一：html

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？

\* HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。

绘画 canvas;

用于媒介回放的 video 和 audio 元素;

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;

语意化更好的内容元素，比如 article、footer、header、nav、section;

表单控件，calendar、date、time、email、url、search;

新的技术webworker, websocket, Geolocation;

移除的元素：

纯表现的元素：basefont，big，center，font, s，strike，tt，u;

对可用性产生负面影响的元素：frame，frameset，noframes；

\* 支持HTML5新标签：

IE8/IE7/IE6支持通过document.createElement方法产生的标签，

可以利用这一特性让这些浏览器支持HTML5新标签，

浏览器支持新标签后，还需要添加标签默认的样式。

当然也可以直接使用成熟的框架、比如html5shim;

<!--[iflt IE 9]>

<script>src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>

<![endif]-->

\* 如何区分HTML5： DOCTYPE声明\新增的结构元素\功能元素

js 原生代码找对象的方法

1. id ： document.getElementById('id')

2. 标签 : document.getElementsByTagName('标签') //获得的是一个标签数组

3. Name :document.getElementsByName（'name名'） // 该方法与 getElementById() 方法相似，但是它查询元素的 name 属性，而不是 id 属性

4. className:document.getElementsByClassName('class') //获得一个class的数组

5.document.images : 获得img对象//获得数组

6.document.links : 获得所有的链接//获得数组

7.document.forms:获得表单对象// 获得数组

8.document.body: body 标签对象

9.document.documentElement :html标签对象

10. event： 事件对象

11. this ：当前对象

添加HTML元素

创建元素：Document.createElement()

找到父级元素 ：可以通过Id、name、标签名、class、匹配指定的css选择器

在指定位置插入元素：element.appendChild(span)在末尾中添加元素、element.insertBefore(span,container.childNodes[0]) 插入到最左边

删除已有的 HTML 元素

var parent=document.getElementById("div1");

var child=document.getElementById("p1");

parent.removeChild(child);

rowIndex 属性返回某一行在表格的行集合中的位置（row index）。

document.getElementById("tr1").rowIndex

开头说下innerText和outerText只在chrome浏览器中有效

innerHTML 输出当前标签的文本内容，如果标签内有子标签，会连子标签本身和子标签内的文本内容一起输出

innerText 只输出当前标签内的文本内容，如果标签内有子标签，那么也只输出子标签内的文本内容

outerHTML 输出当前标签的本身和标签内的文本内容，如果有子标签，那么子标签本身和标签内的文本内容也将一起输出

outerText 只输出当前标签内的文本内容，如果标签内有子标签，那么也只输出子标签内的文本内容

二：js

let a = new Set([1, 2, 3]);

let b = new Set([3, 5, 2]);

// 并集

let unionSet = new Set([...a, ...b]);

//[1,2,3,5]

// 交集

let intersectionSet = new Set([...a].filter(x => b.has(x)));

// [2,3]

// ab差集

let differenceABSet = new Set([...a].filter(x => !b.has(x)));

柯里化函数

function add(n) {

var fn = function(m) {

return add(n + m);

};

fn.valueOf = function() {

return n;

};

fn.toString = function() {

return '' + n;

};

return fn;

}

add(1) // 结果 1

add(1)(2) // 结果 3

add(1)(2)(3) // 结果 6

add(1)(2)(3)(4) // 结果 10

‘abc1234def5678gh’变成abc,1,2,3,4,def,5,6,7,8,gh

var s="abc1234def5678gh";

s.replace(/\d{1}/g,function(s1){return ' ' + s1 + ' '}).trim().replace(/\s+/g,'，');

//"abc,1,2,3,4,def,5,6,7,8,gh"

var s="abcA1234def5678gh";

s.replace(/[a-zA-Z]{1}/g,function(s1){return ' ' + s1 + ' '}).trim().replace(/\s+/g,',');

//"a,b,c,A,1234,d,e,f,5678,g,h"

正则表达式后行断言 • 探索 ES2018 和 ES2019

var s="12345678";

s.replace(/(?=(\d{3})+(?!\d))/g,",");

