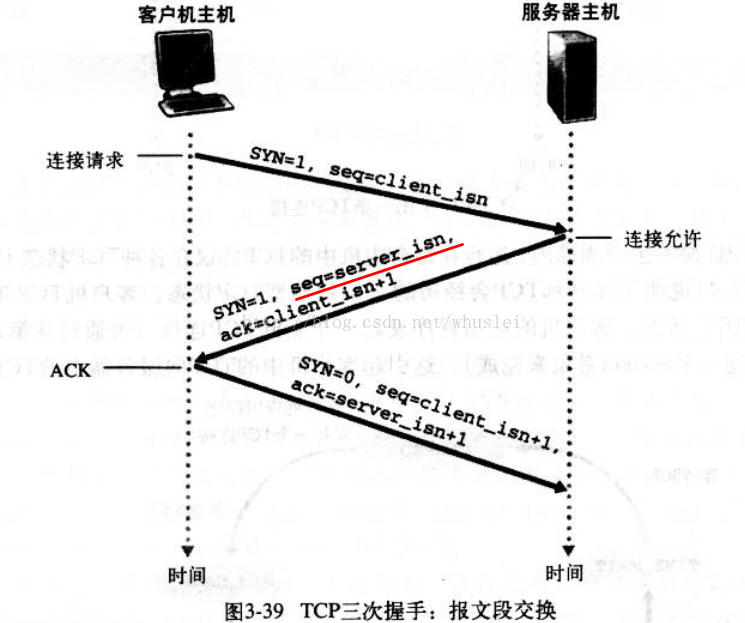
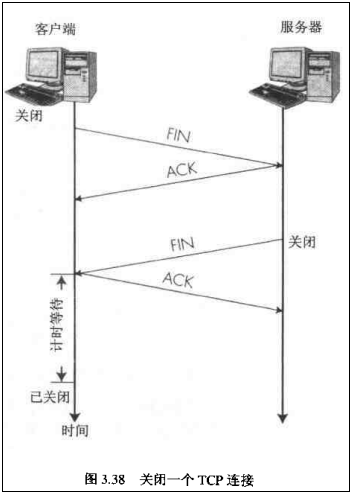
**从输入url到浏览器显示页面发生了什么?**

当用户在浏览器的用户界面的地址栏上url，按下回车。在发起网络请求时会先对URL进行解析，URL是统一资源定位符，顾名思义就是资源所在的位置，它通常由协议.域名.端口号组成。回车之后先解析域名，先依次查找浏览器缓存、操作系统缓存、本地域名服务器缓存，还是没有命中就会去请求根域名服务器解析，根域名服务器给本地域名服务器返回一个所查询域的主域名服务器(gTLD)，5.接受请求的主域名服务器返回这个域名对应的Name Server(网站注册的域名服务器)地址，6.Name Server根据映射关系表找到目标IP，返回本地域名服务器，7.本地域名服务器缓存该域名和对应IP并把解析的结果返回给用户，8.用户根据TTL值缓存到本地系统缓存中。外网IP地址是不会重复的。所以查询出IP地址，就可以找到资源所在的服务器，请求该服务器本地目录下的某资源。向服务器发送请求该资源的报文，将通过三次握手建立TCP连接。客户端发起连接请求，进行第一握手时向服务器发送同步序号SYN = 1，和一个随机的客户端初始序列号seq；服务器收到序列号之后，向客户端发送同步序号SYN = 1，确认号ack = 客户端seq + 1，还有一个随机的服务器序列号，以表示收到了客户机的连接请求并允许客户机的连接请求。最后客户端收到服务器发来的请求确认之后，需要向服务器发送同步序号SYN = 0 ,seq = 客户端本次请求连接发送的初始序列号 + 1，ack = 服务器发送的初始序列号 + 1 。三次握手的目的在于确认客户端和服务器（收发）双方都有收发报文的能力。如果只有两次握手，客户端在某次发送了一个连接请求，但是该请求报文网络延迟了，久久收不到服务器的确认，于是客户端就会重新发送一个新的连接请求，结果网络畅通时服务器就收到了两个请求，建立起两个请求相同资源的请求连接，这样就会有一个连接浪费了服务器资源，达到一个效果多做了一次无用功。建立连接之后就发送Http请求



