**一、TCP（Transmission Control Protocol）**

    TCP传输控制协议，是一种面向连接的、可靠的、基于字节流的传输层通信协议。通讯双方建立一次tcp连接，需要经过三次步骤1、客户端发送syn包（syn=j）到服务器，并入SYN\_SEND状态，等待服务器确认。2、服务器收入syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），既SYN+ACK包，此时服务器进入SYN\_RECV状态。3、客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK（ack=k+1），此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。（tcp在握手过程中并不携带数据，而是在三次握手完成之后才会进行数据传送）

SYN：synchronous 建立联机

ACK：acknowledgement 确认

SYN\_SENT 请求连接

SYN\_RECV 服务端被动打开后，接收到了客户端的SYN并且发送了ACK时的状态，再进一步接收到客户端的ACK就进入ESTABISHED状态。

UDP（User Datagram Protocol ）用户数据报协议

UDP是非面向连接协议，使用udp协议通讯并不需要建立连接，它只负责把数据尽可能发送出去，并不可靠，在接收端，UDP把每个消息断放入队列中，接收端程序从队列中读取数据。

TCP/IP是位于传输层上的一种协议，用于在网络中传输数据；

**二、socket（嵌套字）**

    socket是一组实现TCP/UDP通信的接口API，既无论TCP还是UDP，通过对scoket的编程，都可以实现TCP/UCP通信。（TCP或UDP是一种计算机网络通信中在传输层的一种协议，可以简单的理解成是一种约定，只有履行合同才是实质性的行动，所以无论是TCP还是UDP要产生作用，都需要有实际性的行为去执行才能体现协议的作用。socket就是实现这种作用的方法）socket作为一个通信链的句柄，它包含了网络通信必备的5种信息。1、连接使用的协议。2、本地主机的IP地址。3、本地进程的协议端口。4、远地主机的IP地址。5、远地进程的协议端口。即可知道，socket包含了通信本方和对方的ip和端口以及连接使用的协议（TCP/UDP）。通信双方中的一方（客户端）通过socket对另一方（服务端）发起连接请求，服务端在网络上监听请求，当收到客户端发来的请求之后，根据socket里携带的信息，定位到客户端，就相应请求，把socket描述发给客户端，双方确认之后连接就建立了。

socket连接过程的三个步骤

1、服务器监听：服务器实时监控网络状态等待客户端发来的连接请求

2、客户端请求：客户端根据远程主机服务器的IP地址和协议端口向其发起连接请求

3、连接确认：服务端收到socket的连接请求之后，就响应请求，把服务端socket描述发给客户端，客户端收到后一但确认。则双方建立连接，进行数据交互。

socket连接一旦建立就保持连接状态，而HTTP连接则不一样，它基于tcp协议的短连接，也就是客户端发起请求，服务器响应请求之后，连接就会自动断开，不会一直保持。

**三、URL（Uniform Resource Locator）统一资源定位符**

URL是因特网上标准的资源地址

URL标准格式：协议类型：[//服务器地址[:端口号]][/资源层级UNIX文件路径]文件名？查询

URL的语法规则：scheme：//host/domain:port/path/filename  (1、scheme：定义因特网服务的类型，最常见的有http。2、host：定义域主机（http默认主机是www）。3、domain：定义因特网域名，比如“[www.baidu.com”。4、path：定义服务器上的路径。5、filename：资源名)](http://www.imooc.com/article/271528)

scheme类型

1、http 超文本传输协议：以http：//开头的不同网页，不加密

2、https 安全超文本传输协议。安全网页，加密所有信息交换

3、ftp 文件传输协议 用于将文件下载或上传

4、file 本地文件

HTTP协议是应用层协议，基于TCP协议，用于包装数据，程序使用它进行通信，可以简单高效的处理通信中数据的传输和识别处理

**四、DNS（Domain Name Server）域名服务器**

DNS是进行域名(domain name)和与之相对应的IP地址 (IP address)转换的服务器。DNS中保存了一张域名(domain name)和与之相对应的IP地址 (IP address)的表，以解析消息的域名。