//"12,345,678"

https://juejin.im/entry/5b4808e9f265da0f9c678ff4

输入框连续搜索请求两次，第二次结果先到，怎么取消第一次的搜索结果渲染？

方法一：后台将搜索关键词一并返给前端，前端通过判断是否为当前搜索关键词决定是否渲染。

方法二：

var CancelToken = axios.CancelToken;

var source = CancelToken.source();

axios.get('/user/12345', {

cancelToken: source.token

}).then(()=>{

// 你的逻辑

}).catch(function(thrown) {

// 如果调用了cancel方法，那么这里的res就是cancel传入的信息

// 你的逻辑

});

source.cancel('这里你可以输出一些信息，可以在catch中拿到');

方法三：

var native = new XMLHttpRequest();

native.open("GET","https://api.github.com/");

native.send();

native.onreadystatechange=function(){

if(native.readyState==4&&native.status==200){

console.log(native.response);

}else{

console.log(native.status);

}

}

native.abort();

算法排序

1、冒泡排序

function bubbleSort(arr) {

　　var len = arr.length;

　　for (var i = 0; i < len; i++) {

　　　　for (var j = 0; j < len - 1 - i; j++) {

　　　　　　if (arr[j] > arr[j+1]) { //相邻元素两两对比

　　　　　　　　[arr[j], arr[j+1]] = [arr[j+1], arr[j]]

　　　　　　}

　　　　}

　　}

　　return arr;

}

var arr=[3,44,38,5,47,15,36,26,27,2,46,4,19,50,48];

console.log(bubbleSort(arr));

2、插入排序

function insertionSort(array) {

　　for (var i = 1; i < array.length; i++) {

　　　　var key = array[i];

　　　　var j = i - 1;

　　　　while ( array[j] > key) {

　　　　　　array[j + 1] = array[j];

　　　　　 j--;

　　　　}

　　　　array[j + 1] = key;

　　}

　　return array;

}

var arr=[3,44,38,5,47,15,36,26,27,2,46,4,19,50,48];

console.log(insertionSort(arr));

3、快速排序

function quickSort(arr) {

　　if (arr.length <= 1) { return arr; }

　　var pivotIndex = Math.floor(arr.length / 2);

　　var pivot = arr.splice(pivotIndex, 1)[0];

　　var left = [];

　　var right = [];

　　for (var i = 0; i < arr.length; i++){

　　　　if (arr[i] < pivot) {

　　　　　　left.push(arr[i]);

　　　　} else {

　　　　　　right.push(arr[i]);

　　　　}

　　}

　　return quickSort(left).concat([pivot], quickSort(right));

};

var arr=[3,44,38,5,47,15,36,26,27,2,46,4,19,50,48];

console.log(quickSort2(arr));

三：css

css3新特性

Transition：

transition-property: 规定应用过渡的CSS属性的名称。

transition-duration: 规定完成过渡效果需要多长时间。

transition-delay: 规定过渡效果何时开始，默认是0。

transition-timing-function: 规定过渡效果的时间曲线，默认是”ease”，还有linear、ease-in、ease-out、ease-in-out和cubic-bezier等过渡类型。

