·HTML

浏览器的页面分为那三层

结构层（HTML）、表示层（CSS）、行为层（JavaScript）

HTML实现页面结构，CSS完成页面的表示与风格、JavaScript实现一些客户端的功能与业务

Doctype 作用？严格模式与混杂模式如何区分？他们有何意义？

 (1) 声明位于文档中的最前面的位置，处于标签之前。此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范来解析这个文档。该标签可声明三种 DTD 类型，分别表示严格版本、过渡版本以及基于框架的HTML 文档。

(2)所谓的标准模式是指，浏览器按 W3C 标准解析执行代码；混杂模式则是使用浏览器自己的方式解析执行代码，因为不同浏览器解析执行的方式不一样，所以我们称之为混杂模式。 严格模式是浏览器根据web标准去解析页面，是一种要求严格的DTD，不允许使用任何表现层的语法，

严格模式的排版和JS 运作模式是以该浏览器支持的最高标准运行。混杂模式则是一种向后兼容的解析方法，说的透明点就是可以实现IE5.5以下版本浏览器的渲染模式。

(3)浏览器解析时到底使用标准模式还是混杂模式，与你网页中的 DTD 声明直接相关， DTD 声明定义了标准文档的类型（标准模式解析）文档类型，会使浏览器使用相应的方式加载网页并显示，忽略 DTD 声明 ,将使网页进入混杂模式。

HTML5有哪些新特性、移除了哪些元素？

* 图像Canvas
* 多媒体video、audio
* 本地存储localStorage、sessionStorage
* 语义化更好的内容元素aticle、header、footer、nav、section
* 表单控件date、time、canlendar、url、search
* 新的技术webworker、websocket、Geolocation

移除的元素：

* 纯表现的元素u、big、center、strike、tt、font、basefont
* 框架集frame、frameset、noframes

浏览器的内核分别是

a、 IE: trident 内核

b、 Firefox ： gecko 内核

c、 Safari: webkit 内核

d、 Opera: 以前是 presto 内核， Opera 现已改用 Google Chrome 的 Blink 内核

e、 Chrome: Blink( 基于 webkit ， Google 与 Opera Software 共同开发 )

对Web标准以及W3C的理解与认识？

Web标准是一系列标准的集合，主要包括结构标准化语言html、XHTML、XML等、表现标准语言css、行为标准语言JavaScript等，这些标准语言大部分由万维网起草和发布

W3C：万维网联盟、是一个web开发的国际性联盟

XHTML：要求标签闭合、标签小写、不乱嵌套

HTML5行内元素、块级元素、空元素有哪些？

行内元素： a、img、input、span、strong（等修饰性的元素）

块元素: div form h1-h6 hr p ul ol

可变元素：button

空元素：br、hr、input、img、link、meta

Cookie 和 localStorage 和SessionStorage的区别？

Cookie是网站为了表示用户身份而存储在用户本地终端的数据，通常在同源的http请求中携带，在浏览器和服务器之间传递。Cookie的数据大小很小不超过4K。Cookie在设置的Cookie过期前一直有效。

local Storage和sessionStorage是H5提供的，可以在浏览器端存储数据、可以避免数据在浏览器和服务器之间来回传递。数据大小大约5M左右。

sessionStorage是在同源同窗口之间一直存在的数据，关闭窗口后数据立马被销毁。

localStorage 用于持久话的本地存储、除非手动删除，否则就会永久存在

对HTML语义化的理解？

HTML5引入了很多语义化的标签比如header\footer\nav\aside\section\article等

1、为了在没有CSS的情况下页面也能呈现出良好的页面解构

2、为了方便SEO和搜素引擎建立良好的沟通，有助于爬虫抓取更多有效的信息。

3、为了便于团队的维护和开发

link和import的区别？

link和import都是外部引入CSS的方式、存在一下区别:

1、link是XHML标签、在页面加载的同时加载，而import需要页面完全载入后加载

2、link是html标签，不存在兼容性问题，import是CSS2.1提出的，不适合低版本的浏览器

3、link支持JavaScript控制DOM去改变样式，但是import不支持

SVG的理解？

SVG是可缩放矢量图形，可以允许用户任意缩放图像显示而且不破坏图片的清晰度。

SVG图像文件比GIF和JPEG格式的文件小很多，而且下载很快

SVG图像文本独立，文字独立于图像，文字保留可编辑和可搜素的状态，也不再有字体限制

HTML的全局属性有哪些？

class规定元素的一个或多个类名 draggable规定元素是否可拖lang 规定元素内容的语言

id 规定元素的唯一id style 规定元素CSS行内元素 title 规定有关元素的额外信息

accesskey 规定激活元素的快捷键；contenteditable 规定元素内容是否可编辑

data-\* 用于存储页面或应用程序的私有定制数据。dir 规定元素中内容的文本方向

超链接target属性的取值和作用？

target属性指定所连接的页面在浏览器的打开方式

\_blank 在新的浏览器窗口打开文件

\_self 在同一浏览器窗口打开文件，为默认值

\_top 在当前整个浏览器窗口打开所链接的文件，因而会删除所有的框架

\_parent 将链接的文件载入含有该链接框架的父框架集或父窗口中。没有则同 \_self

data\_\*属性 的理解？

data\_\*属性允许在标准HTML元素中存储私额外信息，data\_\* 长常用来设置一些自定义属性。

浏览器内核的理解？

主要分为渲染引擎和JS引擎。

渲染引擎负责获取页面的XML、HTML等内容以及CSS等，然后会输出至显示器。浏览器内核不同对页面语法解释不同渲染效果就不同。

JS引擎：解析和执行JavaScript来实现页面的动态效果。后来JS越来越独立，就倾向于只指渲染引擎。

IE的内核为Trident。Safi的内核为ｗｅｂｋｉｔ　而谷歌和欧鹏的内核为Blink基于（ｗｅｂｋｉｔ）fire fix的内核为　gecko

iframe有那些缺点？

１、iframe会产生很多页面，不容易管理，iframe会让代码复杂，不利于搜索引擎优化。

２、iframe会阻碍ｏｎｌｏａｄ时间，现在iframe基本被Ajax替代。

label标签的作用

label标签定义表单控件之间的关系，当用户选择该标签的时候，浏览器会自动将焦点转到和标签相关的表单空间上。

一般使用一对label标签包裹表单控件。

如何实现浏览器标签的多个页面之间的通信？

可以使用localStorage等本地存储方式

localStorage在浏览上下文里被添加修改或删除的时候就会触发一个事件，可以监听该事件，控制他的值来实现页面信息的通信。

src和href的区别？

src用于指向当前元素，href用于建立当前文档和引用资源之间的联系。

src指向外部资源的位置，指向的内容会嵌入到文档中并代替当前标签的位置。在请求src资源的时候会将指向的资源下载并应用到文档中。

herf是指向网络资源的位置，建立和当前文档之间的链接。

canvas的理解？

canvas是H5新增的标签。canvas相当于一个画布，其本身没有画图能力，是使用JavaScriptAPI操作，可以在网页中实现2D或者3D图像技术

表单的基本组成部分有哪些，表单的主要用途是什么？

组成：表单标签，表单域，表单按钮

表单标签：包含处理表单数据所用CGI程序的URL以及数据提交到服务器的方法。

表单域：包含了文本框、密码框、多行文本、复选框、单选框等

表单按钮：包括提交按钮、复位按钮一般按钮。

作用: 表单在网页中负责数据采集的功能，和向服务器传送数据。

Get和Post的区别？

get是常用于向服务器查询某些信息，可以将要查询的数据放在URL的末尾，发送给服务器。

post 是常用于向服务器发送应该保存的数据，post请求应该把数据作为请求的主体提交，post请求的主题可以包含非常多的数据而且格式不限。

get 是从服务器上获取数据，post是向服务器传送数据

get是将数据放在URL中，在URL中可以看到数据， Post是将表单各个字段放到Header内，用户看不到

get传送的数据量小安全性低，post传送的数据量大安全性高

HTML5提供了哪些新的API？

Media API /User Interaction/History API/Command API/ Data Transfer API/Constraint Validation API /Text Track API

HTML5 应用程序缓存和浏览器缓存有什么区别？

HTML5 的应用程序缓存即appcache，就是从浏览器的缓存中分出来一部分缓存区，在这个区内保存应用的数据。

要是有描述文件实现，在描述文件内列出要下载和缓存的资源，

要将描述问价与页面关联起来， 在html的manifest文件中指定这个文件的路径。

浏览器缓存包括cookie、以及localStorage和sessionStorage等。

应用程序缓存是HTML5的重用特征之一，提供了离线使用的功能，让应用程序可以获取本地网站的内容，提高了网页性能。

他的实现借助于 manifest文件。

与传统浏览器缓存相比，它不强制用户访问的网站内容被缓存。

CSS

CSS盒模型？

标准的CSS盒子模型包括 content padding border margin。分为IE盒模型和W3C 的盒模型， 其中IE的盒模型指content部分包括了padding和border

W3C的盒模型content只包括content

就相当于box-sizing设置和模型的类型 content-box指W3C的，而IE的是border-box

CSS的选择器？

标签选择器：div p

类选择器 .class

id选择器 #id

组合选择器

伪类选择器： a: visited

伪元素选择器： first-child

优先级：！important> 内联样式（1000）>ID 选择器 （100）> 伪类选择器 >类选择器（10）>标签选择器（1）>通用选择器（\*）

同类型同级别的样式后者优于前者。

常见浏览器兼容性问题？

问题1、随便写的几个标签在不加限制的情况下各自的marginpadding差异较大

解决：在CSS内设置初始样式，可以使用\*{margin：0，padding：0}

问题2、块属性标签float后，又有横行的margin情况下，在IE6显示margin比设置大（后面的内容被顶到下一行）

解决：在float标签样式控制中加入display：inline，将其转化为行内元素。

问题 3：图片默认有间距，几个图片放在一起后有些浏览器会有默认的间距。

解决： 设置fon-size=0

清除浮动的方法：

1.浮动元素后面加一个空的div，然后clear：both

2. 给浮动的父元素加上overflow： hidden（浮动的元素没有定义height，使用overflowhidden的时候浏览器会自动检查浮动的高度）

3. 给浮动的元素加上一个伪类clearfix .clearfix: display: inline-block .clearfix:after{height:0;line-height:0; content=””;clear:both}

block inline 和inline-block的对比？

block是块级元素，每个block元素的都会独占一行，可以设置宽度高度的值。也可以设置padding和margin

inline元素不会独占一行、设置宽高无效、设置上下边距和填充无效

inline-block 既有block的宽高特性也有inline的同行特性

浮动带来的问题以及解决方案？

1、父元素的高度无法被撑开、影响父元素同级的元素

2、与浮动元素同级的元素会紧随其后

解决方式见清除浮动的方式

性能优化的方法？

减少http请求次数

1、规划好CSS代码，减少CSS文件个数和文件大小，用link代替import。

2、背景整合，对于多个很小的图片使用精灵图。

3、使用外部的css和js，样式放在页面顶部，脚本放在页面的底部。

４、合理设置适当的缓存。

５、减少DOM的访问，设置合理的变量存储内容。

用innerHTML代替DOM操作，减少DOM操作次数。

样式CSS放在顶部、将JS代码放在页面底部

边距折叠的理解？

相邻两个多个外边距在垂直方向上会合并成一个外边距

a、参加折叠的margin都是正值：取其中 margin 较大的值为最终 margin 值。

b、参与折叠的 margin 都是负值：取的是其中绝对值较大的，然后，从 0 位置，负向位移。

c、参与折叠的 margin 中有正值，有负值：先取出负 margin 中绝对值中最大的，然后，和正 margin 值中最大的 margin 相加

隐藏元素的方式有哪些？

display：none 不会占有原来的位置

visibility：hidden 会占有原来的位置

opacity： 0 占有原来的位置，不会改变网页布局、可以触发点击事件

几个常用的BOM对象？

location　history　navigator

浏览器在构建DOM的时候，如果在HTML中遇到一个<script>…</script>标签，他必须立即执行。如果脚本是来自外部的，那么他必须先下载脚本

为什么之前遇到js脚本就会立即暂停解析，而是等待JavaScript引擎执行完js代码呢？

是因为脚本可以改变HTML以及它的产物DOM。

脚本可以通过document.creatElement()方法添加节点来改变DOM的结构，也可以使用document.write()的方式来改变HTML。

在DOM和CSSOM准备好以前，浏览器什么都不会显示。

怎么让首屏渲染的更快？

1、预解析

所有的浏览器都会预加载　脚本、外部的CSS来自img标签的图片。

浏览器的并行下载的文件的数量是有限制的。为了更快的渲染页面，浏览器对每个要下载的文件都要设置优先级来优化下载。 在进行预解析的时候，浏览器是不会执行内敛的JavaScript代码块。这就意味着它不会发现任何脚本的注入资源，这些资源会排到抓取队列的最后面。