在浏览器输入域名后的解析过程

浏览器根据地址去本身缓存中查找dns解析记录，如果有，则直接返回IP地址，否则浏览器会查找操作系统中（hosts文件）是否有该域名的dns解析记录，如果有则返回。

如果浏览器缓存和操作系统hosts中均无该域名的dns解析记录，或者已经过期，此时就会向域名服务器发起请求来解析这个域名。

请求会先到LDNS（本地域名服务器），让它来尝试解析这个域名，如果LDNS也解析不了，则直接到根域名解析器请求解析

根域名服务器给LDNS返回一个所查询余的主域名服务器（gTLDServer）地址。

此时LDNS再向上一步返回的gTLD服务器发起解析请求。

gTLD服务器接收到解析请求后查找并返回此域名对应的Name Server域名服务器的地址，这个Name Server通常就是你注册的域名服务器（比如阿里dns、腾讯dns等）

Name Server域名服务器会查询存储的域名和IP的映射关系表，正常情况下都根据域名得到目标IP记录，连同一个TTL值返回给DNS Server域名服务器

返回该域名对应的IP和TTL值，Local DNS Server会缓存这个域名和IP的对应关系，缓存的时间有TTL值控制。

把解析的结果返回给用户，用户根据TTL值缓存在本地系统缓存中，域名解析过程结束。

**五、HTTP请求发起和响应**

在一个web程序开发中，一般都有前端和后端之分，前端负责向后端请求数据和展示页面，后端负责接收请求和做出响应发回给前端，他们之间的协作桥梁是API，而API其实就是一个URL，作为HTTP连接的一种具体载体。

用户输入URL到浏览器显现给用户页面经过了什么过程

用户输入URL，浏览器获取到URL

浏览器(应用层)进行DNS解析（直接输入IP地址既跳过该步骤）

根据解析出的IP地址+端口，浏览器（应用层）发起HTTP请求，请求中携带（请求头header（也可细分为请求行和请求头）、请求体body）

请求到达传输层，tcp协议为传输报文提供可靠的字节流传输服务，它通过三次握手等手段来保证传输过程中的安全可靠。通过对大块数据的分割成一个个报文段的方式提供给大量数据的便携传输。

到网络层， 网络层通过ARP寻址得到接收方的Mac地址，IP协议把在传输层被分割成一个个数据包传送接收方。

数据到达数据链路层，请求阶段完成

接收方在数据链路层收到数据包之后，层层传递到应用层，接收方应用程序就获得到请求报文。

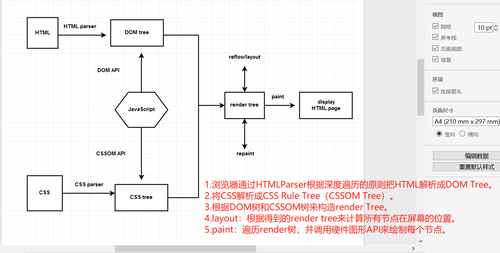
接收方收到发送方的HTTP请求之后，进行请求文件资源（如HTML页面）的寻找并响应报文

发送方收到响应报文后，如果报文中的状态码表示请求成功，则接受返回的资源（如HTML文件），进行页面渲染。

（header：1、请求的方法（get、post、put..）2、协议（http、https、ftp、sftp…）3目标url（具体的请求路径已经文件名）4一些必要信息（缓存、cookie之类）。)

（body包含请求的内容）

**六、页面渲染过程**



内容解释

HTML parser：HTML解析器，其本质是将HTML文本解释成DOM tree。

CSS  parser：CSS解析器，其本质是讲DOM中各元素对象加入样式信息

JavaScript引擎：专门处理JavaScript脚本的虚拟机，其本质是解析JS代码并且把逻辑（HTML和CSS的操作）应用到布局中，从而按程序要的要求呈现相应的结果

DOM tree:文档对象模型树，也就是浏览器通过HTMLparser解析HTML页面生成的HTML树状结构以及相应的接口。

render tree：渲染树，也就是浏览器引擎通过DOM Tree和CSS Rule Tree构建出来的一个树状结构，和dom tree不一样的是，它只有要最终呈现出来的内容，像或者带有display:none的节点是不存在render tree中的。

layout：也叫reflow 重排，渲染中的一种行为。当rendertree中任一节点的几何尺寸发生改变了，render tree都会重新布局。

repaint：重绘，渲染中的一种行为。render tree中任一元素样式属性（几何尺寸没改变）发生改变了，render tree都会重新画，比如字体颜色、背景等变化。

