目录

[JS基础知识 1](#_Toc10056270)

[1. 闭包 1](#_Toc10056271)

[2.原型 2](#_Toc10056272)

[3.call() 和 .apply() 的作用和区别？ 2](#_Toc10056273)

[4.事件 2](#_Toc10056274)

[6.了解浏览器渲染页面原理 2](#_Toc10056275)

[7.浏览器从输入URL到渲染出页面发生了什么 4](#_Toc10056276)

[（1）假设是简单的http请求（GET），IPV4，无代理。 4](#_Toc10056277)

[.DOM操作——怎样添加、移除、移动、复制、创建和查找节点? 5](#_Toc10056278)

[数组对象有哪些原生方法， 7](#_Toc10056279)

[赋值方法 (Mutator methods): 7](#_Toc10056280)

[访问方法(Accessor methods) : 这些方法只是返回相应的结果，而不会修改数组本身 ; 7](#_Toc10056281)

[迭代方法（Iteration methods）: 8](#_Toc10056282)

[CSS 8](#_Toc10056283)

[Display 8](#_Toc10056284)

[Float 8](#_Toc10056285)

[z-index 8](#_Toc10056286)

[BFC 8](#_Toc10056287)

# JS基础知识

### 闭包

闭包，官方对闭包的解释是：一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数），因而这些变量也是该表达式的一部分。闭包的特点：  
　　1. 作为一个函数变量的一个引用，当函数返回时，其处于激活状态。  
　　2. 一个闭包就是当一个函数返回时，一个没有释放资源的栈区。

3.函数内再嵌套函数

4.内部函数可以引用外层的参数和变量

5.参数和变量不会被垃圾回收机制回收

　　简单的说，**Javascript允许使用内部函数---即函数定义和函数表达式位于另一个函数的函数体内。而且，这些内部函数可以访问它们所在的外部函数中声明的所有局部变量、参数和声明的其他内部函数。当其中一个这样的内部函数在包含它们的外部函数之外被调用时，就会形成闭包**

### 2.原型

### 3.call() 和 .apply() 的作用和区别？

作用

call（）方法和apply（）方法的作用相同：改变this指向。

区别

他们的区别在于接收参数的方式不同：

call（）：第一个参数是this值没有变化，变化的是其余参数都直接传递给函数。在使用call（）方法时，传递给函数的参数必须逐个列举出来。

apply（）：传递给函数的是参数数组

如下代码做出解释：

function add(c, d){

return this.a + this.b + c + d;

}

var o = {a:1, b:3};

add.call(o, 5, 7); // 1 + 3 + 5 + 7 = 16

add.apply(o, [10, 20]); // 1 + 3 + 10 + 20 = 34

### 4.事件

事件流描述的是从页面中接收事件的顺序。 DOM 结构是树形结构，当页面中的某一个元素触发了某个一个事件，事件会从最顶层的 window 对象开始，向下传播到目标元素，途径的祖先节点都会触发对应的事件，如果当前节点的该事件绑定了事件处理函数的话，则会执行该函数当事件达到目标元素并执行绑定函数（如果有绑定的话）后，事件又会向上传播到 window 元素，途径的祖先节点都会触发对应的事件（如果绑定事件处理函数的话）

**事件流包含三个阶段：**

1. 事件捕捉阶段 2.处于目标阶段 3.事件冒泡阶段

      事件捕捉阶段：事件开始由顶层对象触发，然后逐级向下传播，直到目标的元素；

      处于目标阶段：处在绑定事件的元素上；

      事件冒泡阶段：事件由具体的元素先接收，然后逐级向上传播，直到不具体的元素；

### 事件委托：

利用冒泡事件，只指定一件事情，处理程序，就可以管理某一类型的所有事件。

只需在dom树中尽量在最高的层次上添加一个事件处理程序

### 6.了解浏览器渲染页面原理

  1. 用户输入网址（假设是个 HTML 页面，第一次访问，无缓存情况），浏览器向服务器发出HTTP请求，服务器返回 HTML 文件； （善用缓存，减少HTTP请求，减轻服务器压力）