要想首屏渲染的越快，就越不应该在首屏就加载JS文件

本来DOM和CSSOM是井水不犯河水的，但是由于使用JavaScript后，CSSOM也开始阻塞DOM的构建，只有在CSSOM构建完毕的时候，DOM再恢复。

这是因为JavaScript不仅可以改变DOM，还要改变样式，也就是它可以更改CSSOM。因为不完整的CSSOM是无法使用的，如果JavaScript想访问并更改它，在执行JavaScript的时候，必须要拿到完整的CSSOM的下载和构建。

回流和重绘？

当浏览器生成渲染树以后，会根据渲染数来进行布局。这种叫做回流

这一阶段浏览器要做的事情就是弄清楚各个节点在页面中确切的位置和大小，这一行为也叫做自动重排。

重绘：当我们对DOM的操作导致了样式的变化、却并没有影响其几何属性时，浏览器不需要从新计算元素的几何属性，直接为该元素绘制样式

回流：当我们对DOM的修改引发了DOM的几何尺寸的变化。（宽高，隐藏显示）时，浏览器就会从新计算元素的几何属性，然后再将计算的结果绘制出来。这个过程叫回流（重排）

HTTP缓存？

强缓存与协商缓存

强缓存：不关心资源是否更新，通过两个头部实现

1、expire： Http1.0的内容，expire可以指定文件到期的时间，但是它设置的是一个固定的时间，但是有时候客户端和服务端的时间可能有误差。

2、chache-control: 使用max-age指定资源被缓存多久。　max-age+请求时间

协商缓存：在强制的缓存失效以后，浏览器携带的缓存标识请求服务器，缓存表示生效。

一般通过两个头部实现

1、last-modefied, 是服务器告诉浏览器该资源的最后修改时间，第一次访问的时候在响应的头部添加，浏览器下一次请求资源的时候，检测到之前请求的有这个header就会添加if-modified-since字段，值和上次浏览器给自己的资源的最后修改时间，如果服务器中该资源的最后修改时间没有发生变化，就会返回３０４。如果值变大了就会返回２００和新的资源

2、Etag 是由服务端的特定算法生成的该文件的唯一标识，只要资源有变化就会重新生成ETag, 浏览器下次请求资源时会将之前的ETag值放在if-None-Match字段中发往服务器比较

ａｊａｘ取消缓存

ｓｅｔＲｅｑｕｅｓｔＨｅａｄｅｒ（“if-modefied-since：０）cache-control：ｎｏ－ｃｈａｃｈｅ

HTML5 应用程序缓存和浏览器缓存有什么区别？

应用程序缓存

HTML5 的应用程序缓存即appcache，就是从浏览器的缓存中分出来一部分缓存区，在这个区内保存应用的数据。

要是有描述文件实现，在描述文件内列出要下载和缓存的资源，

要将描述问价与页面关联起来， 在html的manifest文件中指定这个文件的路径。

应用程序缓存是HTML5的重用特征之一，提供了离线使用的功能，让应用程序可以获取本地网站的内容，提高了网页性能。

他的实现借助于 manifest文件。

与传统浏览器缓存相比，它不强制用户访问的网站内容被缓存。

浏览器缓存

浏览器缓存包括cookie、以及localStorage和sessionStorage等。

Cookie 和 localStorage 和SessionStorage的区别？

Cookie是网站为了表示用户身份而存储在用户本地终端的数据，通常在同源的http请求中携带，在浏览器和服务器之间传递。Cookie的数据大小很小不超过4K。Cookie在设置的Cookie过期前一直有效。

local Storage和sessionStorage是H5提供的，可以在浏览器端存储数据、可以避免数据在浏览器和服务器之间来回传递。数据大小大约5M左右。

sessionStorage是在同源同窗口之间一直存在的数据，关闭窗口后数据立马被销毁。

localStorage 用于持久话的本地存储、除非手动删除，否则就会永久存在

ａｊａｘ取消缓存

ｓｅｔＲｅｑｕｅｓｔＨｅａｄｅｒ（“if-modefied-since：０）cache-control：ｎｏ－ｃｈａｃｈｅ

响应报文包括状态行，首部和主题，

状态行由HTTP版本，状态码和状态短语组成

状态码由三位数组成1xx 对请求的通知消息。2xx表示请求成功。 3xx表示把资源重定向到另一个URL。4xx表示客户端出现差错。5xx表示服务器端出 现差错。

Vue

Vue是使用数据劫持加发布者和订阅者模式，当一个普通的JavaScript对象传入vue的实例作为data的实例，Vue就会遍历此对象的所有属性，并且使用Object.defineproperty将这些属性都转换成setter和getter模式。每个组件都有一个watcher就相当于是一个订阅者，他会在组件渲染过程中将接触过的数据记录为依赖，当依赖项的setter触发的时候，就会通知watcher，然后使其触发的组件从新渲染。在检测数据的时候要注意Vue不能检测对象和数组的变化。也就是说对于对象，vue不能检测到property的添加和删除，所以使用的时候第一个使在data中初始化，还有就是使用set 和$set方法对嵌套对象添加响应式属性。

对于数组vue不能识别直接使用数组索引添加元素和使用length属性。所以可以使用Vue.set方法和 vm.items.splice(indexValue, 1, newValue)

Vue 是异步更新队列，Vue在更新DOM时是异步执行的，只要监听到数据变化，vue就将开启一个队列，并缓存在同一事件中发生的所有的数据变更，如果同一个watcher被多次触发，只会被推入到队列中一次，这种缓冲可以去除重复数据避免重复计算和操作DOM。当在下一个事件中，　Vue就会刷新队列执行实际工作。

但是如果想基于更新后的DOM进行某些操作，就使用Ｖｕｅ.nextTick(callback)，这样回调函数在DOM更新完被调用。使用vm.$nextTick()非常方便，因为他不需要全局的vue，并且回调函数中的this就会自动的绑定到vue的实例中。

this.nextTick（callback）实际上类似于Promise.resolve().then(callback), 或者 setTimeout(callback, 0)