**七、href与src**

href (Hypertext Reference)指定网络资源的位置，从而在当前元素或者当前文档和由当前属性定义的需要的锚点或资源之间定义一个链接或者关系。（目的不是为了引用资源，而是为了建立联系，让当前标签能够链接到目标地址。）

src source（缩写），指向外部资源的位置，指向的内容将会应用到文档中当前标签所在位置。

href与src的区别

1、请求资源类型不同：href 指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的联系。在请求 src 资源时会将其指向的资源下载并应用到文档中，比如 JavaScript 脚本，img 图片；

2、作用结果不同：href 用于在当前文档和引用资源之间确立联系；src 用于替换当前内容；

3、浏览器解析方式不同：当浏览器解析到src ，会**暂停其他资源的下载和处理，**直到将该资源加载、编译、执行完毕，图片和框架等也如此，类似于将所指向资源应用到当前内容。这也是为什么建议把 js 脚本放在底部而不是头部的原因。

**八、link和@import的区别**

两者都是外部引用 CSS 的方式，但是存在一定的区别：

（1）link是XHTML标签，除了能够加载CSS，还可以定义RSS等其他事务；而@import属于CSS范畴，只可以加载CSS。

（2）link引用CSS时，在页面载入时同时加载；@import需要页面完全载入以后再加载。

（3）link是XHTML标签，无兼容问题；@import则是在CSS2.1提出的，低版本的浏览器不支持。

（4）link支持使用Javascript控制DOM改变样式；而@import不支持。

**九、HTML**

1.HTML5的新特性：

新的内容标签：header nav content footer article aside

更好的单元格体系:

音频、视频API:video radio

画布(Canvas) API

地理(Geolocation) API

网页存储(Web storage) API:localStorage,sessionStorage

拖拽释放(Drag and drop) API

2.Doctype作用？标准模式与兼容模式各有什么区别?

<!DOCTYPE>声明位于位于HTML文档中的第一行，处于 <html> 标签之前。告知浏览器的解析器用什么文档标准解析这个文档。DOCTYPE不存在或格式不正确会导致文档以兼容模式呈现。

3.行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？

首先：CSS规范规定，每个元素都有display属性，确定该元素的类型，每个元素都有默认的display值，如div的display默认值为“block”，则为“块级”元素；span默认display属性值为“inline”，是“行内”元素。

（1）行内元素有：a b span img input select strong（强调的语气）

（2）块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p

（3）常见的空元素：    <br> <hr> <img> <input> <link> <meta>

    鲜为人知的是：    <area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>

不同浏览器（版本）、HTML4（5）、CSS2等实际略有差异

4.介绍一下你对浏览器内核的理解？

主要分成两部分：渲染引擎(layout engineer或Rendering Engine)和JS引擎。

渲染引擎：负责取得网页的内容（HTML、XML、图像等等）、整理讯息（例如加入CSS等），以及计算网页的显示方式，然后会输出至显示器或打印机。浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不相同。所有网页浏览器、电子邮件客户端以及其它需要编辑、显示网络内容的应用程序都需要内核。

JS引擎则：解析和执行javascript来实现网页的动态效果。

最开始渲染引擎和JS引擎并没有区分的很明确，后来JS引擎越来越独立，内核就倾向于只指渲染引擎。

常见内核

Trident内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称MSHTML]

Gecko内核：Netscape6及以上版本，FF,MozillaSuite/SeaMonkey等

Presto内核：Opera7及以上。      [Opera内核原为：Presto，现为：Blink;]

Webkit内核：Safari,Chrome等。   [ Chrome的：Blink（WebKit的分支）]

