**前端常见面试**

**技术题**

**一、页面类**

①、**如何解决跨域？ 能说1，2，7，8就行。**

1、 通过jsonp跨域

2、CORS

3、 document.domain + iframe跨域

4、 location.hash + iframe

5、 window.name + iframe跨域

6、 postMessage跨域

7、 nginx代理跨域

8、 nodejs中间件代理跨域

9、 WebSocket协议跨域

**②、前端优化页面（提高页面加载速度）**

（1） 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip，CDN托管，data缓存 ，图片服务器。

（2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数

（3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。

（4） 当需要设置的样式很多时设置className而不是直接操作style。

（5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。

（6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

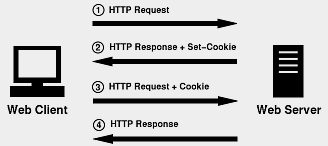
（8）压缩js，css文件，减小文件体积

### **③、Cookie、sessionStorage和localStorage的区别**

Cookie是客户端的会话跟踪技术，用来记录信息确定用户身份。通过document.cookie可以获取或设置Cookie值。

Cookie由”; ”隔开的键值对构成，作用是将网页中的数据保存到浏览器中。Cookie必须从服务器访问，直接打开html文件则设置Cookie无效。

Cookie有个数（20~50）和大小（4KB左右）的限制。默认情况下，Cookie的生命周期是一次会话，如果通过expires设置了过期时间，过期立即删除保存的数据。Cookie由每个对服务器的请求来传递，使得Cookie速度很慢且效率也不高。安全性问题，明文传递。



WebStorage克服了Cookie的一些限制，有两种API：localStorage（本地存储，没有时间限制）和sessionStorage（会话存储，在浏览器关闭后消失）。通过setItem(key, value)、getItem(key)、removeItem(key)、clear()操作数据。要访问同一个WebStorage对象，页面必须来自同一个域名（子域名无效），使用同一种协议，在同一个端口上。

WebStorage只能存储字符串类型，可使用JSON对象的stringify和parse来处理。保存在客户端，不与服务器进行交互通信。

WebStorage存储大小一般是5MB。不会传送到服务器，存储在本地的数据可以直接获取，节省了网络流量，且速度更快。安全性相对Cookie较高。

**④、响应式设计的基本技巧和好处**

响应式设计以媒体查询为基础，通过统一的设计和代码实现让一套Web系统能适配所有可能的终端。不必去为不同设备定制不同的网站，开发成本降低；资源冗余，即有些信息被加载，但是不显示。

（1）布局技巧及meta标签

丢弃：无关紧要的div、内联元素（inline）、少用js或flash、没用的绝对定位和浮动样式、不使用100%设置。

使用：HTML5 Doctype和相关指南、重置样式reset.css、一个简单的有语义的核心布局、给重要的网页元素使用简单的技巧（如导航菜单）。

<meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="HandheldFriendly" content="true">

（2）媒体查询Medial Queries：根据屏幕大小展示适配的样式。

（3）字体设置：单位使用rem，相对于根元素的字体大小。

（4）宽度：使用百分比且box-sizing:border-box;，或flex布局。

（5）图片：将图片的最大宽度设置为100%，背景图片大小设置为contain。根据不同屏幕展示不同大小的图片，将图片地址保存至data-属性中。使用精灵图

**⑤、HTTP常用状态码及其含义？**

1xx：指示信息--表示请求已接收，继续处理

100 Continue 初始的请求已经接受，客户应当继续发送请求的其余部分。（HTTP 1.1新）

101 Switching Protocols 服务器将遵从客户的请求转换到另外一种协议（HTTP 1.1新）

2xx：成功--表示请求已被成功接收、理解、接受

200 OK 一切正常，对GET和POST请求的应答文档跟在后面。

201 Created 服务器已经创建了文档，Location头给出了它的URL。

202 Accepted 已经接受请求，但处理尚未完成。

203 Non-Authoritative Information 文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝（HTTP 1.1新）。

204 No Content 没有新文档，浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的

205 Reset Content 没有新的内容，但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容（HTTP 1.1新）。

206 Partial Content 客户发送了一个带有Range头的GET请求，服务器完成了它（HTTP 1.1新）。

3xx：重定向--要完成请求必须进行更进一步的操作

300 Multiple Choices 客户请求的文档可以在多个位置找到，这些位置已经在返回的文档内列出。如果服务器要提出优先选择，则应该在Location应答头指明。

301 Moved Permanently 客户请求的文档在其他地方，新的URL在Location头中给出，浏览器应该自动地访问新的URL。

302 Found 类似于301，但新的URL应该被视为临时性的替代，而不是永久性的。注意，在HTTP1.0中对应的状态信息是“Moved Temporatily”。出现该状态代码时，浏览器能够自动访问新的URL，因此它是一个很有用的状态代码。注意这个状态代码有时候可以和301替换使用。例如，如果浏览器错误地请求http://host/~user（缺少了后面的斜杠），有的服务器返回301，有的则返回302。严格地说，我们只能假定只有当原来的请求是GET时浏览器才会自动重定向。请参见307。