transition:width 1s linear 2s; 简写属性，用于在一个属性中设置四个过渡属性。

Transform：

none: 定义不进行转换。

matrix(n,n,n,n,n,n): 定义2D转换，使用六个值的矩阵。

matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n): 定义3D转换，使用16个值的4x4矩阵。

translate(x,y): 定义2D位移转换。

translate3d(x,y,z): 定义3D位移转换。

translateX(x): 定义位移转换，只是用X轴的值。

translateY(y): 定义位移转换，只是用Y轴的值。

translateZ(z): 定义3D位移转换，只是用Z轴的值。

scale(x,y): 定义2D缩放转换。

scale3d(x,y,z): 定义3D缩放转换。

scaleX(x): 通过设置X轴的值来定义缩放转换。

scaleY(y): 通过设置Y轴的值来定义缩放转换。

scaleZ(z): 通过设置Z轴的值来定义3D缩放转换。

rotate(angle): 定义2D旋转，在参数中规定角度。

rotate3d(x,y,z,angle): 定义3D旋转。

rotateX(angle): 定义沿着X轴的3D旋转。

rotateY(angle): 定义沿着Y轴的3D旋转。

rotateZ(angle): 定义沿着Z轴的3D旋转。

skew(x-angle,y-angle): 定义沿着X和Y轴的2D倾斜转换。

skewX(angle): 定义沿着X轴的2D倾斜转换。

skewY(angle): 定义沿着Y轴的2D倾斜转换。

perspective(n): 为3D转换元素定义透视视图。

Animation：动画

animation-name 规定需要绑定到选择器的 keyframe 名称。。

animation-duration 规定完成动画所花费的时间，以秒或毫秒计。

animation-timing-function 规定动画的速度曲线。

animation-delay 规定在动画开始之前的延迟。

animation-iteration-count 规定动画应该播放的次数。

animation-direction 规定是否应该轮流反向播放动画。

CSS3新增了三个边框属性，分别是border-radius、box-shadow和border-image。border-radius可以创建圆角边框，box-shadow可以为元素添加阴影，border-image可以使用图片来绘制边框。IE9+支持border-radius和box-shadow属性。Firefox、Chrome以及Safari支持所有新的边框属性。

word-wrap:break-word;属性允许您允许文本强制文本进行换行

text-overflow它与word-wrap是协同工作的，word-wrap设置或检索当当前行超过指定容器的边界时是否断开转行，而 text-overflow则设置或检索当当前行超过指定容器的边界时如何显示

text-shadow: 5px 5px 5px #FF0000;可向文本应用阴影。能够规定水平阴影、垂直阴影、模糊距离，以及阴影的颜色。

text-decoration：

CSS3里面开始支持对文字的更深层次的渲染，具体有三个属性可供设置：

text-fill-color: 设置文字内部填充颜色

text-stroke-color: 设置文字边界填充颜色

text-stroke-width: 设置文字边界宽度

linear-gradient(线性渐变)和radial-gradient(径向渐变)

@font-face特性

多列布局

column-count: 规定元素应该被分隔的列数。

column-gap: 规定列之间的间隔。

column-rule: 设置列之间的宽度、样式和颜色规则

resize 属性规定是否可由用户调整元素尺寸。如果希望此属性生效，需要设置元素的 overflow 属性，值可以是 auto、hidden 或 scroll

box-sizing属性可设置的值有content-box、border-box和inherit。

outline-offset属性对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。

四：浏览器

js文件加载阻塞以及defer和async的区别

<script src="script.js"></script>

没有 defer 或 async，浏览器会立即加载并执行指定的脚本，“立即”指的是在渲染该 script 标签之下的文档元素之前，也就是说不等待后续载入的文档元素，读到就加载并执行

<script async src="script.js"></script>

有 async，加载和渲染后续文档元素的过程将和 script.js 的加载与执行并行进行（异步）。

<script defer src="myscript.js"></script>

有 defer，加载后续文档元素的过程将和 script.js 的加载并行进行（异步），但是 script.js 的执行要在所有元素解析完成之后，DOMContentLoaded 事件触发之前完成。

css加载是否会阻塞页面渲染

1、css加载不会阻塞DOM树的解析

2、css加载会阻塞DOM树的渲染

3、css加载会阻塞后面js语句的执行

为了避免让用户看到长时间的白屏时间，我们应该尽可能的提高css加载速度，比如可以使用以下几种方法:

使用CDN(因为CDN会根据你的网络状况，替你挑选最近的一个具有缓存内容的节点为你提供资源，因此可以减少加载时间)

对css进行压缩(可以用很多打包工具，比如webpack,gulp等，也可以通过开启gzip压缩)

合理的使用缓存(设置cache-control,expires,以及E-tag都是不错的，不过要注意一个问题，就是文件更新后，你要避免缓存而带来的影响。其中一个解决防范是在文件名字后面加一个版本号)