1、虚拟DOM

虚拟DOM是为了解决浏览器性能问题而设计出来的，如果一次操作中有好几次有更新DOM的动作，虚拟DOM不会立即操作DOM，而是将每次更新的diff内容保存到本地的一个JS对象中，最终将这个JS对象一次性的挂载到DOM树上，再进行后续的操作，避免很多无畏的计算。

所以使用虚拟DOM的好处是页面的更新可以先映射到虚拟的DOM上，等更新完成后再映射成真正的DOM，交给浏览器绘制。

ｄｉｆｆ算发，比较两个DOM树的差异。深度优先遍历。

Vue的computed和watch的区别

computed是计算结果并返回，只有当要计算的值发生改变的时候才会触发。

计算属性的结果会被缓存，除非依赖的响应式属性变化才会重新计算。

watch是监听某个值，当被监听的值发生变化时，执行对应的操作。

watch更适合于监听某一个值的变化并作出对应的操作，而computed适合计算已有的值并返回结果。

单页面应用：第一次进入页面会请求HTML文件，（当路径发生改变切换到其他的组件的时候，并不会从新请求HTML页面，JS可以感知页面的变化，清除不要的内容，并把需要的内容显示到页面上。

也就是前端路由。

vue-router的实现。vue-router是更新页面但是并不从新请求页面，使用的时候时router-veiw标签，里边有from to以及tag属性。

单页面跳转有history和hash两种模式+。

history。使用的时候配置mode:history。然后使用history.pushState和history.replaceState两种API完成url的跳转但是不发送请求。

hash模式。hash就是url#后面的内容。当hash值变化的时候不会导致浏览器重新向服务器发送请求，也不会刷新页面。当hash值变化的时候还会触发一个hashchange事件，可以通过监听这个事件知道hash发生了哪些变化并实现更新页面的部分操作。

实现：

受限时数据驱动，可以使用router对象来表示路由状态，然后使用vue的数据驱动模型，

首先定义一个observer使用defineproperty来劫持属性的setter和getter方法。然后定义一个watcher，watcher就相当于一个订阅者。当页面变化的时候只需要监听hashchange、popstate事件，做对应的路由转化，触发router-view做对应的渲染。

vue组件参数传递：

父子组件使用$emit 和 props进行传递

非父子组件使用bus总线的方式进行传递。

给vue的原型添加一个bus，vue.prototype.bus = new vue() 然后使用bus.$emit 和 bus.$on()进行传递。

垃圾回收机制：

JavaScript中有垃圾回收机制，垃圾收集器会定期找出哪些不继续使用的变量，然后释放其内存。

全局变量的生命周期直到浏览器卸载页面才会结束。

标记清除：当变量进入环境时，比如在函数中声明一个变量，就将这个变量标记为进入变量。从逻辑上将永远不能释放进入环境的变量所占的内存。当变量离开环境的时候，则将其标记为离开环境。垃圾回收器在运行的时候会给存储在内存上的所有变量都加上标记，然后去掉环境中的变量和被环境中的变量引用的变量的标记（闭包）。以后再被加上标记的变量将会被视为即将被删除的变量。原因是环境中的变量已经无法访问到这些变量了。然后垃圾回收完成内存清除工作，删除哪些带标记的值并收回他们占用的空间。

引用计数

一个变量被声明并将一个引用值付给变量时这个值的引用次数就是1.如果这个值又赋给另一个变量则引用值加一，如果包含这个值的引用变量取得另外的值，则值减1。这样当垃圾回收器再次运行的时候就会释放引用次数为0的值的所占的内存。

循环引用会造成内存泄漏。

em: 文本相对长度单位。相对于当前对象内文本的字体尺寸。如果当前文本的字体没有被设置，则是相对于浏览器的16px的倍数。相对于父元素的字体大小的单位。

rem是相对于根元素的字体大小单位。

http和https？

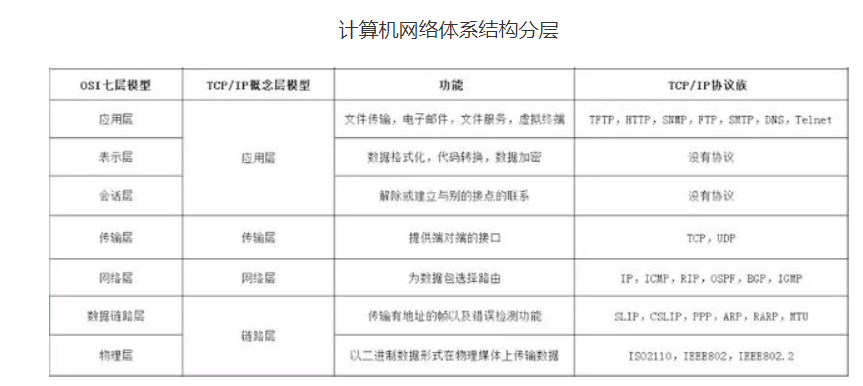
http是超文本传输协议，而https是http和SSL的结合，即使用HTTP协议进行传输，但是在上面加一个SSL。

客户端使用HTTPS 的时候是要先进性加密操作。

首先是浏览器端向服务器端发送请求，建立SSL链接，服务器端收到客户端的请求之后会将网站的安全证书（包含公约）传送给客户端。客户端和服务器端开始协商SSL链接的安全等级，进行加密，然后建立会话，使用公约来加密传输，服务器端使用私约解密会话内容。