建立TCP连接需要三次握手，断开TCP连接需要四次挥手。首先客户端发送FIN报文请求中断，“我要和你断连”，服务端收到FIN报文时服务端的数据也许还没有发送完毕，服务端先回复确认“收到，我知道了，先等我一下”，等服务端的数据发送完毕了，他才会发送一个FIN报文同意断连，“好了，拜拜”，客户端确认断连，“拜拜啦”。

****

**Web缓存是什么？**Web缓存是可以自动保存常见文档副本的HTTP设备。当Web请求抵达缓存时，如果有本地缓存的副本，就有机会（还要判断是否可用）从本地缓存中提取这个文档。

**这样做有什么好处？**1.**冗余的数据传输**。很多客户端频繁访问同一个服务器页面时，服务器会多次传输同一个文档，相同的字节在网络中重复传输，这些冗余的数据传输会消耗网络带宽、降低传输速度、加重Web服务器的负载。保留缓存副本就可以减少这些损失。2.**带宽瓶颈。**很多网络为本地客户端提供的带宽远宽于提供给远程服务器的带宽。如果客户端从一个快速局域网的缓存中得到一个副本就可以提高性能了。3.**瞬间拥塞**。很多人同时向服务器访问一个Web文档就会给服务器造成巨大的压力，缓存减轻了服务器的压力。4.**距离时延。**即使客户端和浏览器之间没有太多的路由器，光速也会造成显著的时延。将缓存放在附近就会减少时延。

缓存的处理步骤：1.缓存接受http请求。2.缓存解析该报文，提取URL和各种首部。3.缓存查看本地是否有副本，如果没有就去请求源服务器获取一份副本并保存在本地。4.有了缓存副本还不行，要检查副本的有效性。HTTP1.0的产物expires字段是服务器响应头字段，它给出了一个副本过期的绝对时间。由于是绝对时间，用户可能会将客户端本地时间进行修改，从而导致浏览器判断缓存失效。已知Expires的缺点之后，HTTP1.1新增了Cache-control字段。常见的Cache-control的值有以下：

1. max-age：定义了文档的最大有效时间。以时间间隔标志失效时间，解决了Expires服务器和客户端相对时间的问题。
2. no-cache：不强缓存，要服务器再验证（协商缓存）
3. no-store：绝对禁止所有的缓存
4. public：客户端和中间代理服务器都能缓存
5. private：只能客户端缓存

Cache-control的优先级高于Expires。

缓存再验证（协商缓存）中使用两个条件首部。

1. If-Modified-Since:<date> ：可以和服务器响应首部Last-Modified配合使用，原始服务器会将最后修改日期附加到所提供的文档，如果从指定日期之后文档被修改过，就要向服务器请求。
2. If-None-Match:<tags>：服务器可以为文档提供特殊的标签标志该文档，如果已缓存的标签和服务器文档标签不一样，就要向服务器请求。