减少http请求数，将多个css文件合并，或者是干脆直接写成内联样式(内联样式的一个缺点就是不能缓存)

堆栈

https://juejin.im/post/5ae09017f265da0b8f623aa3

权限控制

https://juejin.im/post/5c0a96e66fb9a049ec6ae305

安全

https://juejin.im/entry/584b880a8e450a006c590576

缓存

https://juejin.im/post/5b0ea4f1518825155d66a97b

https://juejin.im/post/5ae081aaf265da0b767d263a

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；

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

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(layoutengineer或RenderingEngine)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。

移动端兼容问题

1、input 有placeholder情况下不要设置行高，否则文本会偏上

2、input为fixed定位在ios下的bug问题，点击焦点input，input跳到中间位置

软键盘唤起后，页面的 fixed 元素将失效（即无法浮动，也可以理解为变成了 absolute 定位），所以当页面超过一屏且滚动时，失效的 fixed 元素就会跟随滚动了

既然在 iOS 下由于软键盘唤出后，页面 fixed 元素会失效，导致跟随页面一起滚动，那么假如——页面不会过长出现滚动，那么即便 fixed 元素失效，也无法跟随页面滚动，也就不会出现上面的问题了。

那么按照这个思路，如果使 fixed 元素的父级不出现滚动，而将原 body 滚动的区域域移到 main 内部

3、一些情况下对非可点击元素如(label,span)监听click事件，ios下不会触发，css增加cursor:pointer就搞定了

4、new Date('2019-01-30')在IOS上无效，使用moment转一次

5、关于 iOS 系统中，中文输入法输入英文时，字母之间可能会出现一个六分之一空格；

this.value = this.value.replace(/\u2006/g,'');

6、手机上的flex布局时会有兼容性问题，只用新版本的会出现安卓手机不识别的现象

.box{

display: -webkit-box;

/\* 老版本语法: Safari, iOS, Android browser, older WebKit browsers. \*/

display: -moz-box; /\* 老版本语法: Firefox (buggy) \*/

display: -ms-flexbox; /\* 混合版本语法: IE 10 \*/

display: -webkit-flex; /\* 新版本语法: Chrome 21+ \*/

display: flex; /\* 新版本语法: Opera 12.1, Firefox 22+ \*/

}

事件流

JavaScript中的事件流分为捕获阶段、目标阶段、冒泡阶段，我们可以JS控制需要操作捕获还是冒泡阶段的事件。

根据事件流的原理，我们可以优化事件处理方法，比如进行事件委托。

“DOM2级事件”定义了两个方法，用于处理指定和删除事件处理程序的操作：addEventListener( )和removeEventListener( )。

所有的DOM节点中都包含这两个方法，并且它们接收3个参数：要处理的事件，作为事件处理程序的函数和一个布尔值。布尔值如果是true，表示在捕获阶段调用事件处理程序，如果是false，表示在冒泡阶段调用事件处理程序。

WEB开发中常见的漏洞  
1、SQL注入漏洞   
SQL注入攻击（SQL Injection），简称注入攻击、SQL注入，被广泛用于非法获取网站控制权，是发生在应用程序的数据库层上的安全漏洞。在设计程序，忽略了对输入字符串中夹带的SQL指令的检查，被数据库误认为是正常的SQL指令而运行，从而使数据库受到攻击，可能导致数据被窃取、更改、删除，以及进一步导致网站被嵌入恶意代码、被植入后门程序等危害。   
通常情况下，SQL注入的位置包括：   
（1）表单提交，主要是POST请求，也包括GET请求；   
（2）URL参数提交，主要为GET请求参数；   
（3）Cookie参数提交；   
（4）HTTP请求头部的一些可修改的值，比如Referer、User\_Agent等；   
（5）一些边缘的输入点，比如.mp3文件的一些文件信息等。   
常见的防范方法   
（1）所有的查询语句都使用数据库提供的参数化查询接口，参数化的语句使用参数而不是将用户输入变量嵌入到SQL语句中。当前几乎所有的数据库系统都提供了参数化SQL语句执行接口，使用此接口可以非常有效的防止SQL注入攻击。   
（2）对进入数据库的特殊字符（’”<>&\*;等）进行转义处理，或编码转换。   
（3）确认每种数据的类型，比如数字型的数据就必须是数字，数据库中的存储字段必须对应为int型。   
（4）数据长度应该严格规定，能在一定程度上防止比较长的SQL注入语句无法正确执行。   
（5）网站每个数据层的编码统一，建议全部使用UTF-8编码，上下层编码不一致有可能导致一些过滤模型被绕过。   
（6）严格限制网站用户的数据库的操作权限，给此用户提供仅仅能够满足其工作的权限，从而最大限度的减少注入攻击对数据库的危害。   
（7）避免网站显示SQL错误信息，比如类型错误、字段不匹配等，防止攻击者利用这些错误信息进行一些判断。   
（8）在网站发布之前建议使用一些专业的SQL注入检测工具进行检测，及时修补这些SQL注入漏洞。  
  