HTTPS协议有SSL和HTTP协议组成可以进行加密传输，用户身份认证，要比HTTP安全。

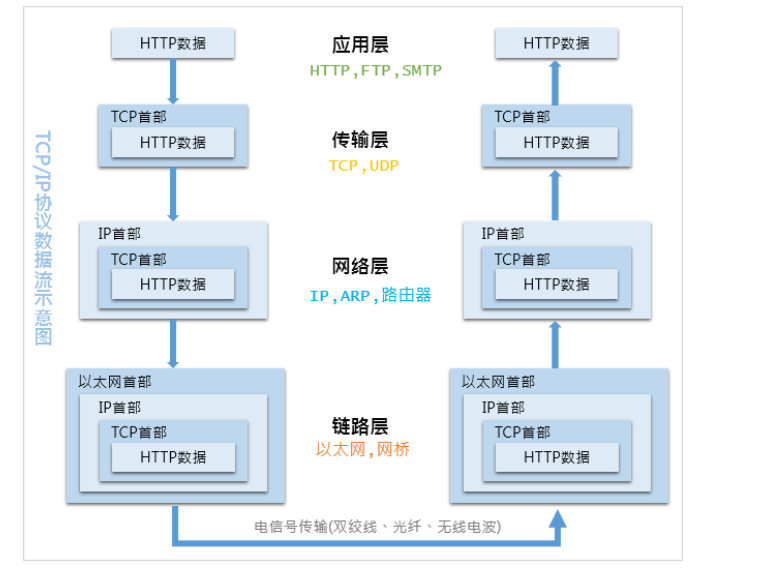
OSI 模型以及其对应的协议



传输层是提供端到端的接口，其中两个代表的协议就是**TCP**和**UDP**

网络层是为数据包选择路由，主要的协议是**IP和ICMP**

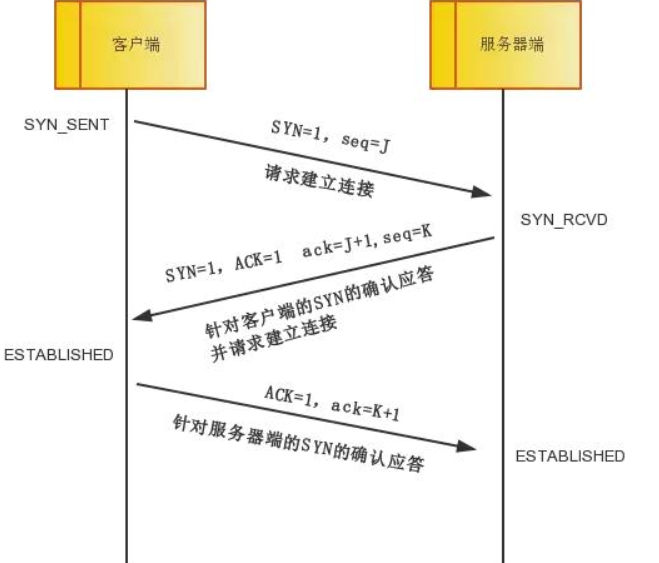
1. 传输过程



1. 传输层中的TCP和UDP
2. TCP 是面向连接的、可靠的**流协议**。TCP 为提供可靠性传输，实行“顺序控制”或“重发控制”机制。此外还具备“流控制（流量控制）”、“拥塞控制”、提高网络利用率等众多功能。（流指不间断的数据结构）
3. UDP 是不具有可靠性的**数据报协议**。UDP尽最大努力交付，即不保证可靠交付。

**TCP的三次握手**

TCP 提供面向有连接的通信传输，三次握手是指建立一个 TCP 连接时需要客户端和服务器端总共发送三个包以确认连接的建立。



除了三次握手还有四次挥手

即A发送关闭请求，B回复关闭请求，B发送关闭请求 A回复关闭请求

1. WebSocket的实现和应用

Websocket是一个应用层协议，它必须依赖 HTTP 协议进行一次握手 ，握手成功后，数据就直接从 TCP 通道传输，与 HTTP 无关了。即：websocket分为握手和数据传输阶段，即进行了HTTP握手 + 双工的TCP连接。

WebSocket是HTML5中的协议，支持持久连续

补充知识：1.http协议的特点是无状态连接。即http的前一次连接与后一次连接是相互独立的 2. http的交互模型是请求/应答模型 HTTP的生命周期通过Request来界定，也就是Request一个Response

1. HTTP的请求方式

HTTP1.0 定义了三种请求方法： GET, POST 和 HEAD方法。

**GET：** 请求指定页面信息，并返回实体主体

**HEAD** 类似于GET请求，只不过返回的响应中没有实体具体内容，用于获取头部

**POST：** 向指定资源提交数据进行处理请求（例如提交表单或者处理文件）数据包含在请求体中。POST请求可能会导致新的资源建立或已有的资源的修改

PUT　从客户端向服务器传送的数据取代指定的文档内容

DELETE　请求服务器删除指定页面

CONNECT　HTTP／１.１中预留给能够将链接改为管道方式的代理服务器

TRACE　回显服务器收到的请求，主要用于测试和诊断

PATCH　是对PUT方法的补充，用于对已知资源进行局部更新

**GET和POST的区别**

get参数通过url传递，post放在request body中。

get请求在url中传递的参数是有长度限制的，而post没有。

get比post更不安全，因为参数直接暴露在url中，所以不能用来传递敏感信息。

get请求只能进行url编码，而post支持多种编码方式

get请求会浏览器主动cache，而post支持多种编码方式。

get请求参数会被完整保留在浏览历史记录里，而post中的参数不会被保留。

GET和POST本质上就是TCP链接，并无差别。但是由于HTTP的规定和浏览器/服务器的限制，导致他们在应用过程中体现出一些不同。

GET产生一个TCP数据包；POST产生两个TCP数据包。

1. 几个很实用的BOM属性对象方法
2. location对象

location对象提供了与当前窗口中加载的文档有关的信息。

location对象即是window对象的属性又是document对象的属性，所以document.location和 window.location引用的是同一个对象

location.assign(‘http://www.baidu.com’)传入一个URL改变浏览器的位置。也可以给location.herf设置对于的URL也可以改变浏览器的位置

location.reload()是从新加载页面

location.search()查询字符串参数，返回？后面的内容

**location.href-**- 返回或设置当前文档的URL

location.search -- 返回URL中的查询字符串部分。例如 http://www.dreamdu.com/dreamdu.php?id=5&name=dreamdu 返回包括(?)后面的内容?id=5&name=dreamdu

location.hash -- 返回URL#后面的内容，如果没有#，返回空

**location.host** -- 返回URL中的域名部分，例如www.dreamdu.com

l**ocation.hostname -**- 返回URL中的主域名部分，例如dreamdu.com

**location.pathname** -- 返回URL的域名后的部分。例如 http://www.dreamdu.com/xhtml/ 返回/xhtml/

**location.port** -- 返回URL中的端口部分。例如 http://www.dreamdu.com:8080/xhtml/ 返回8080

location.protocol -- 返回URL中的协议部分。例如 http://www.dreamdu.com:8080/xhtml/ 返回(//)前面的内容http:

location.assign -- 设置当前文档的URL

location.replace() -- 设置当前文档的URL，并且在history对象的地址列表中移除这个URL location.replace(url);

**location.reload**() -- 重载当前页面

２.．History对象

history对象是一个window对象，每个窗口每个标签页都有自己的history。

history.go() -- 前进或后退指定的页面数 history.go(num);  
history.back() -- 后退一页  
history.forward() -- 前进一页