有些情况下使用If-Modified-Since进行再验证是不够的。比如

* 更新间隙发生在一秒粒度单位以下。
* 有些文档被周期性重写，即使内容无变化
* 有些修改不重要。不需要让缓存替换副本。

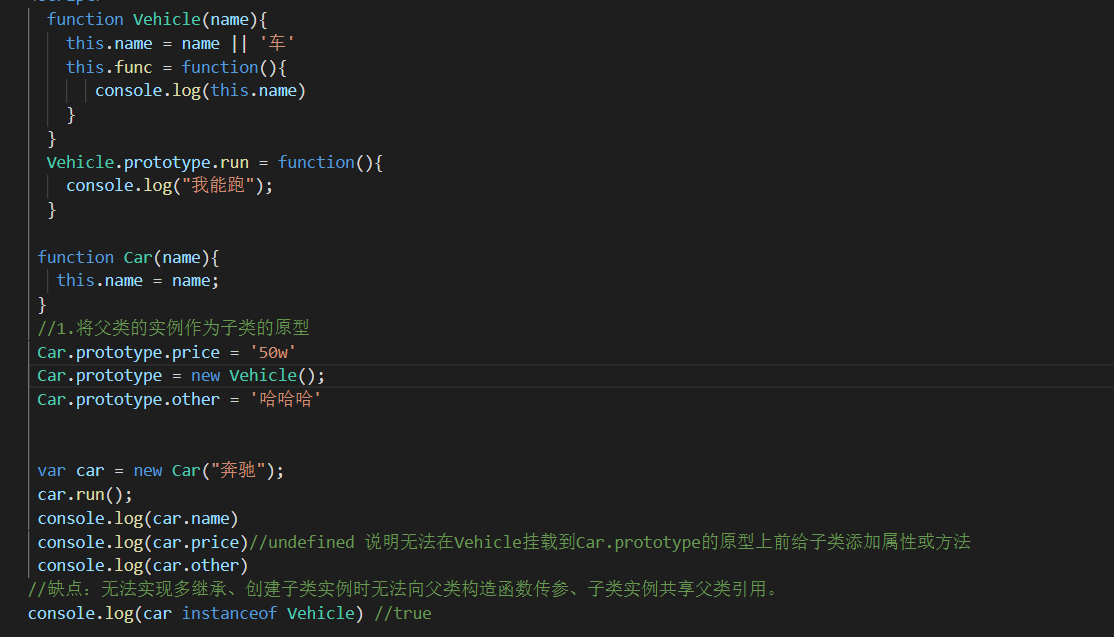
**JavaScript采用的是单线程模式**，也就是说一次只能做一件事。虽然H5中的Web Worker为JavaScript创建了多线程环境，但是子线程完全受主线程控制，而且不能操作DOM元素。这没有改变JavaScript的单线程机制，打个比方，如果JavaScript是多线程的，一个线程要删除一DOM元素，而另一线程要往这个DOM元素添加内容。那引擎该听谁的呢？JavaScript只能单线程，所以 JavaScript的所有任务都需要排队完成。JavaScript的任务又分为同步任何和异步任务。同步任务直接放在主线程中排队以此执行，所有的同步任务在主线程中执行，异步任务会放到异步队列里。调用栈是JavaScript中的一个数据结构，当函数执行时，会把函数的变量、参数、方法入栈。当主线程中的同步任务执行完，先在宏任务队列里取出一个宏任务放入调用栈中让主线程执行调用栈的任务，当执行过程中遇到微任务时，把微任务放入微任务队列里，宏任务执行完毕之后立即执行当前微任务队列中的所有微任务，以此类推。常见的宏任务有setTimeout、script、setInterval，微任务有Promise.then(),catch finally,process.nextTick。那么**为什么要区分宏任务和微任务**呢？个人理解，区分这两种任务是为了插队。因为Event Loop中微任务队列里所有任务执行完毕才能进行下一个Event Loop去执行一个宏任务。当微任务在执行过程中又碰到一个微任务，这依然会在本次的Event Loop中执行，所以宏任务得等。

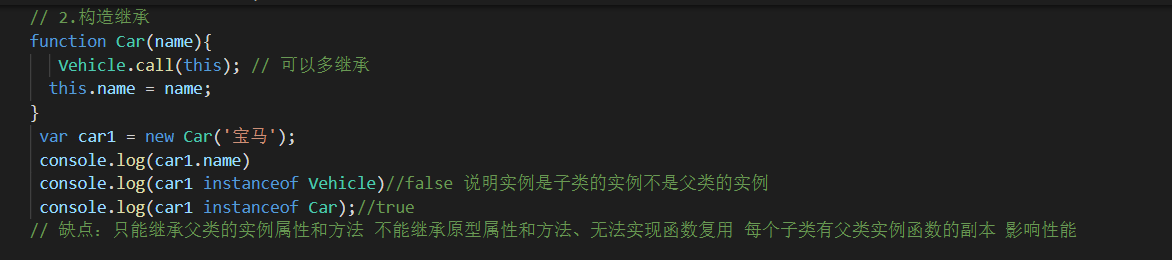
JavaScript执行过程分为解析阶段和运行阶段。解析阶段分为词法分析和语法分析。词法分析就是把代码中的字符串分割成token（词法单元）；语法分析即是将token流转换成一棵抽象语法树。在JS代码正式运行之前还会进行预解析的工作，比如进行变量提升、函数提升。这使得我们可以在变量赋值前使用变量，在函数声明之上调用函数。