2、跨站脚本漏洞   
跨站脚本攻击（Cross-site scripting，通常简称为XSS）发生在客户端，可被用于进行窃取隐私、钓鱼欺骗、窃取密码、传播恶意代码等攻击。   
XSS攻击使用到的技术主要为HTML和JavaScript，也包括VBScript和ActionScript等。XSS攻击对WEB服务器虽无直接危害，但是它借助网站进行传播，使网站的使用用户受到攻击，导致网站用户帐号被窃取，从而对网站也产生了较严重的危害。   
XSS类型包括：   
（1）非持久型跨站：即反射型跨站脚本漏洞，是目前最普遍的跨站类型。跨站代码一般存在于链接中，请求这样的链接时，跨站代码经过服务端反射回来，这类跨站的代码不存储到服务端（比如数据库中）。上面章节所举的例子就是这类情况。   
（2）持久型跨站：这是危害最直接的跨站类型，跨站代码存储于服务端（比如数据库中）。常见情况是某用户在论坛发贴，如果论坛没有过滤用户输入的javascript代码数据，就会导致其他浏览此贴的用户的浏览器会执行发贴人所嵌入的Javascript代码。   
（3）DOM跨站（DOM XSS）：是一种发生在客户端DOM（Document Object Model文档对象模型）中的跨站漏洞，很大原因是因为客户端脚本处理逻辑导致的安全问题。   
常用的防止XSS技术包括：   
（1）与SQL注入防护的建议一样，假定所有输入都是可疑的，必须对所有输入中的script、iframe等字样进行严格的检查。这里的输入不仅仅是用户可以直接交互的输入接口，也包括HTTP请求中的Cookie中的变量，HTTP请求头部中的变量等。   
（2）不仅要验证数据的类型，还要验证其格式、长度、范围和内容。   
（3）不要仅仅在客户端做数据的验证与过滤，关键的过滤步骤在服务端进行。   
（4）对输出的数据也要检查，数据库里的值有可能会在一个大网站的多处都有输出，即使在输入做了编码等操作，在各处的输出点时也要进行安全检查。   
（5）在发布应用程序之前测试所有已知的威胁。  
  
3、弱口令漏洞   
弱口令(weak password) 没有严格和准确的定义，通常认为容易被别人（他们有可能对你很了解）猜测到或被破解工具破解的口令均为弱口令。设置密码通常遵循以下原则：   
（1）不使用空口令或系统缺省的口令，这些口令众所周之，为典型的弱口令。   
（2）口令长度不小于8个字符。   
（3）口令不应该为连续的某个字符（例如：AAAAAAAA）或重复某些字符的组合（例如：tzf.tzf.）。   
（4）口令应该为以下四类字符的组合，大写字母(A-Z)、小写字母(a-z)、数字(0-9)和特殊字符。每类字符至少包含一个。如果某类字符只包含一个，那么该字符不应为首字符或尾字符。   
（5）口令中不应包含本人、父母、子女和配偶的姓名和出生日期、纪念日期、登录名、E-mail地址等等与本人有关的信息，以及字典中的单词。   
（6）口令不应该为用数字或符号代替某些字母的单词。   
（7）口令应该易记且可以快速输入，防止他人从你身后很容易看到你的输入。   
（8）至少90天内更换一次口令，防止未被发现的入侵者继续使用该口令。  
  