303 See Other 类似于301/302，不同之处在于，如果原来的请求是POST，Location头指定的重定向目标文档应该通过GET提取（HTTP 1.1新）。

304 Not Modified 客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。

305 Use Proxy 客户请求的文档应该通过Location头所指明的代理服务器提取（HTTP 1.1新）。

307 Temporary Redirect 和302（Found）相同。许多浏览器会错误地响应302应答进行重定向，即使原来的请求是POST，即使它实际上只能在POST请求的应答是303时 才能重定向。由于这个原因，HTTP 1.1新增了307，以便更加清除地区分几个状态代码：当出现303应答时，浏览器可以跟随重定向的GET和POST请求；如果是307应答，则浏览器只能跟随对GET请求的重定向。（HTTP 1.1新）

4xx：客户端错误--请求有语法错误或请求无法实现

400 Bad Request 请求出现语法错误。

401 Unauthorized 客户试图未经授权访问受密码保护的页面。应答中会包含一个WWW-Authenticate头，浏览器据此显示用户名字/密码对话框，然后在填写合适的Authorization头后再次发出请求。

403 Forbidden 资源不可用。服务器理解客户的请求，但拒绝处理它。通常由于服务器上文件或目录的权限设置导致。

404 Not Found 无法找到指定位置的资源。这也是一个常用的应答。

405 Method Not Allowed 请求方法（GET、POST、HEAD、DELETE、PUT、TRACE等）对指定的资源不适用。（HTTP 1.1新）

406 Not Acceptable 指定的资源已经找到，但它的MIME类型和客户在Accpet头中所指定的不兼容（HTTP 1.1新）。

407 Proxy Authentication Required 类似于401，表示客户必须先经过代理服务器的授权。（HTTP 1.1新）

408 Request Timeout 在服务器许可的等待时间内，客户一直没有发出任何请求。客户可以在以后重复同一请求。（HTTP 1.1新）

409 Conflict 通常和PUT请求有关。由于请求和资源的当前状态相冲突，因此请求不能成功。（HTTP 1.1新）

410 Gone 所请求的文档已经不再可用，而且服务器不知道应该重定向到哪一个地址。它和404的不同在于，返回407表示文档永久地离开了指定的位置，而404表示由于未知的原因文档不可用。（HTTP 1.1新）

411 Length Required 服务器不能处理请求，除非客户发送一个Content-Length头。（HTTP 1.1新）

412 Precondition Failed 请求头中指定的一些前提条件失败（HTTP 1.1新）

413 Request Entity Too Large 目标文档的大小超过服务器当前愿意处理的大小。如果服务器认为自己能够稍后再处理该请求，则应该提供一个Retry-After头（HTTP 1.1新）。

414 Request URI Too Long URI太长（HTTP 1.1新）。

416 Requested Range Not Satisfiable 服务器不能满足客户在请求中指定的Range头。（HTTP 1.1新）

5xx：服务器端错误--服务器未能实现合法的请求

500 Internal Server Error 服务器遇到了意料不到的情况，不能完成客户的请求。

501 Not Implemented 服务器不支持实现请求所需要的功能。例如，客户发出了一个服务器不支持的PUT请求。

502 Bad Gateway 服务器作为网关或者代理时，为了完成请求访问下一个服务器，但该服务器返回了非法的应答。

503 Service Unavailable 服务器由于维护或者负载过重未能应答。例如，Servlet可能在数据库连接池已满的情况下返回503。服务器返回503时可以提供一个Retry-After头。

504 Gateway Timeout 由作为代理或网关的服务器使用，表示不能及时地从远程服务器获得应答。（HTTP 1.1新）505 HTTP Version Not Supported 服务器不支持请求中所指明的HTTP版本。（HTTP 1.1新）