　　３.　Navigator对象

navigator对象用来识别客户端浏览器，每个浏览器中的navigator对象都有自己的标准

navigator.userAgent -- 返回用户代理头的字符串表示(就是包括浏览器版本信息等的字符串)  
navigator.cookieEnabled -- 返回浏览器是否支持(启用)cookie

七、　HTML5 的 drag API

dragstart：事件主体是被拖放元素，在开始拖放被拖放元素时触发，。

darg：事件主体是被拖放元素，在正在拖放被拖放元素时触发。

dragenter：事件主体是目标元素，在被拖放元素进入某元素时触发。

dragover：事件主体是目标元素，在被拖放在某元素内移动时触发。

dragleave：事件主体是目标元素，在被拖放元素移出目标元素是触发。

drop：事件主体是目标元素，在目标元素完全接受被拖放元素时触发。

dragend：事件主体是被拖放元素，在整个拖放操作结束时触发

八、404 和401 403状态码

(1)404状态码：请求无效

产生原因：

前端提交数据的字段名称和字段类型与后台的实体没有保持一致

前端提交到后台的数据应该是json字符串类型，但是前端没有将obj对象JSON.stringify转化成字符串

解决方法

对照字段的名称，保持一致性

将obj对象通过JSON.stringify实现序列化

（2）401状态码：当前请求需要用户验证

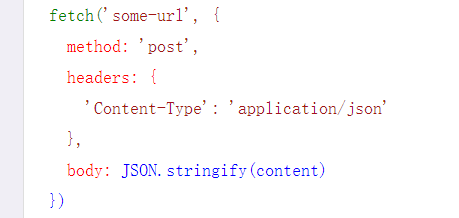
(3)403状态码：服务器已经得到请求，但是拒绝执行

九、 fetch发送2次请求的原因 （腾讯一面）

因为你用fetch的post请求的时候，导致fetch 第一次发送了一个Options请求，询问服务器是否支持修改的请求头，如果服务器支持，则在第二次中发送真正的请求。

fetch(url).then(response)f

fetch只需要发送你想要资源的url，然后使用then来进行你要进行的操作。



1. Cookie、sessionStorage、localStorage的区别 （腾讯一面）

Cookie： 1、.cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递

2、cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下, 存储容量小，只有大约4K左右

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

SessionStorage： 1.仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持

2.不在不同的浏览器窗口中共享，即使是同一个页面

sessionStorage是在同源同窗口之间一直存在的数据，关闭窗口后数据立马被销毁。

localStorage：1.始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据

2.localStorage 在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的

localStorage 用于持久话的本地存储、除非手动删除，否则就会永久存在

**浏览器优化？怎么优化一个页面让其加载速度变快？（阿里腾讯**

减少http的请求次数

从设计层次来说可以简化页面，比如规划好CSS代码，减少CSS文件大小、进来减少CSS的个数，背景整合，对于很小的图片可以使用sprite

样式上可以尽量使用外部的CSS和JS，样式表放在页面的顶部，将脚本放在页面底部，使用link代替＠import。

减少DOM的访问，对于多次需要操作的DOM元素可以设置合理的变量存储内容

3.合理设置适当的缓存。

十九、JS实现跨域（腾讯

同源策略限制一个源加载的文档不能与来自另一个源的资源进行交互。

同源就是：同协议、同域名、同端口

CORS跨域：CORS的思想是让浏览器与服务器进行沟通，也是依赖的XHR对象，实现是浏览器给服务器发送一个请求，添加一个额外的origin头部，其中包括请求页面的源信息，服务器根据这个头部信息决定是否给与响应，如果服务器认可这个请求，就Access-Control-Allow-Origin中回发相同的源信息。

JSONP跨域：JSONP由回调函数和数据两部分组成。回调函数就是当响应到来时在页面上调用的函数，回调函数的名字一般在发送请求中指定。数据就是传入回调函数的JSON数据。

实现过程是动态的创建一个script标签，通过标签的src属性发送一个请求，包括请求的地址以及回调函数的名字。

var script = document.createElement(“script”);

script.src = ‘http://ddddd.net/?callback=handleResponse’

document.body.insertBefore(script,document.body.firstChild)

function handleResponse(response){JSSON数据}

JSONP的缺点是，JSONP是直接从其他域加载代码，如果其他域不安全，就可能有一些恶意的代码。

如何跨域：

1. JSONP

动态创建script标签，通过标签的src属性发送请求，网页通过添加一个<script>元素，向服务器请求JSON数据，其不受同源政策限制，服务器收到请求后，将数据放在一个指定名字的回调函数里传回来。

服务端受到请求后，将数据放置到指定的回调函数返回

浏览器收到后因为是Script标签所以浏览器会当作是脚本进行运行，从而达到跨域获取数据的目的

JSONP 由两部分组成：回调函数和数据。回调函数是当响应到来时应该在页面中调用的函数。回调函数的名字一般是在请求中指定的。而数据就是传入回调函数中的 JSON 数据。

回调函数可以是：‘

function handleResponse(response){

// 对response数据进行操作代码

}

（（（（使用jQuery封装的$.ajax

对于经常用jQuery的开发者来说，能注意到jQuery封装的$.ajax中有一个dataType属性，如果将该属性设置成dataType:"jsonp"，就能实现JSONP跨域了



1. CORS

CORS 是一个 W3C 标准，全称是"跨域资源共享"（Cross-origin resource sharing）它允许浏览器向跨源服务器，发出 XMLHttpRequest 请求，从而克服了 ajax 只能同源使用的限制。

CORS和ａｊａｘ在服务器的请求一样，但是在服务端需要加上服务端设置Access-Control-Allow-Origin

３、location.hash + iframe跨域：a欲与b跨域相互通信，通过中间页c来实现。 三个页面，不同域之间利用iframe的location.hash传值，相同域之间直接js访问来通信。

４、window.name + iframe跨域：通过iframe的src属性由外域转向本地域，跨域数据即由iframe的window.name从外域传递到本地域。

十九、HTTP　２.０　的特性（腾讯一面）

http2.0的特性如下：

1、内容安全，应为http2.0是基于https的，天然具有安全特性，通过http2.0的特性可以避免单纯使用https的性能下降

2、二进制格式，http1.X的解析是基于文本的，http2.0将所有的传输信息分割为更小的消息和帧，并对他们采用二进制格式编码，基于二进制可以让协议有更多的扩展性，比如引入了帧来传输数据和指令