4、HTTP报头追踪漏洞   
HTTP/1.1（RFC2616）规范定义了HTTP TRACE方法，主要是用于客户端通过向Web服务器提交TRACE请求来进行测试或获得诊断信息。当Web服务器启用TRACE时，提交的请求头会在服务器响应的内容（Body）中完整的返回，其中HTTP头很可能包括Session Token、Cookies或其它认证信息。攻击者可以利用此漏洞来欺骗合法用户并得到他们的私人信息。该漏洞往往与其它方式配合来进行有效攻击，由于HTTP TRACE请求可以通过客户浏览器脚本发起（如XMLHttpRequest），并可以通过DOM接口来访问，因此很容易被攻击者利用。   
防御HTTP报头追踪漏洞的方法通常禁用HTTP TRACE方法。  
  
5、Struts2远程命令执行漏洞   
ApacheStruts是一款建立Java web应用程序的开放源代码架构。Apache Struts存在一个输入过滤错误，如果遇到转换错误可被利用注入和执行任意Java代码。   
网站存在远程代码执行漏洞的大部分原因是由于网站采用了Apache Struts Xwork作为网站应用框架，由于该软件存在远程代码执高危漏洞，导致网站面临安全风险。CNVD处置过诸多此类漏洞，例如：“GPS车载卫星定位系统”网站存在远程命令执行漏洞(CNVD-2012-13934)；Aspcms留言本远程代码执行漏洞（CNVD-2012-11590）等。   
修复此类漏洞，只需到Apache官网升级Apache Struts到最新版本：IMG_256http://struts.apache.org  
  
6、文件上传漏洞   
文件上传漏洞通常由于网页代码中的文件上传路径变量过滤不严造成的，如果文件上传功能实现代码没有严格限制用户上传的文件后缀以及文件类型，攻击者可通过 Web 访问的目录上传任意文件，包括网站后门文件（webshell），进而远程控制网站服务器。   
因此，在开发网站及应用程序过程中，需严格限制和校验上传的文件，禁止上传恶意代码的文件。同时限制相关目录的执行权限，防范webshell攻击。  
  
7、私有IP地址泄露漏洞   
IP地址是网络用户的重要标示，是攻击者进行攻击前需要了解的。获取的方法较多，攻击者也会因不同的网络情况采取不同的方法，如：在局域网内使用Ping指令，Ping对方在网络中的名称而获得IP；在Internet上使用IP版的QQ直接显示。最有效的办法是截获并分析对方的网络数据包。攻击者可以找到并直接通过软件解析截获后的数据包的IP包头信息，再根据这些信息了解具体的IP。   
针对最有效的“数据包分析方法”而言，就可以安装能够自动去掉发送数据包包头IP信息的一些软件。不过使用这些软件有些缺点，譬如：耗费资源严重，降低计算机性能；访问一些论坛或者网站时会受影响；不适合网吧用户使用等等。现在的个人用户采用最普及隐藏IP的方法应该是使用代理，由于使用代理服务器后，“转址服务”会对发送出去的数据包有所修改，致使“数据包分析”的方法失效。一些容易泄漏用户IP的网络软件(QQ、MSN、IE等)都支持使用代理方式连接Internet，特别是QQ使用“ezProxy”等代理软件连接后，IP版的QQ都无法显示该IP地址。虽然代理可以有效地隐藏用户IP，但攻击者亦可以绕过代理，查找到对方的真实IP地址，用户在何种情况下使用何种方法隐藏IP，也要因情况而论。  
  
8、未加密登录请求   
由于Web配置不安全，登陆请求把诸如用户名和密码等敏感字段未加密进行传输，攻击者可以窃听网络以劫获这些敏感信息。建议进行例如SSH等的加密后再传输。  
  