           2. 浏览器载入 HTML 代码，发现 <head> 内有一个 <link> 引用外部 CSS 文件,则浏览器立即发送CSS文件请求，获取浏览器返回的CSS文件；  （CSS文件合并，减少HTTP请求）

           3. 浏览器继续载入 HTML 中 <body> 部分的代码，并且 CSS 文件已经拿到手了，可以开始渲染页面了；                              （CSS文件需要放置最上面，避免网页重新渲染）

           4. 浏览器在代码中发现一个 <img> 标签引用了一张图片，向服务器发出请求。此时浏览器不会等到图片下载完，而是继续渲染后面的代码；    （图片文件合并，减少HTTP请求）

           5. 服务器返回图片文件，由于图片占用了一定面积，影响了后面段落的排布，因此浏览器需要回过头来重新渲染这部分代码；   （最好图片都设置尺寸，避免重新渲染）

           6. 浏览器发现了一个包含一行 JavaScript 代码的 <script> 标签，会立即运行该js代码；              （script最好放置页面最下面）

           7. js脚本执行了语句，它令浏览器隐藏掉代码中的某个 <div>,突然就少了一个元素，浏览器不得不重新渲染这部分代码；   （页面初始化样式不要使用js控制）

           8. 终于等到了 </html> 的到来，浏览器泪流满面……

           9. 等等，还没完，用户点了一下界面中的“换肤”按钮，JavaScript 让浏览器换了一下 <link> 标签的 CSS 路径；

          10. 浏览器召集了在座的各位 <div><span><ul><li> 们，“大伙儿收拾收拾行李，咱得重新来过……”，浏览器向服务器请求了新的CSS文件，重新渲染页面。

           浏览器每天就这么来来回回跑着，要知道不同的人写出来的 HTML 和 CSS 代码质量参差不齐，说不定哪天跑着跑着就挂掉了。

           好在这个世界还有这么一群人——页面重构工程师，平时挺不起眼，也就帮视觉设计师们切切图啊改改字，其实背地里还是干了不少实事的。

**影响页面渲染速度主要有：reflow(回流)和repaint(重绘)**

**reflow(回流)**

           说到页面为什么会慢？那是因为浏览器要花时间、花精力去渲染，尤其是当它发现某个部分发生了点变化影响了布局，需要倒回去重新渲染， 该过程称为reflow（回流）。

           reflow 几乎是无法避免的。现在界面上流行的一些效果，比如树状目录的折叠、展开（实质上是元素的显 示与隐藏）等，都将引起浏览器的 reflow。鼠标滑过、点击……只要这些行为引起了页面上某些元素的占位面积、定位方式、边距等属性的变化，都会引起它内部、周围甚至整个页面的重新渲 染。通常我们都无法预估浏览器到底会 reflow 哪一部分的代码，它们都彼此相互影响着。

**repaint(重绘)**

               如果只是改变某个元素的背景色、文 字颜色、边框颜色等等不影响它周围或内部布局的属性，将只会引起浏览器 repaint（重绘）。

            repaint 的速度明显快于 reflow（在IE下需要换一下说法，reflow 要比 repaint 更缓慢）。

              尽量避免reflow(回流)

               reflow(回流)是导致DOM脚本执行低效的关键因素之一。页面上任何一个结点触发reflow，都会导致它的子结点及祖先结点重新渲染。

            在哪些情况下会导致reflow发生：

* 1. 改变窗囗大小
  2. 改变文字大小
  3. 添加/删除样式表
  4. 内容的改变，如用户在输入框中敲字
  5. 激活伪类，如:hover (IE里是一个兄弟结点的伪类被激活)
  6. 操作class属性
  7. 脚本操作DOM
  8. 计算offsetWidth和offsetHeight
  9. 设置style属性

            reflow是不可避免的，只能将reflow对性能的影响减到最小。

* 1. 尽可能限制reflow的影响范围。需要改变元素的样式，不要通过父级元素影响子元素。最好直接加在子元素上。
  2. 通过设置style属性改变结点样式的话，每设置一次都会导致一次reflow。所以最好通过设置class的方式。