5、请描述一下 cookies，sessionStorage 和 localStorage 的区别？

cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。

cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递。

sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

存储大小：

    cookie数据大小不能超过4k。

    sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。

有期时间：

    localStorage    存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；

    sessionStorage  数据在当前浏览器窗口关闭后自动删除。

    cookie          设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭

6.iframe有那些缺点？

iframe会阻塞主页面的Onload事件；

搜索引擎的检索程序无法解读这种页面，不利于SEO;

iframe和主页面共享连接池，而浏览器对相同域的连接有限制，所以会影响页面的并行加载。

使用iframe之前需要考虑这两个缺点。如果需要使用iframe，最好是通过javascript动态给iframe添加src属性值，这样可以绕开以上两个问题

7.如何实现浏览器内多个标签页之间的通信? (阿里)

WebSocket、SharedWorker；

也可以调用localstorge、cookies等本地存储方式；

localstorge另一个浏览上下文里被添加、修改或删除时，它都会触发一个事件，

我们通过监听事件，控制它的值来进行页面信息通信；

注意quirks：Safari 在无痕模式下设置localstorge值时会抛出 QuotaExceededError 的异常

8.HTML5的离线储存怎么使用，工作原理能不能解释一下？

在用户没有与因特网连接时，可以正常访问站点或应用，在用户与因特网连接时，更新用户机器上的缓存文件。

原理：HTML5的离线存储是基于一个新建的.appcache文件的缓存机制(不是存储技术)，通过这个文件上的解析清单离线存储资源，这些资源就会像cookie一样被存储了下来。之后当网络在处于离线状态下时，浏览器会通过被离线存储的数据进行页面展示。

如何使用：

1、页面头部像下面一样加入一个manifest的属性；

2、在cache.manifest文件的编写离线存储的资源；

    CACHE MANIFEST

    #v0.11

    CACHE:

    js/app.js

    css/style.css

    NETWORK:

    resourse/logo.png

    FALLBACK:

    / /offline.html

3、在离线状态时，操作window.applicationCache进行需求实现。

**十、CSS**

1.CSS选择符有哪些？哪些属性可以继承？

\*   1.id选择器（ # myid）

    2.类选择器（.myclassname）

    3.标签选择器（div, h1, p）

    4.相邻选择器（h1 + p）

    5.子选择器（ul > li）

    6.后代选择器（li a）

    7.通配符选择器（ \* ）

    8.属性选择器（a[rel = "external"]）

    9.伪类选择器（a:hover, li:nth-child）

\*   可继承的样式： font-size font-family color, UL LI DL DD DT;

\*   不可继承的样式：border padding margin width height ;

2.CSS优先级算法如何计算？

\*   优先级就近原则，同权重情况下样式定义最近者为准;

\*   载入样式以最后载入的定位为准;

优先级为:

    同权重: 内联样式表（标签内部）> 嵌入样式表（当前文件中）> 外部样式表（外部文件中）。

    !important >  id > class > tag

    important 比 内联优先级高

3.CSS3新增伪类有那些？

  举例：

    p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。

    p:last-of-type  选择属于其父元素的最后 <p> 元素的每个 <p> 元素。

    p:only-of-type  选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。

    p:only-child        选择属于其父元素的唯一子元素的每个 <p> 元素。

    p:nth-child(2)  选择属于其父元素的第二个子元素的每个 <p> 元素。

    :after          在元素之前添加内容,也可以用来做清除浮动。

    :before         在元素之后添加内容

    :enabled

    :disabled       控制表单控件的禁用状态。

    :checked        单选框或复选框被选中。

4.如何居中div？

1、水平居中：给div设置一个宽度，然后添加margin:0 auto属性

div{

    width:200px;

    margin:0 auto;