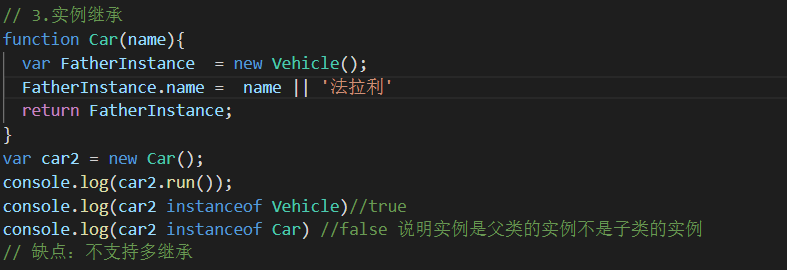
JavaScript也是一门基于面向对象思想创造的语言，它也有很多内置函数，比如用于解决异步编程回调地狱问题的Promise、或者Number、Boolean、Error、Date等等。如今于前端开发的http库——axios 的原理也是基于Promise实现的，还有Vue中会使用到各种实例化对象。但它并不是一门真正的面向对象语言，OOP的三大特性是：封装、继承和多态。前端的封装和后端的封装目的都是为了减少重复的劳动，把功能相似的封装起来，后续用到时直接引入调用。JAVA的继承意味着拷贝超类的所有公有的属性和方法，而JavaScript的继承是原型链上的关系，一个对象不会复制到其它对象，它们会被关联起来。prototype是Javascript中继承关系的联结点，当我们用.(点)语法或者getProperty访问子类实例的一个属性时，实际上，会先在该子类实例本身找，如果找不到，就通过\_\_proto\_\_这个纽带往prototype(原型对象)上就近原则查找，原型链的终点是Object.prototype == null，所有普通的prototype链都会指向内置的Object.prototype。调用Object.create()即凭空创建一个新对象并把它的proptotype和指定对象关联。Java的多态分为重载和重写两种，Java中的重载就是说，同名方法，不同的参数列表。JavaScript中没有重载的概念，同名方法会按赋值的顺序覆盖，用的是覆盖后的方法。Java的重写是子类重写父类的同名同参数列表的方法，子类实例化的对象调用同名方法时只用自己的方法，JavaScript的重写也许是改变原型链上原型对象的方法。

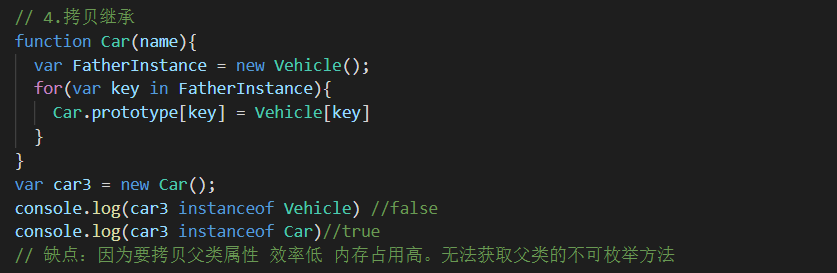
**JavaScript的基本数据类型**有：undefined、number、null、string、boolean、bigInt、Symbol。基本数据类型变量放在栈内存里，而且栈内存的大小是固定的，如果存储的大小大于栈内存大小就会溢栈。因为栈有先入后出的特点，所以方便查找基本数据类型变量。而引用类型数据放在堆内存中，堆内存是人为new分配的且人为释放的，运行效率低。栈内存的数据运行结束就被系统自动回收。堆内存的对象回收标准是否可达，在V8中对象先分配到新生代的From中，如果不可达直接释放，如果可达，就复制到To中，然后将To和From互换。当多次复制后依然没有回收，就放入老生代中，进行标记回收。之后将内存碎片进行整合放到一端。