* 实现元素的动画，它的position属性应当设为fixed或absolute，这样不会影响其它元素的布局。
* 权衡速度的平滑。比如实现一个动画，以1个像素为单位移动这样最平滑，但reflow就会过于频繁，CPU很快就会被完全占用。如果以3个像素为单位移动就会好很多。
* 不要用tables布局的另一个原因就是tables中某个元素一旦触发reflow就会导致table里所有的其它元素reflow。在适合用table的场合，可以设置table-layout为auto或fixed，这样可以让table一行一行的渲染，这种做法也是为了限制reflow的影响范围。
* 很多情况下都会触发reflow，如果css里有expression，每次都会重新计算一遍。
* 减少不必要的 DOM 层级（DOM depth）。改变 DOM 树中的一级会导致所有层级的改变，上至根部，下至被改变节点的子节点。这导致大量时间耗费在执行 reflow 上面。
* 避免不必要的复杂的 CSS 选择器，尤其是后代选择器（descendant selectors），因为为了匹配选择器将耗费更多的 CPU。

### 7.浏览器从输入URL到渲染出页面发生了什么

## （1）假设是简单的http请求（GET），IPV4，无代理。

1. 浏览器先查看浏览器缓存-系统缓存-路由器缓存，若缓存中有，请略过中间步骤，直接跳到第9步~若没有，则按照下面的步骤进行操作。
2. 浏览器从**url中解析出服务器的主机名，并将主机名转换成服务器的IP地址。**PS：[DNS查找域名的过程](http://www.cnblogs.com/xsilence/p/6035559.html)
3. 浏览器从**url中解析出端口号，默认80**
4. 浏览器建立一条与服务器的tcp连接（建立过程：三次握手）。  
   PS：[一个完整的TCP连接](http://www.cnblogs.com/xsilence/p/6034361.html)
5. 浏览器通过**tcp连接向服务器发送http请求，请求数据包**。
6. 服务器处理**HTTP请求，返回响应。**
7. 浏览器检查HTTP响应是否为一个重定向（3XX结果状态码）、一个验证请求（401）、错误（4XX、5XX）等等，这些都需要根据具体情况分类处理。PS：[浏览器对于常见HTTP状态码的反应](http://www.cnblogs.com/xsilence/p/6035917.html)
8. 浏览器接收HTTP响应并且可能关掉TCP连接，或者是重新建立连接使用新情求，获得新响应。
9. 浏览器解码响应，如果响应可以缓存，则存入缓存。
10. 浏览器显示HTML页面。
11. 浏览器发送请求获取嵌入在HTML中的资源（html，css，javascript，图片，音乐······），对于未知类型，会弹出对话框。
12. 浏览器发送异步请求。
13. 页面全部渲染结束。

### 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

### .DOM操作——怎样添加、移除、移动、复制、创建和查找节点?

(1)创建新节点

createDocumentFragment() //创建一个DOM片段

createElement() //创建一个具体的元素

createTextNode() //创建一个文本节点

(2)添加、移除、替换、插入

appendChild()

removeChild()

replaceChild()

insertBefore() //在已有的子节点前插入一个新的子节点

(3)查找

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)

