响也不大。而后者是异步导入，因为用于浏览器，需要下载文件，如果也采 用导入会对渲染有很大影响 • 前者在导出时都是值拷贝，就算导出的值变了，导入的值也不会改变，所以如 果想更新值，必须重新导入一次。但是后者采用实时绑定的方式，导入导出的 值都指向同一个内存地址，所以导入值会跟随导出值变化 • 后者会编译成 require/exports 来执行的

17. call, apply, bind 区别

答：call 和 apply 都是为了解决改变 this 的指向。作用都是相同的，只是传参的方 式不同。 除了第一个参数外，call 可以接收一个参数列表，apply 只接受一个参数数组。需要注意，对于call 和 apply，如果不传入第一个参数，那么默认为 window。

18.手写bind、apply、call

答：

19.ES7中的async 和 await，

答：一个函数如果加上 async ，那么该函数就会返回一个 Promise，可以把 async 看成将函数返回值使用 Promise.resolve() 包裹了下，并且 await 只能在 async 函数中使用。async 和 await 相比直接使用 Promise 来说，优势在于处理 then 的调用链，能 够更清晰准确的写出代码。缺点在于滥用 await 可能会导致性能问题，因为 await 会 阻塞代码，也许之后的异步代码并不依赖于前者，但仍然需要等待前者完成，导致代码 失去了并发性。

20.函数防抖与函数节流

答：如果在频繁的事件回调中做复杂计算，很有可能导致页面卡顿（比如按钮的点击提交），不如将多次计算合并为一次计算，只在一 个精确点做操作。实现思路：对于按钮防点击来说，一旦我开始一个定时器，只要我定时器还在，不 管你怎么点击都不会执行回调函数。一旦定时器结束并设置为 null，就可以再 次点击了；对于延时执行函数来说，每次调用防抖动函数都会判断本次调用和之前 的时间间隔，如果小于需要的时间间隔，就会重新创建一个定时器，并且定时 器的延时为设定时间减去之前的时间间隔。一旦时间到了，就会执行相应的回调函数。防抖动和节流本质是不一样的。防抖动是将多次执行变为最后一次执行，节流是将 多次执行变成每隔一段时间执行。

具体代码可以参考underscore.js 的源码。

21. Proxy

答：