 }

2、水平垂直居中一

确定容器的宽高 宽500 高 300 的层

设置层的外边距

div {

    position: relative;     /\* 相对定位或绝对定位均可 \*/

    width:500px;

    height:300px;

    top: 50%;

    left: 50%;

    margin: -150px 0 0 -250px;      /\* 外边距为自身宽高的一半 \*/

    background-color: pink;     /\* 方便看效果 \*/

 }

3、水平垂直居中二

未知容器的宽高，利用 `transform` 属性

div {

    position: absolute;     /\* 相对定位或绝对定位均可 \*/

    width:500px;

    height:300px;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

    background-color: pink;     /\* 方便看效果 \*/

}

4、水平垂直居中三

利用 flex 布局

实际使用时应考虑兼容性

.container {

    display: flex;

    align-items: center;        /\* 垂直居中 \*/

    justify-content: center;    /\* 水平居中 \*/

}

.container div {

    width: 100px;

    height: 100px;

    background-color: pink;     /\* 方便看效果 \*/

}

5.CSS3有哪些新特性？

 新增各种CSS选择器  （: not(.input)：所有 class 不是“input”的节点）

  圆角            （border-radius:8px）

  多列布局      （multi-column layout）

  阴影和反射   （Shadow\Reflect）

  文字特效      （text-shadow、）

  文字渲染      （Text-decoration）

  线性渐变      （gradient）

  旋转            （transform）

  缩放,定位,倾斜,动画,多背景

  例如:transform:\scale(0.85,0.90)\ translate(0px,-30px)\ skew(-9deg,0deg)\Animation:

6.请解释一下CSS3的Flexbox（弹性盒布局模型）,以及适用场景？

 一个用于页面布局的全新CSS3功能，Flexbox可以把列表放在同一个方向（从上到下排列，从左到右），并让列表能延伸到占用可用的空间。

 较为复杂的布局还可以通过嵌套一个伸缩容器（flex container）来实现。

 采用Flex布局的元素，称为Flex容器（flex container），简称"容器"。

 它的所有子元素自动成为容器成员，称为Flex项目（flex item），简称"项目"。

 常规布局是基于块和内联流方向，而Flex布局是基于flex-flow流可以很方便的用来做局中，能对不同屏幕大小自适应。

 在布局上有了比以前更加灵活的空间。

7.为什么要初始化CSS样式

- 因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。

- 当然，初始化样式会对SEO有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。

最简单的初始化方法： \* {padding: 0; margin: 0;} （强烈不建议）

淘宝的样式初始化代码：

body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td { margin:0; padding:0; }

body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }

h1, h2, h3, h4, h5, h6{ font-size:100%; }

address, cite, dfn, em, var { font-style:normal; }

code, kbd, pre, samp { font-family:couriernew, courier, monospace; }

small{ font-size:12px; }

ul, ol { list-style:none; }

a { text-decoration:none; }

a:hover { text-decoration:underline; }

sup { vertical-align:text-top; }

sub{ vertical-align:text-bottom; }

legend { color:#000; }

fieldset, img { border:0; }

button, input, select, textarea { font-size:100%; }

table { border-collapse:collapse; border-spacing:0; }

**十一、js**

js的基本数据类型。

 Undefined、Null、Boolean、Number、String、

 ECMAScript 2015 新增:Symbol(创建后独一无二且不可变的数据类型 )

2.JavaScript原型，原型链 ? 有什么特点？

每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，

如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，

于是就这样一直找下去，也就是我们平时所说的原型链的概念。

关系：instance.constructor.prototype = instance.\_\_proto\_\_

特点：

JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变。

 当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，

 就会查找他的Prototype对象是否有这个属性，如此递推下去，一直检索到 Object 内建对象。    function Func(){}

    Func.prototype.name = "Sean";

    Func.prototype.getInfo = function() {      return this.name;

    }    var person = new Func();//现在可以参考var person = Object.create(oldObject);

    console.log(person.getInfo());//它拥有了Func的属性和方法

    //"Sean"    console.log(Func.prototype);    // Func { name="Sean", getInfo=function()}

3.如何实现数组的随机排序？

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | 方法一：  ```javascript    　　var arr = [1,2,3,4,5,6,7,8,9,10];       function randSort1(arr){          for(var i = 0,len = arr.length;i < len; i++ ){              var rand = parseInt(Math.random()\*len);              var temp = arr[rand];              arr[rand] = arr[i];              arr[i] = temp;          }         return arr;       }       console.log(randSort1(arr));    ```  方法二：  ```javascript        var arr = [1,2,3,4,5,6,7,8,9,10];        function randSort2(arr){            var mixedArray = [];            while(arr.length > 0){                var randomIndex = parseInt(Math.random()\*arr.length);                  mixedArray.push(arr[randomIndex]);                  arr.splice(randomIndex, 1);              }            return mixedArray;          }          console.log(randSort2(arr));    ```  方法三：  ```javascript        var arr = [1,2,3,4,5,6,7,8,9,10];          arr.sort(function(){            return Math.random() - 0.5;          })          console.log(arr); |