getElementById() //通过元素Id，唯一性

### 数组对象有哪些原生方法，

# 赋值方法 (Mutator methods):

pop 和 push

shift 和 unshift

splice

reverse

sort

# 访问方法(Accessor methods) : 这些方法只是返回相应的结果，而不会修改数组本身 ;

concat

join

slice

toString

indexOf 和 lastIndexOf

# 迭代方法（Iteration methods）:

###### forEach

###### map

###### filter

###### every 和 some

###### reduce 和 reduceRight

# CSS

### **Display**

### Float

### z-index

### BFC

a. 前端代码模块化与构建，gulp、webpack等

b. NodeJS

c. MVVM

d. React、Vue、backbone、angular随便说说

一定的工程能力

a. 了解门后端语言，最起码知道后端同学在说什么

b. 了解些web 安全知识

c. 性能优化

d. 前端代码工作及发布体系

1.怎么去设计一个组件封装

组件封装的目的是为了重用，提高开发效率和代码质量低耦合，单一职责，可复用性，可维护性前端组件化设计思路

2.js 异步加载的方式

渲染引擎遇到 script 标签会停下来，等到执行完脚本，继续向下渲染defer 是“渲染完再执行”，async 是“下载完就执行”，defer 如果有多个脚本，会按照在页面中出现的顺序加载，多个async 脚本不能保证加载顺序加载 es6模块的时候设置 type=module，异步加载不会造成阻塞浏览器，页面渲染完再执行，可以同时加上async属性，异步执行脚本（利用顶层的this等于undefined这个语法点，可以侦测当前代码是否在 ES6 模块之中）

3.css 动画和 js 动画的差异

代码复杂度，js 动画代码相对复杂一些动画运行时，对动画的控制程度上，js 能够让动画，暂停，取消，终止，css动画不能添加事件动画性能看，js 动画多了一个js 解析的过程，性能不如 css 动画好

4.XSS 与 CSRF 两种跨站攻击

xss 跨站脚本攻击，主要是前端层面的，用户在输入层面插入攻击脚本，改变页面的显示，或则窃取网站 cookie，预防方法：不相信用户的所有操作，对用户输入进行一个转义，不允许 js 对 cookie 的读写csrf 跨站请求伪造，以你的名义，发送恶意请求，通过 cookie 加参数等形式过滤我们没法彻底杜绝攻击，只能提高攻击门槛

5.事件委托，目的，功能，写法

把一个或者一组元素的事件委托到它的父层或者更外层元素上优点，减少内存消耗，动态绑定事件target 是触发事件的最具体的元素，currenttarget是绑定事件的元素(在函数中一般等于this)JavaScript 事件委托详解

6.线程，进程

线程是最小的执行单元，进程是最小的资源管理单元一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程

7.负载均衡

当系统面临大量用户访问，负载过高的时候，通常会使用增加服务器数量来进行横向扩展，使用集群和负载均衡提高整个系统的处理能力服务器集群负载均衡原理？

8.什么是CDN缓存

CDN 是一种部署策略，根据不同的地区部署类似nginx 这种服务服务，会缓存静态资源。前端在项目优化的时候，习惯在讲台资源上加上一个 hash 值，每次更新的时候去改变这个 hash，hash 值变化的时候，服务会去重新取资源(CDN)是一个经策略性部署的整体系统，包括分布式存储、负载均衡、网络请求的重定向和内容管理4个要件CDN\_百度百科

9.闭包的写法，闭包的作用，闭包的缺点

使用闭包的目的——隐藏变量，间接访问一个变量,在定义函数的词法作用域外，调用函数闭包的内存泄露，是IE的一个 bug，闭包使用完成之后，收回不了闭包的引用，导致内存泄露「每日一题」JS 中的闭包是什么？闭包造成内存泄露的实验

10.跨域问题，谁限制的跨域，怎么解决

浏览器的同源策略导致了跨域用于隔离潜在恶意文件的重要安全机制[jsonp ，允许 script 加载第三方资源]https://segmentfault.com/a/11…nginx 反向代理（nginx 服务内部配置 Access-Control-Allow-Origin \*）cors 前后端协作设置请求头部，Access-Control-Allow-Origin 等头部信息iframe 嵌套通讯，postmessage

11.javascript 中常见的内存泄露陷阱

内存泄露会导致一系列问题，比如：运行缓慢，崩溃，高延迟内存泄露是指你用不到（访问不到）的变量，依然占居着内存空间，不能被再次利用起来意外的全局变量，这些都是不会被回收的变量（除非设置 null 或者被重新赋值），特别是那些用来临时存储大量信息的变量周期函数一直在运行，处理函数并不会被回收，jq 在移除节点前都会，将事件监听移除js 代码中有对 DOM 节点的引用，dom 节点被移除的时候，引用还维持JavaScript 中 4 种常见的内存泄露陷阱

12.babel把ES6转成ES5或者ES3之类的原理是什么

它就是个编译器，输入语言是ES6+，编译目标语言是ES5babel 官方工作原理解析：将代码字符串解析成抽象语法树变换：对抽象语法树进行变换操作再建：根据变换后的抽象语法树再生成代码字符串