9、敏感信息泄露漏洞   
SQL注入、XSS、目录遍历、弱口令等均可导致敏感信息泄露，攻击者可以通过漏洞获得敏感信息。针对不同成因，防御方式不同  
  
10、CSRF   
IMG_257http://www.cnblogs.com/hyddd/archive/2009/04/09/1432744.html

http  
1.HTTP协议，即超文本传输协议(Hypertext transfer protocol)。是一种详细规定了浏览器和万维网(WWW = World Wide Web)服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议。  
2.HTTP协议作为TCP/IP模型中应用层的协议也不例外。HTTP协议通常承载于TCP协议之上，有时也承载于TLS或SSL协议层之上，这个时候，就成了我们常说的HTTPS。  
3.HTTP是一个应用层协议，由请求和响应构成，是一个标准的客户端服务器模型。HTTP是一个无状态的协议。  
4.HTTP默认的端口号为80，HTTPS的端口号为443。  
5.浏览网页是HTTP的主要应用，但是这并不代表HTTP就只能应用于网页的浏览。HTTP是一种协议，只要通信的双方都遵守这个协议，HTTP就能有用武之地。比如咱们常用的QQ，迅雷这些软件，都会使用HTTP协议(还包括其他的协议)。  
  
100 继续。客户端应继续其请求  
101 切换协议。服务器根据客户端的请求切换协议。只能切换到更高级的协议，例如，切换到HTTP的新版本协议  
200 请求成功。一般用于GET与POST请求  
201 已创建。成功请求并创建了新的资源  
202 已接受。已经接受请求，但未处理完成  
204 无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档  
300 多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择  
301 永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替  
302 临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI  
303 查看其它地址。与301类似。使用GET和POST请求查看  
304 未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源  
305 使用代理。所请求的资源必须通过代理访问  
307 临时重定向。与302类似。使用GET请求重定向  
400 客户端请求的语法错误，服务器无法理解  
401 请求要求用户的身份认证  
403 服务器理解请求客户端的请求，但是拒绝执行此请求  
404 服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面  
405 客户端请求中的方法被禁止  
408 服务器等待客户端发送的请求时间过长，超时  
413 由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息  
414 请求的URI过长（URI通常为网址），服务器无法处理  
500 服务器内部错误，无法完成请求  
501 服务器不支持请求的功能，无法完成请求  
502 作为网关或者代理工作的服务器尝试执行请求时，从远程服务器接收到了一个无效的响应  
503 由于超载或系统维护，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的Retry-After头信息中  
504 充当网关或代理的服务器，未及时从远端服务器获取请求  
505 服务器不支持请求的HTTP协议的版本，无法完成处理

五：框架

react父子组件的渲染顺序

父componentWillMount

父render

子componentWillMount

子render

子componentDidMount

父componentDidMount

父render

子componentWillReceiveProps

子render

Vue和React的数据绑定对比以单向数据流和单双向数据绑定

https://www.jianshu.com/p/6e124ad23c68

https://www.zhihu.com/question/31585377

在数据绑定上来说，vue的特色是双向数据绑定，而在react中是单向数据绑定。表单的双向绑定，说到底不过是 （value 的单向绑定 + onChange 事件侦听）的一个语法糖

Vue和React遵循从上到下的数据流向，即单向数据流。

Vue 双向数据绑定实现

function defineReactive(obj, key, value){

var dep = new Dep();

Object.defineProperty(obj, key, {

enumerable: true,

configurable: true,

get: function reactGetter(){

if(Dep.target){

dep.depend();

}

return value;

},

set: function reactSetter(newVal){

if (value === newVal) {

return;

} else {

value = newVal;

//如果数据发生改变，则通知所有的 watcher（借助 dep.notify()）

dep.notify();

}

}

})

}

react父组件怎么触发子组件方法

方法一：

<Child onRef={this.onRef} />

this.props.onRef(this)

子组件将this传递给父组件，父组件获取到后触发子组件this上的方法

方法二:

通过componentWillReceiveProps来判断属性是否改变然后触发子组件方法