**⑥、Http和Https的区别？**

1、https协议需要到ca申请证书，一般免费证书较少，因而需要一定费用。

2、http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。

3、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。

4、http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全

**⑦如何实现浏览器多标签页之间的通信？**

调用localStorage、cookie本地存储方式。

1. **HTML**

### **1、input输入类型并简述其用途**

text：默认，单行文本框，默认宽度为20字符。

radio：单选按钮。

checkbox：复选框。

file：输入字段和“浏览”按钮，供文件上传。

button：可点击按钮。

reset：重置按钮。

submit：提交按钮。

image：图像形式的提交按钮。

password：密码字段，该字段中字符被掩码。

hidden：隐藏的输入字段。

email：email地址。

tel：电话号码。

url：url地址。

number：数值。

range：滑动条，包含一定范围内的数值。

color：选择颜色。

Date pickers(date, month, week, time, datetime, datetime-local)。

search：搜索字段。

**2、<img>的alt和title**

alt：描述图片含义。地址写错或网络状况不好，无法查看图像时，显示该说明。可用于屏幕阅读软件和搜索引擎。

title：光标悬浮在图像上时以提示的方式显示该特性内容。

**3、href和src的区别**

href标识超文本引用，在当前元素和引用资源之间建立联系。<link><a>。当浏览器解析到css文件时会下载，但不会停止对当前文档的处理。

src引用资源，指向外部资源的位置，将外部资源下载并替换当前元素。用在<img><script><iframe>。当浏览器解析到含src的标签时，会暂停其它资源的下载和处理，直至将该资源加载、编译、执行完毕，将该元素所指向的资源嵌套至当前标签内。

**4、页面导入样式时，使用link和@import有什么区别?**

　　差别1：本质的差别：link属于XHTML标签，而@import完全是CSS提供的一种方式。

　　差别2：加载顺序的差别：当一个页面被加载的时候(就是被浏览者浏览的时候)，link引用的CSS会同时被加载，而@import引用的CSS会等到页面全部被下载完再被加载。所以有时候浏览@import加载CSS的页面时开始会没有样式(就是闪烁)，网速慢的时候还挺明显。

　　差别3：兼容性的差别：@import是CSS2.1提出的，所以老的浏览器不支持，@import只有在IE5以上的才能识别，而link标签无此问题。

　　差别4：使用dom(document o bject model文档对象模型 )控制样式时的差别：当使用javascript控制dom去改变样式的时候，只能使用link标签，因为@import不是dom可以控制的.

**三、CSS**

**1、清除浮动的几种方式，各自的优缺点**

清除浮动：使父元素围住浮动的子元素，避免对后续元素造成影响。

（1）给父元素设置overflow:hidden;或float:left;或position:absolute;或position:relative;。触发块级格式化上下文（BFC），从正常文档流脱离，父元素本身实现独立布局，围住其中的浮动元素。但专门这样设置，可能会改变布局，对于后三者，使block变为inline-block，父元素宽度变化。

适用于父元素也需要浮动。

（2）在子元素后面添加空的block元素，并设置其样式为clear:both。添加了额外的元素，且block元素带有margin等。

（3）在父元素上加clearfix类名，在父元素的最后加块级元素.（伪元素选择器），通过clear:both使父元素括住浮动的元素来清除float的影响，使.不可见。常用这种方式，Bootstrap支持clearfix类名。

1. **块级元素水平垂直居中的方法**
2. Flex布局，justify-content: center;align-items: center;

**3、CSS选择器有哪些？优先级如何计算**

（1）普通选择器：通配符选择器\*、标签选择器、id选择器#、类选择器.、属性选择器[]、伪类选择器:

（2）伪元素选择器::

（3）组合选择器：后代选择器 、子选择器、兄弟选择器+、只要前面有某种元素~

优先级：将权重相加。a=行内样式style，1000；b=id选择器，100；c=类、伪类和属性选择器，10；d=标签选择器和伪元素选择器，1。相同属性优先级高覆盖低、后面覆盖前面，不同属性合并。

超链元素伪类的设置顺序:link :visited :hover :active

**4、px、em和rem的区别**

px，像素，相对于显示器屏幕分辨率而言；

em，相对长度单位，相对于当前对象内文本的字体大小，若字体大小未显式设置，则相对于浏览器默认的字体大小；

rem，相对长度单位，相对于html根元素的字体大小。

**5、display:none和visibility:hidden的区别**

display:none设置元素不显示，是彻底消失，不在文档流中占位，浏览器也不会解析该元素。visibility:hidden是视觉上消失，可理解为透明度为0的效果，在文档流中占位，浏览器会解析该元素。