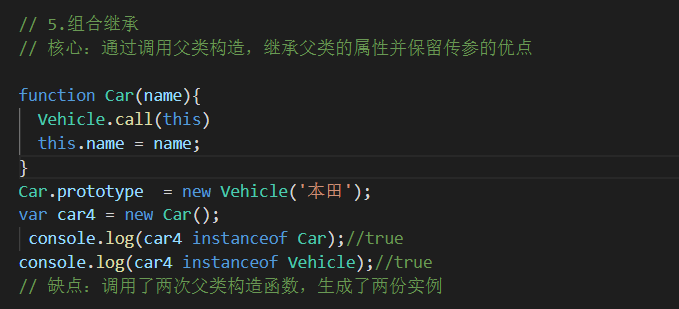
**JavaScript中实现继承**的方式又有哪些呢？

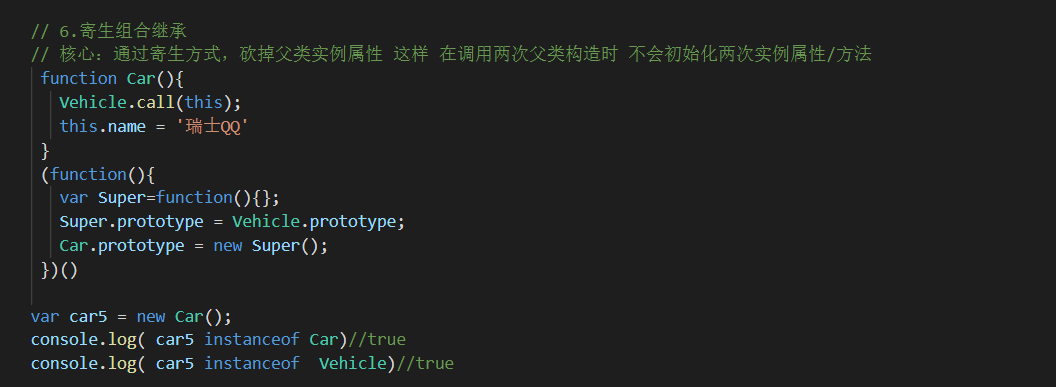












谈谈JavaScript学习过程中常被提及的闭包。说到闭包要了解**函数作用域**链。JavaScript中每一个方法里都有一个内置属性[[scope]]，它由JavaScript引擎存储管理，其实这个[[scope]]属性里也就是它的作用域链。一个function在定义的时候作用域链的顶端是它所处在的外部作用域，当function执行时，function作用域链的顶端加入一个自己本身的作用域，执行完后又把这个作用域销毁。这就是作用域链。一个外部函数返回一个引用外部函数变量的内部函数，当外部函数执行完毕，作用域链销毁，但内部函数还引用着对外部函数的活动对象，这就形成了**闭包**。**闭包的优点**：1.可以从内部访问到外部变量，且这个变量常驻内存，可供之后使用。2.JavaScript在多人合作时，如果定义过多的全局变量，有可能会造成全局变量命名冲突，使用闭包可以避免变量污染全局。3.把变量存在独立的作用域，作为私有变量存在。**闭包的缺点**：1.有可能会获取到意外的值。2.因保持着对外部作用域活动变量的引用，造成内存泄露。3.闭包的层级决定了引用的外部变量在查找时经过的作用域链长度，层级太深对处理速度有影响。**闭包的应用场景一：**JavaScript设计模式中的模块模式，就是利用闭包封装私有状态和私有组织，对外只提供一个公共的API，其它的一切维持在私有闭包里。**闭包的应用场景二：**循环中创建闭包，防止取到意外的值。

**如何判断this绑定？**判断this绑定需要知道函数的调用位置，分析调用栈，我们关心的调用位置就在当前正在执行的函数的前一个调用中。有如下四种绑定规则：一个是**默认绑定**。独立函数调用时采用默认绑定规则，严格模式下默认绑定undefined，非严格模式下默认绑定window。一个是**隐式绑定**。当函数引用有上下文对象时，隐式绑定会把this绑定到这个上下文对象中。有时候会发生丢失绑定对象的现象，丢失绑定对象就会采用默认绑定。一个是**显式绑定**。 当我们不想隐式绑定（在对象中包含函数引用），想让某个对象中强制调用某个函数，就可以采用显式绑定的方法。使用call(),apply()。还有一种叫**new绑定**。在Java中，使用new初始化一个类时会调用类的构造函数。而JavaScript中,new不会实例化一个类，只是用new操作符调用这一个函数，它不是传统的构造函数。当使用new调用函数时会发生如下：首先，会构造一个全新对象。接着，这个新对象会被原型连接到调用的函数。然后这个新对象被绑定到函数调用的this。最后，如果函数没有返回别的对象，new表达式中的函数调用会自动返回这个新对象。**This绑定的优先级**：4321。By the way,箭头函数中的并不采用如上四种绑定规则，因为箭头函数没有自己的this,它的this继承自外层函数调用的this。

箭头函数和普通函数的区别除了上述的this指向之外，还有以下几点：1.箭头函数没有argument属性。2.箭头函数不能作为构造器，因此不能通过new来调用，所以它没有new.target属性。3.不能使用call，apply,bind，通过显式绑定修改箭头函数的this指向。4.箭头函数不能作为generator函数。

什么是回调地狱？说起回调地狱，首先要知道什么是异步，举个例子，当我们去肯德基点餐，点餐结束拿到单据之后我们不需要傻愣着等待出餐，可以坐着玩会手机，等服务员叫号了再去取餐。简言之就是发起者不关心事件的状态，不需要等待接收者返回信息。我们常说的回调地狱就是一个异步事件嵌套着另一个异步事件，一个异步事件依赖于另一个异步事件，