13.Promise 模拟终止

当新对象保持“pending”状态时，原Promise链将会中止执行。return new Promise(()=>{}); // 返回“pending”状态的Promise对象从如何停掉 Promise 链说起(promise内存泄漏问题)

14.promise 放在try catch里面有什么结果

Promise 对象的错误具有冒泡性质，会一直向后传递，直到被捕获为止，也即是说，错误总会被下一个catch语句捕获当Promise链中抛出一个错误时，错误信息沿着链路向后传递，直至被捕获

15.网站性能优化

http 请求方面，减少请求数量，请求体积，对应的做法是，对项目资源进行压缩，控制项目资源的 dns 解析在2到4个域名，提取公告的样式，公共的组件，雪碧图，缓存资源，压缩资源，提取公共资源压缩，提取 css ，js 公共方法不要缩放图片，使用雪碧图，使用字体图表（阿里矢量图库）使用 CDN，抛开无用的 cookie减少重绘重排，CSS属性读写分离，最好不要用js 修改样式，dom 离线更新，渲染前指定图片的大小js 代码层面的优化，减少对字符串的计算，合理使用闭包，首屏的js 资源加载放在最底部

16.js 自定义事件实现

原生提供了3个方法实现自定义事件createEvent，设置事件类型，是 html 事件还是 鼠标事件initEvent 初始化事件，事件名称，是否允许冒泡，是否阻止自定义事件dispatchEvent 触发事件

18.get与post 通讯的区别

Get 请求能缓存，Post 不能Post 相对 Get 安全一点点，因为Get 请求都包含在 URL 里，且会被浏览器保存历史纪录，Post 不会，但是在抓包的情况下都是一样的。Post 可以通过 request body来传输比 Get 更多的数据，Get 没有这个技术URL有长度限制，会影响 Get 请求，但是这个长度限制是浏览器规定的，不是 RFC 规定的Post 支持更多的编码类型且不对数据类型限制

19.有没有去研究webpack的一些原理和机制，怎么实现的

解析webpack配置参数，合并从shell传入和webpack.config.js文件里配置的参数，生产最后的配置结果。注册所有配置的插件，好让插件监听webpack构建生命周期的事件节点，以做出对应的反应。从配置的entry入口文件开始解析文件构建AST语法树，找出每个文件所依赖的文件，递归下去。在解析文件递归的过程中根据文件类型和loader配置找出合适的loader用来对文件进行转换。递归完后得到每个文件的最终结果，根据entry配置生成代码块chunk。输出所有chunk到文件系统。

**1 DOCTYPE有什么作用？标准模式与混杂模式如何区分？它们有何意义?**

告诉浏览器使用哪个版本的HTML规范来渲染文档。DOCTYPE不存在或形式不正确会导致HTML文档以混杂模式呈现。  
标准模式（Standards mode）以浏览器支持的最高标准运行；混杂模式（Quirks mode）中页面是一种比较宽松的向后兼容的方式显示。

**2 HTML5为什么只需要写 <!DOCTYPE HTML>？**

HTML5不基于SGML（Standard Generalized Markup Language 标准通用标记语言），因此不需要对DTD（DTD 文档类型定义）进行引用，但是需要DOCTYPE来规范浏览器行为。

HTML4.01基于SGML，所以需要引用DTD。才能告知浏览器文档所使用的文档类型，如下：  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

**3 行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？**

行内元素：a span img input select   
块级元素：div ul ol li dl dt dd h1 p  
空元素：<br> <hr> <link> <meta>

**4 页面导入样式时，使用link和@import有什么区别？**

相同的地方，都是外部引用CSS方式，区别：

1. link是xhtml标签，除了加载css外，还可以定义RSS等其他事务；@import属于CSS范畴，只能加载CSS
2. link引用CSS时候，页面载入时同时加载；@import需要在页面完全加载以后加载，而且@import被引用的CSS会等到引用它的CSS文件被加载完才加载
3. link是xhtml标签，无兼容问题；@import是在css2.1提出来的，低版本的浏览器不支持
4. link支持使用javascript控制去改变样式，而@import不支持
5. link方式的样式的权重高于@import的权重
6. import在html使用时候需要<style type="text/css">标签