使用visibility:hidden比display:none性能上要好。display:none切换显示时，页面产生回流reflow（页面中的部分元素需要改变规模尺寸、布局、显示隐藏等，页面重新构建）。而visibility:hidden切换是否显示时则不会引起回流，只是元素的外观被改变，且在没有改变布局的情况下发生。

**6、用一个div写出三条横线的小图标**

div{

height:5px;

width:30px;

background-color:#DD575C;

background-clip:content-box;

padding-top:5px;

padding-bottom:5px;

border-top:#DD575C solid 5px;

border-bottom:#DD575C solid 5px;

}

**7、用一个div实现倒三角**

div{

width:0;

border-top:10px solid #DD575C;

border-left:5.77px solid transparent;

border-right:5.77px solid transparent;

}

**8、position的absolute和fixed的共同点与不同点**

共同点：脱离文档流，文档流中没有它的空间，未设置偏移时在原位置。默认宽度为内容宽度。

absolute：绝对定位。参照物为第一个定位元素（非static）/根元素html（与浏览器窗口同等大小的首屏区域）。元素随页面的滚动而滚动。

fixed：固定定位。参照物是视窗。不随页面的滚动而滚动。

**9、CSS3**

选择器：属性选择器、伪类选择器

盒模型：box-shadow、box-sizing

边框：border-radius、border-image

背景：background-image、background-size、background-origin、background-clip、渐变

文字特效：text-shadow、text-overflow、word-wrap、word-break、字体@font-face

2D/3D转换：transform、transform-origin、transform-style、perspective、perspective-origin、backface-visibility

过渡和动画：transition、animation、@keyframes

多列布局：column-count、column-width、column-fill、column-gap、column-rule、column-span

用户界面：resize、outline-offset 弹性盒子：flex

**10、盒子内阴影和外阴影**

box-shadow: inset? 水平偏移 垂直偏移 模糊半径? 阴影大小? 颜色?（？表示可选；水平、垂直偏移可正可负；颜色默认为文字颜色；这些参数可有多组，用,隔开）

外阴影box-shadow:3px 3px 5px 2px; 内阴影box-shadow:inset 0 0 5px red;

**11、如何设置两种模型**

div {

// 设置标准模型

box-sizing: content-box;

// 设置IE模型

box-sizing: border-box;

}

# **12、CSS“隐藏”元素方法的对比**

1:display:none

　将元素设置为display:none后，元素在页面上将彻底消失，元素本来占有的空间就会被其他元素占有，也就是说它会导致浏览器的重排和重绘。

2:visibility:hidden

　 设置元素的visibility为hidden也是一种常用的隐藏元素的方法，和display:none的区别在于，元素在页面消失后，其占据的空间依旧会保留着，所以它只会导致浏览器重绘而不会重排。

3:opacity:0

这种方法和visibility:hidden的一个共同点是元素隐藏后依旧占据着空间，但我们都知道，设置透明度为0后，元素只是隐身了，它依旧存在页面中。

**四、JavaScript**

### **1、==和===的区别**

===判断左右两边对象或值是否类型相同且值相等。==判断操作符两边对象或值是否相等，类型不同时，使用Number()转换。例外规则，null==undefined，null/undefined进行==运算时不进行隐式类型转换。通常把值转为Boolean值，进行条件判断。

**2、比较typeof和instanceof**

（1）typeof运算符 typeof 变量名; //返回字符串 可以识别标准类型（Null除外），不能识别具体的对象类型（Function除外）。

（2）instanceof 变量名 instanceof 类型; 判别内置对象类型、自定义对象类型，不能判别标准类型。给定引用类型的实例则返回true，检测基本类型值返回false。

### **3、闭包**

闭包是指有权访问另一个函数作用域中的变量的函数，由函数和与其相关的引用环境组成。闭包允许函数访问其引用环境中的变量，有可能是外部环境变量，该变量称为自由变量。创建闭包的常见方式，在一个函数内部创建另一个函数。

闭包的应用，保存变量现场。闭包只能取得包含函数中任何变量的最后一个值，但可以强制创建另一个匿名函数强制让闭包的行为符合预期。

闭包的应用，私有作用域的匿名函数。临时需要一些变量，就可以使用私有作用域，限制向全局作用域中添加过多的变量和函数。私有变量包括函数的参数、局部变量和在函数内部定义的其他函数。

### **4、call、apply和bind的区别**

函数调用时绑定this值：call()、apply()让函数在特定的作用域下进行调用，即设置函数体内this对象的值。扩充函数运行的作用域，对象不需要与方法有任何耦合关系。区别仅在于接受参数的方式不同， call()方法传递的参数逐个列举，第一个参数是作用域；apply()方法接收两个参数，作用域和参数数组。