3、多路复用，

HTTP2采用二进制格式传输，取代了HTTP1.x二进制格式解析更高效。 多路复用代替了HTTP1.x的序列和阻塞机制，所有的相同域名请求都通过同一个TCP连接并发完成。在HTTP1.x中，并发多个请求需要多个TCP连接，浏览器为了控制资源会有6-8个TCP连接都限制。 HTTP2中同域名下所有通信都在单个连接上完成，消除了因多个 TCP 连接而带来的延时和内存消耗。单个连接上可以并行交错的请求和响应，之间互不干扰。

在http1.1 中 默认允许 connect： keep-alive 但是在一个TCP里面 数据通信时按次进行，也就是说第二次请求发送要在第一次响应后进行，若第一次响应慢，则要一直阻塞。这个问题就是对头阻塞。 在HTTP/2.0中，在一个TCP链接中，客户端和服务器可以同时发送多个请求和响应，则避免了对头阻塞， 实现了 双向 实时 多工。 在2.0中 采用了数据流，对同一个请求或响应的所有数据包做了一个独一无二的标识，所以可以不用等待发送。两端会根据标识组装数据流。

十一、 iframe是什么？有什么缺点

定义：iframe元素会创建包含另一个文档的内联框架

提示：可以将提示文字放在<iframe></iframe>之间，来提示某些不支持iframe的浏览器

缺点：

1 会阻塞主页面的onload事件

2 搜索引擎无法解读这种页面，不利于SEO

3 iframe和主页面共享连接池，而浏览器对相同区域有限制所以会影响性能。

十二、 Doctype作用?严格模式与混杂模式如何区分？它们有何意义?

Doctype声明于文档最前面，告诉浏览器以何种方式来渲染页面，这里有两种模式，严格模式和混杂模式。

严格模式的排版和JS 运作模式是 **以该浏览器支持的最高标准运行**。

混杂模式，向后兼容，模拟老式浏览器，防止浏览器无法兼容页面。

十三、Cookie如何防范XSS（跨站脚本攻击）攻击

XSS是指攻击者在返回的HTML中嵌入JavaScript脚本

阻止XSS攻击的方式就是在http头部配置set-cookie 其中http only这个属性会可以防止XSS，因为他会禁止JavaScript脚本来访问cookie

secure - 这个属性告诉浏览器仅在请求为https的时候发送cookie。

CSRF对于某个没有授权的系统有权访问某个资源的情况称为CSRF。

未被授权的系统会伪装自己，让处理请求的服务器认为它是合法的。

解决方式：　使用SSL链接来访问可以通过HXR请求的资源。

每次请求都附带经过相应算法计算得到的验证码。

CSRF （跨站请求伪造） 伪造用户身份，以用户的名义发送恶意的请求，冒充用户在站内的正常操作。

（黑客之所以可以伪造用户的请求是因为用户的所有的验证信息都是存在cookie中的，因此黑客可以在不知道这些验证信息的情况下直接利用用户的cookie通过验证。要抵御CSRF就是要在请求中放一个黑客无法伪造的信息，该信息不存在cookie中）可以使用请求地址中随机的token，在服务端建立一个拦截器来验证这个token（可以用户登陆放在session中），如果请求中没有token或者token内容不正确，则认为是CSRF攻击/或者也可以验HTTP协议头部中的referer字段。

十五、Click在iOS上有300ms的延迟残生的原因以及解决方式

1. 原因

用户一次点击屏幕之后，浏览器并不能立刻判断用户是要进行双击缩放，还是想要进行单击操作。因此，iOS Safari 就等待 300 毫秒，以判断用户是否再次点击了屏幕。于是，300 毫秒延迟就这么诞生了。

1. 解决使用fastclick FastClick

在检测到 touchend 事件的时候，会通过 [DOM 自定义事件](https://developer.mozilla.org/en-US/docs/Web/API/CustomEvent)立即触发一个模拟click 事件的click事件（自定义事件），并把浏览器在 300 毫秒之后真正触发的 click 事件阻止掉。

window.addEventListener( "load", function() {

    FastClick.attach( document.body );

}, false );

1. 粗暴型，禁用缩放

<meta name="viewport" content="width=device-width, user-scalable=no">

十六、介绍知道的http返回的状态码

100    Continue    继续。客户端应继续其请求

101    Switching Protocols    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    OK    请求成功。一般用于GET与POST请求**

**201    Created    已创建。成功请求并创建了新的资源**

**202    Accepted    已接受。已经接受请求，但未处理完成**

**203    Non-Authoritative Information    非授权信息。**请求成功。但返回的meta信息不在原始的服务器，而是一个副本

**204    No Content    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

205    Reset Content    重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域

206    Partial Content    部分内容。服务器成功处理了部分GET请求

300     Choices    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    Moved Permanently    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    Found    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    See Other    查看其它地址。与301类似。使用GET和POST请求查看**

**304    Not Modified    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    Use Proxy    使用代理。所请求的资源必须通过代理访问

306    Unused    已经被废弃的HTTP状态码

307    Temporary Redirect    临时重定向。与302类似。使用GET请求重定向

400    Bad Request    客户端请求的语法错误，服务器无法理解

401    Unauthorized    请求要求用户的身份认证

402    Payment Required    保留，将来使用

403    Forbidden    服务器理解请求客户端的请求，但是拒绝执行此请求

**404    Not Found    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    Method Not Allowed    客户端请求中的方法被禁止

406    Not Acceptable    服务器无法根据客户端请求的内容特性完成请求

407    Proxy Authentication Required    请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权

408    Request Time-out    服务器等待客户端发送的请求时间过长，超时

409    Conflict    服务器完成客户端的PUT请求是可能返回此代码，服务器处理请求时发生了冲突

410    Gone    客户端请求的资源已经不存在。410不同于404，如果资源以前有现在被永久删除了可使用410代码，网站设计人员可通过301代码指定资源的新位置

411    Length Required    服务器无法处理客户端发送的不带Content-Length的请求信息

412    Precondition Failed    客户端请求信息的先决条件错误

413    Request Entity Too Large    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    Request-URI Too Large    请求的URI过长（URI通常为网址），服务器无法处理

415    Unsupported Media Type    服务器无法处理请求附带的媒体格式

416    Requested range not satisfiable    客户端请求的范围无效

417    Expectation Failed    服务器无法满足Expect的请求头信息

**500    Internal Server Error    服务器内部错误，无法完成请求**

501    Not Implemented    服务器不支持请求的功能，无法完成请求

502    Bad Gateway    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    Service Unavailable    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    Gateway Time-out    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    HTTP Version not supported    服务器不支持请求的HTTP协议的版本，无法完成处理