**5 无样式内容闪烁（FOUC）Flash of Unstyle Content**

@import导入CSS文件会等到文档加载完后再加载CSS样式表。因此，在页面DOM加载完成到CSS导入完成之间会有一段时间页面上的内容是没有样式的。

解决方法：使用link标签加载CSS样式文件。因为link是顺序加载的，这样页面会等到CSS下载完之后再下载HTML文件，这样先布局好，就不会出现FOUC问题。

**6 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 Engine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。  
JS引擎：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。

**7 常见的浏览器内核有哪些？**

1. Trident( MSHTML )：IE MaxThon TT The World 360 搜狗浏览器
2. Geckos：Netscape6及以上版本 FireFox Mozilla Suite/SeaMonkey
3. Presto：Opera7及以上(Opera内核原为：Presto，现为：Blink)
4. Webkit：Safari Chrome

**8 HTML5有哪些新特性,移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分HTML和HTML5**

新增加了图像、位置、存储、多任务等功能。  
新增元素：

1. canvas
2. 用于媒介回放的video和audio元素
3. 本地离线存储。localStorage长期存储数据，浏览器关闭后数据不丢失;sessionStorage的数据在浏览器关闭后自动删除
4. 语意化更好的内容元素，比如 article footer header nav section
5. 位置API：Geolocation
6. 表单控件，calendar date time email url search
7. 新的技术：web worker(web worker是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 在后台运行) web socket
8. 拖放API：drag、drop

移除的元素：

1. 纯表现的元素：basefont big center font s strike tt u
2. 性能较差元素：frame frameset noframes

区分：

1. DOCTYPE声明的方式是区分重要因素
2. 根据新增加的结构、功能来区分

**9 简述一下你对HTML语义化的理解？**

1. 去掉或丢失样式的时候能够让页面呈现出清晰的结构。
2. 有利于SEO和搜索引擎建立良好沟通，有助于爬虫抓取更多的信息，爬虫依赖于标签来确定上下文和各个关键字的权重。
3. 方便其它设备解析。
4. 便于团队开发和维护，语义化根据可读性。

**10 HTML5的文件离线储存怎么使用，工作原理是什么？**

在线情况下，浏览器发现HTML头部有manifest属性，它会请求manifest文件，如果是第一次访问，那么浏览器就会根据manifest文件的内容下载相应的资源，并进行离线存储。如果已经访问过并且资源已经离线存储了，那么浏览器就会使用离线的资源加载页面。然后浏览器会对比新的manifest文件与旧的manifest文件，如果文件没有发生改变，就不会做任何操作，如果文件改变了，那么就会重新下载文件中的资源，并且进行离线存储。例如，

在页面头部加入manifest属性

<**html** manifest='cache.manifest'>

在cache.manifest文件中编写离线存储的资源

1. CACHE MANIFEST
2. **#v0**.11
3. CACHE:
4. js/app.js
5. css/style.css
6. NETWORK:
7. Resourse/logo.png
8. FALLBACK:
9. *//offline.html*

**11 cookies，sessionStorage和localStorage的区别？**

共同点：都是保存在浏览器端，且是同源的。

区别：

1. cookies是为了标识用户身份而存储在用户本地终端上的数据，始终在同源http请求中携带，即cookies在浏览器和服务器间来回传递，而sessionstorage和localstorage不会自动把数据发给服务器，仅在本地保存。
2. 存储大小的限制不同。cookie保存的数据很小，不能超过4k，而sessionstorage和localstorage保存的数据大，可达到5M。
3. 数据的有效期不同。cookie在设置的cookie过期时间之前一直有效，即使窗口或者浏览器关闭。sessionstorage仅在浏览器窗口关闭之前有效。localstorage始终有效，窗口和浏览器关闭也一直保存，用作长久数据保存。
4. 作用域不同。cookie在所有的同源窗口都是共享；sessionstorage不在不同的浏览器共享，即使同一页面；localstorage在所有同源窗口都是共享