为函数绑定this值但并不调用：bind()传入新作用域，返回新函数，该函数在bind作用域执行。第二个参数是参数列表，与call一致。

**5、垃圾回收机制和内存管理**

js具有自动垃圾回收机制，找出不再使用的变量，垃圾收集器按一定时间间隔（影响性能）释放其占用的内存。“标记清除”是目前主流的垃圾回收算法，给当前不使用的值加上标记，再回收内存。另一种算法是“引用计数”，跟踪记录所有值被引用的次数，当代码中存在循环引用现象时会导致问题。及时解除不再使用的全局对象、全局对象属性及循环引用变量的引用，确保有效回收内存。

1. 谈谈对ES6 async函数的认识

解决了异步回调问题，同步流程表达异步操作

async 函数中可能会有 await 表达式，async 函数执行时，如果遇到 await 就会先暂停执行 ，等到触发的异步操作完成后，恢复 async 函数的执行并返回解析值。

await 关键字仅在 async function 中有效。如果在 async function 函数体外使用 await ，你只会得到一个语法错误

1. call和apply的区别？

**共同点：**

都可以用来代替另一个对象调用一个方法，将一个函数的对象上下文从初始的上下文改变为由thisObj指定的新对象。

另一种说法，都能够改变方法的执行上下文（执行环境），将一个对象的方法交给另一个对象来执行，并且是立即执行。

**不同点：**

apply();//最多只能有两个参数--新this对象和一个数组argArray，如果给该方法传递多个参数，则把参数都写进这个数组里边，当然，即使只有一个参数，也要写进数组里边。

call();//可以接收多个参数，第一个参数apply()一样，后面则是一串参数列表。

实际上，apply和call的功能是一样的，只是传入的参数列表的形式不同。

1. **JavaScript DOM**

**1、节点操作**

（1）获取节点 getElementById getElementsByTagName getElementsByClassName

querySelector/querySelectorAll 获取指定元素的后代元素中符合选择器的节点，第一个符合条件的元素或所有元素的列表

（2）创建节点 createElement createDocumentFragment

（3）修改节点内容 textContent innerText

（4）插入节点 appendChild insertBefore

（5）删除节点 removeChild replaceChild

**2、样式操作**

（1）增加内联样式 element.style.color='red'; element.style.cssText='border-color:red;color:red;';

（2）更新class element.className+='invalid';

（3）更换样式表

**3、offsetWidth/clientWidth/scrollWidth的区别**

offsetWidth：元素在水平方向上占用的空间大小。

clientWidth：元素内容区的宽度加左右内边距的宽度。

scrollWidth：在没有滚动条的情况下，元素内容的总宽度。

**4、什么是事件流？**

事件流是指从页面接收事件的顺序。也就是说，当一个事件发生时，这个事件的传播过程就是事件流。

**5、什么是事件冒泡？**

IE中的事件流就叫做事件冒泡。事件冒泡是由事件开始时由最具体的元素接收，然后逐级向上传播到较为不具体的节点。对于HTML来说当以一个元素产生一个事件时，它会把这个事件传递给它的父元素，父元素接收到之后，还要传递给它上一级的元素，就这样一直传播到document对象。

**6、事件冒泡机制？**

当一个对象上触发了某种事件，此对象定义了此事件的处理程序，那么就会调用这个处理程序；如果没有定义处理程序或者事件返回true，那么这个事件会向父级对象传播，从里到外，直到它被处理，或者到达对象层次的最顶层

**7、什么是事件捕获**

事件捕获就是值不太具体的元素更早的接收到事件，而最具体的节点最后接收到事件。它们的用意就是在事件达到目标之前就捕获它，过程与冒泡相反。

**8、事件**

|  |  |
| --- | --- |
| onload 和 onunload | 用户进入或离开页面 |
| onmouseover 和 onmouseout | 用户的鼠标移至 HTML 元素上方或移出元素 |
| onmousedown, onmouseup ，onclick | 点击鼠标按钮，释放鼠标按钮，完成鼠标点击 |
| onkeydown、onkeyup、onkeypress | 按下一个键，松开一个键，按下再松开一个键 |
| onfocus,onblur | 光标聚焦,光标离开 |
| oncontextmenu | 单击鼠标右键 |
| onchange | 文本框内容改变 |
| onselect | 文本框内容选中 |
| oncopy,oncut,onpaste | 用户拷贝元素内容,用户剪切元素内容,用户粘贴元素内容 |
| onsubmit | 表单提交 |