4.Javascript如何实现继承？

1、构造继承2、原型继承3、实例继承4、拷贝继承

原型prototype机制或apply和call方法去实现较简单，建议使用构造函数与原型混合方式。

```javascript        function Parent(){            this.name = 'wang';

        }        function Child(){            this.age = 28;

        }

        Child.prototype = new Parent();//继承了Parent，通过原型

        var demo = new Child();

        alert(demo.age);

        alert(demo.name);//得到被继承的属性

5.javascript创建对象的几种方式？

javascript创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用JSON；但写法有很多种，也能混合使用。1、对象字面量的方式

    person={firstname:"Mark",lastname:"Yun",age:25,eyecolor:"black"};2、用function来模拟无参的构造函数    function Person(){}    var person=new Person();//定义一个function，如果使用new"实例化",该function可以看作是一个Class

    person.name="Mark";

    person.age="25";

    person.work=function(){

    alert(person.name+" hello...");

    }

    person.work();3、用function来模拟参构造函数来实现（用this关键字定义构造的上下文属性）    function Pet(name,age,hobby){       this.name=name;//this作用域：当前对象

       this.age=age;       this.hobby=hobby;       this.eat=function(){

          alert("我叫"+this.name+",我喜欢"+this.hobby+",是个程序员");

       }

    }    var maidou =new Pet("麦兜",25,"coding");//实例化、创建对象

    maidou.eat();//调用eat方法4、用工厂方式来创建（内置对象）     var wcDog =new Object();

     wcDog.name="旺财";

     wcDog.age=3;

     wcDog.work=function(){

       alert("我是"+wcDog.name+",汪汪汪......");

     }

     wcDog.work();5、用原型方式来创建    function Dog(){

     }

     Dog.prototype.name="旺财";

     Dog.prototype.eat=function(){

     alert(this.name+"是个吃货");

     }     var wangcai =new Dog();

     wangcai.eat();5、用混合方式来创建    function Car(name,price){      this.name=name;      this.price=price;

    }

     Car.prototype.sell=function(){

       alert("我是"+this.name+"，我现在卖"+this.price+"万元");

      }    var camry =new Car("凯美瑞",27);

    camry.sell();

6.闭包

闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。

闭包的特性：1.函数内再嵌套函数2.内部函数可以引用外层的参数和变量3.参数和变量不会被垃圾回收机制回收//li节点的onclick事件都能正确的弹出当前被点击的li索引

 <ul id="testUL">

    <li> index = 0</li>

    <li> index = 1</li>

    <li> index = 2</li>

    <li> index = 3</li>

</ul>

<script type="text/javascript">    var nodes = document.getElementsByTagName("li");    for(i = 0;i<nodes.length;i+= 1){

        nodes[i].onclick = (function(i){                  return function() {

                     console.log(i);

                  } //不用闭包的话，值每次都是4                })(i);

    }</script>执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在

使得Javascript的垃圾回收机制GC不会收回say667()所占用的资源

因为say667()的内部函数的执行需要依赖say667()中的变量

这是对闭包作用的非常直白的描述  function say667() {    // Local variable that ends up within closure

    var num = 666;    var sayAlert = function() {

        alert(num);

    }

    num++;    return sayAlert;

} var sayAlert = say667();

 sayAlert()//执行结果应该弹出的667

7.Ajax 是什么? 如何创建一个Ajax？

ajax的全称：Asynchronous Javascript And XML。

异步传输+js+xml。

所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。

(1)创建XMLHttpRequest对象,也就是创建一个异步调用对象

(2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息

(3)设置响应HTTP请求状态变化的函数

(4)发送HTTP请求

(5)获取异步调用返回的数据

(6)使用JavaScript和DOM实现局部刷新

作者：雷灵初心  
链接：http://www.imooc.com/article/271528