**12 iframe框架有那些优缺点？**

优点：

1. iframe能够原封不动的把嵌入的网页展现出来。
2. 如果有多个网页引用iframe，那么你只需要修改iframe的内容，就可以实现调用的每一个页面内容的更改，方便快捷。
3. 网页如果为了统一风格，头部和版本都是一样的，就可以写成一个页面，用iframe来嵌套，可以增加代码的可重用。
4. 如果遇到加载缓慢的第三方内容如图标和广告，这些问题可以由iframe来解决。

缺点：

1. 搜索引擎的爬虫程序无法解读这种页面
2. 框架结构中出现各种滚动条
3. 使用框架结构时，保证设置正确的导航链接。
4. iframe页面会增加服务器的http请求

**13 label的作用是什么? 是怎么用的?**

label标签用来定义表单控件间的关系,当用户选择该标签时，浏览器会自动将焦点转到和标签相关的表单控件上。label 中有两个属性是非常有用的, FOR和ACCESSKEY。   
FOR属性功能：表示label标签要绑定的HTML元素，你点击这个标签的时候，所绑定的元素将获取焦点。例如，

<**Label** **FOR**="InputBox">姓名</**Label**><**input** ID="InputBox" **type**="text">

ACCESSKEY属性功能：表示访问label标签所绑定的元素的热键，当您按下热键，所绑定的元素将获取焦点。例如，

<**Label** **FOR**="InputBox" ACCESSKEY＝"N">姓名</**Label**><**input** ID="InputBox" **type**="text">

**14 HTML5的form如何关闭自动完成功能？**

HTML的输入框可以拥有自动完成的功能，当你往输入框输入内容的时候，浏览器会从你以前的同名输入框的历史记录中查找出类似的内容并列在输入框下面，这样就不用全部输入进去了，直接选择列表中的项目就可以了。但有时候我们希望关闭输入框的自动完成功能，例如当用户输入内容的时候，我们希望使用AJAX技术从数据库搜索并列举而不是在用户的历史记录中搜索。

方法：

1. 在IE的internet选项菜单中里的自动完成里面设置
2. 设置form输入框的autocomplete为on或者off来来开启输入框的自动完成功能

**15 如何实现浏览器内多个标签页之间的通信?**

1. WebSocket SharedWorker
2. 也可以调用 localstorge、cookies 等本地存储方式。 localstorge 在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：Safari 在无痕模式下设置 localstorge 值时会抛出QuotaExceededError 的异常

**16 webSocket如何兼容低浏览器？**

1. Adobe Flash Socket ActiveX HTMLFile (IE) 基于 multipart 编码发送 XHR 基于长轮询的 XHR
2. 引用WebSocket.js这个文件来兼容低版本浏览器。

**16 页面可见性（Page Visibility）API 可以有哪些用途？**

1. 通过visibility state的值得检测页面当前是否可见，以及打开网页的时间。
2. 在页面被切换到其他后台进程时，自动暂停音乐或视频的播放。

**17 如何在页面上实现一个圆形的可点击区域？**

1. map+area或者svg
2. border-radius
3. 纯js实现，一个点不在圆上的算法

**18 实现不使用 border 画出1px高的线，在不同浏览器的Quirks mode和CSS Compat模式下都能保持同一效果**

<**div** style="height:1px;overflow:hidden;background:red"></**div**>

**19 网页验证码是干嘛的，是为了解决什么安全问题？**

1. 区分用户是计算机还是人的程序;
2. 可以防止恶意破解密码、刷票、论坛灌水；

**20 title与h1的区别、b与strong的区别、i与em的区别？**

title属性没有明确意义，只表示标题；h1表示层次明确的标题，对页面信息的抓取也有很大的影响  
strong标明重点内容，语气加强含义；b是无意义的视觉表示  
em表示强调文本；i是斜体，是无意义的视觉表示  
视觉样式标签：b i u s  
语义样式标签：strong em ins del code

**21 元素的alt和title有什么异同？**

在alt和title同时设置的时候，alt作为图片的替代文字出现，title是图片的解释文字。