1. **Jquery**
2. **事件委托其实是利用的事件冒泡的机制来实现的**  
   原理：把事件加在父元素上，在父元素里判断事件源（目标元素）是否为你想要触发的元素标签，如果是才触发，不是不触发  
   事件冒泡：子元素事件触发后，会依次调用父级元素的同名事件，直到window，事件冒泡默认就存在

**2、使用jQuery操作 DOM**

**1、样式操作**

追加样式

$( ).addClass ( )

删除样式

$( ).removeClass( )

判断样式

$( ).hasClass ( )

切换样式

$( ).toggleClass ( )

**2、内容操作**

代码操作：html( )

内容操作：text( )

属性值操作：val( )

**3、属性操作**

获取和设置元素属性：attr( )

删除元素属性：removeAttr( )

**4、动画**

$( ).animate(styles,speed,easing,callback)

第一个参数：对象，里面可以传需要动画的样式。

第二个参数：speed 动画的执行时间。

第三个参数：动画的执行效果。

第四个参数：回调函数。

**动画停止**

$( ).stop(stopAll,goToEnd)

**3、jquery操作节点**

**（1）文本内容的操作**

方法：①text() ②text(val)

例子：$div = $(‘div’) //获取div元素

Console.log($div.text()) //获取文本内容

$div.text(“aaaaa”) //设置了文本的内容

**（2）HTML内容的操作**

①html():用于获取第一个匹配元素的HTML内容

②html(val)：用于设置全部匹配元素的HTML内容

注意：内容中包含HTML代码可以被浏览器解析

**（3）对元素值的操作**

主要对于input、check、radio元素

①val() //获取值

②val(“”) //设置值

**（4）对于DOM节点的操作**

①创建元素节点：

Var $jsp = $(“<p></p>”) //创建了一个p节点

插入到div里面

$(“div”).append($jsp)

②插入节点：

方法：1）往元素的内部插入

append():为所有匹配的元素的内部追加内容

appendTo()：将所有匹配的元素添加到另一个元素集合中

prepend()：往元素的集合的前面添加内容

prependTo()

2）往元素的外部插入

after():在每个匹配的元素之后插入

insertAfter()：

before()：在每个匹配的元素前边插入

insertBefore()

③删除节点：

remove():用于从DOM中删除匹配的元素（常用）

detach()：也是用于删除匹配的元素，与remove不同的是，detach不会将jQuery中的对象删除。

empty()：把标签的内容清空掉，不是绝对意义上的删除

④复制节点

clone():

示例：$(“#box1 p:eq(1)”).bind(“click”,function(){

$(this).clone().insertAfter($(this))

})

带参数:clone(True):加了这样之后会将事件处理程序一并复制

⑤替换节点

replaceAll(Seletor):用于使所有的元素替换掉Seletor匹配的元素

示例：$(“<div>good</div>”).replaceAll(“div”)

replaceWith(content):用匹配的元素替换掉指定的HTML元素

示例：$(“#box1”).replaceWith(“<div>sunck</div>”)

⑥遍历节点

each():

示例：

$(“p”).each(function(index){ //index表示第几个

$(this).attr(“title”,”我是第”+(index+1)+”个p标签”)

})

⑦包裹节点：

wrap():把匹配到的标签用另一个标签包裹起来

unwarp():去除包裹

warpAll():整体包裹

warpInner():包裹内部

**（5）修改CSS样式**

addClass(class)

removeClass(class)

toggleClass(class):若有这个class类就删除，若无就添加

**（6）修改CSS的属性**

方法：css(attr,value)

示例：

$(“#btn”).bind(“click”,function(){

$(“#box2”).css(“background-color”,”red”) //修改背景颜色的属性

console.log($(“#box2”).css(“background-color”)) //获取值

})

1. **关于Ajax**
2. **ajax请求时get和post的区别？**

get：从服务器上获取数据，传送数据量小，安全性低，请求会被缓存，缓存是针对URL进行缓存的，get请求参数直接加在URL地址后面，一种参数组合就会产生一种URL的缓存，重复的请求结果是相同的；

post：向服务器发送数据；传送数据量大，请求不会被缓存，参数封装在二进制的数据体中，服务器也不会记录参数，相对安全，所以涉及用户隐私的数据都要用post传送；

**2、ajax请求时，如何解释json数据？**

使用eval方法解析的时候，eval();不会去判断该字符串是否合法，而且json对象里的js方法也会被执行，这是非常危险的；推荐使用JSON.parse(); JSON.parse();把字符串转化成json。

**3、Axios配置请求头**

axios.defaults.withCredentials = false;

axios.defaults.headers.common['token'] = token;

axios.defaults.headers.post['Content-Type'] = 'application/json;charset=UTF-8';//配置请求头

axios.interceptors.request.use

请求拦截器

加一个http request拦截器

通过window.localStorage.getItem("accessToken") 来获取token的value

通过config.headers.accessToken = token;将token放到请求头发送给服务器，放在请求头

axios.interceptors.response.use

响应拦截器

请求失败

请求成功

1. **Vue+Vuex**

**1、重中之重、Vue的双向数据绑定原理是什么？**

vue.js 是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。

具体步骤：

第一步：需要observe的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter和getter

这样的话，给这个对象的某个值赋值，就会触发setter，那么就能监听到了数据变化

第二步：compile解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图

第三步：Watcher订阅者是Observer和Compile之间通信的桥梁，主要做的事情是:

1、在自身实例化时往属性订阅器(dep)里面添加自己

2、自身必须有一个update()方法

3、待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。

第四步：MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者，通过Observer来监听自己的model数据变化，通过Compile来解析编译模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据model变更的双向绑定效果。

**2、什么是MVVM？**

MVVM是Model-View-ViewModel的缩写。MVVM是一种设计思想。Model 层代表数据模型，也可以在Model中定义数据修改和操作的业务逻辑；View 代表UI 组件，它负责将数据模型转化成UI 展现出来，ViewModel 是一个同步View 和 Model的对象。

在MVVM架构下，View 和 Model 之间并没有直接的联系，而是通过ViewModel进行交互，Model 和 ViewModel 之间的交互是双向的， 因此View 数据的变化会同步到Model中，而Model 数据的变化也会立即反应到View 上。

ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作DOM, 不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

**3、mvvm和mvc区别？它和其它框架（jquery）的区别是什么？哪些场景适合？**

mvc和mvvm其实区别并不大。都是一种设计思想。主要就是mvc中Controller演变成mvvm中的viewModel。mvvm主要解决了mvc中大量的DOM 操作使页面渲染性能降低，加载速度变慢，影响用户体验。

区别：vue数据驱动，通过数据来显示视图层而不是节点操作。

场景：数据操作比较多的场景，更加便捷

**4、vue的优点是什么？**

低耦合。视图（View）可以独立于Model变化和修改，一个ViewModel可以绑定到不同的"View"上，当View变化的时候Model可以不变，当Model变化的时候View也可以不变。

可重用性。你可以把一些视图逻辑放在一个ViewModel里面，让很多view重用这段视图逻辑。

独立开发。开发人员可以专注于业务逻辑和数据的开发（ViewModel），设计人员可以专注于页面设计。

可测试。界面素来是比较难于测试的，而现在测试可以针对ViewModel来写。

**5、 组件之间的传值？**

父组件与子组件传值

父组件通过标签上面定义传值

子组件通过props方法接受数据

子组件向父组件传递数据

子组件通过$emit方法传递参数

**6、路由之间跳转**

声明式（标签跳转） 编程式（ js跳转）

**7、vue.cli中怎样使用自定义的组件？有遇到过哪些问题吗？**

第一步：在components目录新建你的组件文件（indexPage.vue），script一定要export default {}

第二步：在需要用的页面（组件）中导入：import indexPage from ‘@/components/indexPage.vue’

第三步：注入到vue的子组件的components属性上面,components:{indexPage}

第四步：在template视图view中使用，

例如有indexPage命名，使用的时候则index-page

**8、vue如何实现按需加载配合webpack设置**

webpack中提供了require.ensure()来实现按需加载。以前引入路由是通过import 这样的方式引入，改为const定义的方式进行引入。

不进行页面按需加载引入方式：import home from '../../common/home.vue'

进行页面按需加载的引入方式：

const home = r => require.ensure( [], () => r (require('../../common/home.vue')))

**9、vuex面试相关**

**（1）vuex是什么？怎么使用？哪种功能场景使用它？**

vue框架中状态管理。在main.js引入store，注入。新建一个目录store，…… export 。场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

**（2）vuex有哪几种属性？**

有五种，分别是 State、 Getter、Mutation 、Action、 Module

**vuex的State特性**

A、Vuex就是一个仓库，仓库里面放了很多对象。其中state就是数据源存放地，对应于一般Vue对象里面的data

B、state里面存放的数据是响应式的，Vue组件从store中读取数据，若是store中的数据发生改变，依赖这个数据的组件也会发生更新

C、它通过mapState把全局的 state 和 getters 映射到当前组件的 computed **计算属性中**

**vuex的Getter特性**

A、getters 可以对State进行计算操作，它就是Store的计算属性

B、 虽然在组件内也可以做计算属性，但是getters 可以在多组件之间复用

C、 如果一个状态只在一个组件内使用，是可以不用getters

**vuex的Mutation特性**

Action 类似于 mutation，不同在于：Action 提交的是 mutation，而不是直接变更状态；Action 可以包含任意异步操作。

**（3）不用Vuex会带来什么问题？**

可维护性会下降，想修改数据要维护三个地方；

可读性会下降，因为一个组件里的数据，根本就看不出来是从哪来的；

增加耦合，大量的上传派发，会让耦合性大大增加，本来Vue用Component就是为了减少耦合，现在这么用，和组件化的初衷相背。

**10、 v-show和v-if指令的共同点和不同点**

v-show指令是通过修改元素的display的CSS属性让其显示或者隐藏

v-if指令是直接销毁和重建DOM达到让元素显示和隐藏的效果

**11、计算属性computed和watch的区别**

通俗的来说既能实现computed又能够实现watch的推荐使用computed，重点在于computed的缓存功能。computed是用来声明式的描述一个值依赖其它的值，当所有依赖的值或者变量改变的时候，计算属性也会随着该变；watch主要是监听data里面的定义的量，当该变量变化的时候，触发watch

两者的主要区别

计算属性的依赖值改变会重新执行函数，计算属性是取返回值来作为最新的结果，所以里面不能异步的返回结果（一定要多注意，吃的亏越多，越明白它的重要性），不能写异步逻辑

watch函数的值改变watch重新执行，将一个值重新赋值作为最新结果，所以赋值的时候可以进行一些异步操作。

**12、计算属性和methods的区别：**

计算属性一般就是用来通过其他的数据计算出一个新数据，而且它有一个好处就是它把新的数据缓存下来了，如果data中相关的数据没有发生改变，那么再次调用它的时候会返回它的缓存的数据，这就极大的提高了我们程序的性能；如果写在methods里，每次调用都会重新计算，都是一次新的调用，不会缓存，所以用计算属性比较好，因为有缓存。

1. **ES6**

**1.var、let、const区别**

var 函数级作用域 可变量提升 值可以改变

let 块级作用域 不存在变量提升 值可以改变

const 块级作用域 不存在变量提升 值不可以改变

**2.使用箭头函数需要注意什么？**

（1）this的值不是指向windows,而是父级

（2）不能够使用arguments对象

（3）不能构造函数（不能使用new命令）

（4）不可以使用yield命令，因此箭头函数不能用作Generator函数

**3.es6的模板字符串有什么新特点？**

将表达式嵌入字符串中进行拼接 使用${}

在es5时我们通过反斜杠来做多行字符串或者字符串一行行拼接。es6用()就能够解决

**4.什么是展开运算符 ···**

可以将数组或对象里的值展开

也可以将多个值收集为一个变量

**5.promise构造函数是同步执行还是异步执行？then呢**

promise构造函数是同步执行

then是异步执行

const promise = new Promise((resolve, reject) => {

console.log(1);

resolve();

console.log(2)

})

promise.then(() => {

console.log(3)

})

console.log(4)

在这段代码中输出的结果是1 2 4 3 ，因为promise新建后立即执行，promise.then()内部代码在当次事件的结尾立刻执行 所以会先输出4后输出3.

**6.forEach、for in 、for of 三者区别？**

forEach 更多用来遍历数组

for in 常用遍历对象或json,循环出来的是key

for of 数组对象都可以遍历，遍历对象通过和object.keys(),循环出来的是value

**7.es6的导入导出模块分别采取什么关键字**

import { sum} from “./example.js”

**8.set与map的区别？**

set用于数据重组 ，map数据存储

set应用场景：

（1）成员不能重复

（2）只有键值没有键名，类似数组

（3）可以遍历，方法有add delete has

map应用场景

（1）本质上是键值对的合集，类似集合

（2）可以遍历，可以跟各种数据格式转换

**9.如何使用set去重？**

let arr = [12, 43, 23, 43, 34, 68, 12];

let item = [...new Set(arr)];

console.log(item) //12 43 23 34 68。

1. **map和foreach的区别**

map()会分配内存空间存储新数组并返回，forEach()不会返回数据。

forEach()允许callback更改原始数组的元素。map()返回新的数组。