十七、HTML５新增的标签有哪些

首先html5为了更好的实践web语义化，增加了header，footer，nav, aside, section等语义化标签，

在表单方面，为了增强表单，为input增加了color，emial,　data ,　range等类型，

在存储方面，提供了sessionStorage，localStorage,和离线存储，通过这些存储方式方便数据在客户端的存储和获取，

在多媒体方面规定了音频和视频元素audio和vedio，另外还有地理定位，canvas画布，拖放，多线程编程的web worker和websocket协议。

十八、在地址栏中输入ｕｒｌ到页面呈现出来会发生什么

DNS解析

TCP连接

发送HTTP请求

服务器处理请求并返回HTTP报文

浏览器解析渲染页面

连接结束

DNS解析：　输入url后，首先需要找到这个url域名的服务器ip,为了寻找这个ip，浏览器首先会寻找缓存，查看缓存中是否有记录，缓存的查找记录为：浏览器缓存-》系统缓存-》路由器缓存，缓存中没有则查找系统的hosts文件中是否有记录，如果没有则查询DNS服务器，

TCP链接和发送HTTP请求：　得到服务器的ip地址后，浏览器根据这个ip以及相应的端口号，构造一个http请求，这个请求报文会包括这次请求的信息，主要是请求方法，请求说明和请求附带的数据，并将这个http请求封装在一个tcp包中，这个tcp包会依次经过传输层，网络层，数据链路层，物理层到达服务器，

服务器处理：　服务器解析这个请求来作出响应，返回相应的html给浏览器，

浏览器解析渲染页面：　建立一个DOM树和CSSOM树。 因为html是一个树形结构，浏览器根据这个html来构建DOM树，在dom树的构建过程中如果遇到JS脚本和外部JS连接，则会停止构建DOM树来执行和下载相应的代码，这会造成阻塞，这就是为什么推荐JS代码应该放在html代码的后面，

之后根据外部样式，内部样式，内联样式构建一个CSS对象模型树CSSOM树，构建完成后和DOM树合并为渲染树，这里主要做的是排除非视觉节点，比如script，meta标签和排除display为none的节点，之后进行布局，布局主要是确定各个元素的位置和尺寸，之后是渲染页面，因为html文件中会含有图片，视频，音频等资源，在解析DOM的过程中，遇到这些都会进行并行下载，浏览器对每个域的并行下载数量有一定的限制，一般是4-6个，当然在这些所有的请求中我们还需要关注的就是缓存，缓存一般通过Cache-Control、Last-Modify、Expires等首部字段控制。 Cache-Control和Expires的区别在于Cache-Control使用相对时间，Expires使用的是基于服务器 端的绝对时间，因为存在时差问题，一般采用Cache-Control，在请求这些有设置了缓存的数据时，会先 查看是否过期，如果没有过期则直接使用本地缓存，过期则请求并在服务器校验文件是否修改，如果上一次 响应设置了ETag值会在这次请求的时候作为If-None-Match的值交给服务器校验，如果一致，继续校验 Last-Modified，没有设置ETag则直接验证Last-Modified，再决定是否返回304

(1) 布局视口（layout viewport）

布局视口定义了pc网页在移动端的默认布局行为，布局视口默认为980px。也就是说在不设置网页的viewport的情况下，pc端的网页默认会以布局视口为基准，在移动端进行展示。

(2) 视觉视口（visual viewport）

视觉视口表示浏览器内看到的网站的显示区域，用户可以通过缩放来查看网页的显示内容，从而改变视觉视口。initial-scale的设置就是对visual viewport的设置

(3) 理想视口（ideal viewport）

理想视口是在移动设备中就是指设备的分辨率。换句话说，理想视口或者说分辨率就是给定设备物理像素的情况下，最佳的“布局视口”。

<meta name="viewport" content="width=device-width,initial-scale=1.0,maximum-scale=1.0,minimum-scale=1.0,user-scalable=no">

通常pc端的布局视口是980px，移动端iphone6是375，现在有一个750px\*1134px的视觉稿，那么有：

pc:1 css 像素 = 750/980 = 0.76px;   iphpne6:1css = 750/375=2px

@media媒体查询可以针对不同的媒体类型定义不同的样式，尤其是响应式页面，可以针对不同屏幕的大小编写多套样式，从而达到自适用的效果。

使用百分比单位%来实现响应式的效果。

1、百分比的具体分析

子元素的height和width使用的百分比是相对于子元素的直接父元素；

子元素的top和bottom使用的百分比是相对于直接非static定位父元素的height;

子元素的left和right使用的百分比是相对于直接非static定位父元素的width;

子元素的padding和margin使用的百分水平方向与垂直方向都是相对于直接父元素的width;

CSS篇

一、 CSS的盒模型

CSS的盒子模型是指所有的HTML元素都可以看作一个盒子。盒模型的本质是一个盒子，封装周围的HTML元素，包括margin border padding contend-width

W3C标准的盒模型只是包括内容的宽度

但是IE的盒模型确实包括了内容内边距和边框的宽度的总和

使用box-sizing可以指定盒子的模型

其中content-box是指只包括内容的，表示标准的盒子模型

border-box指包括width+padding+border

二、 移动端的1px边框的问题

采用meta viewport方式

设置为理想视口

<meta name="viewport" content="initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />

1. 采用@media监听devicePixelRatio设备分辨率，然后根据分别率对边框进行适当的缩放，如果分辨率是2则设置0.5px
2. 通过伪元素加transform：scall() 的方式

原先元素的 border 设置为none 然后进行相对定位{border:none; position:relative}，然后利用 :before 或者 :after 重做 border ，根据 transform 的 scale进行合适的缩放，新做的 border 绝对定位。

Border-1px($color)

position:relative

border:none

&:after

Display:block position:absolute left:0 bottom:0 width: 100%

Border-top:1px solid $color content:”

@media (-webkit-min-device-pixel-ratio:1.5) ,(min-device-pixel-ratio:1.5)”

.border-1px

&::after

-weblit-transform:scaleY(.7)transform:scaleY(0.7)

同样device-pixel-ratio为2的时候缩放0.5

使用viewport + rem 实现

三 link 和 import标签的区别

1. Link是html的标签，页面加载的时候link会同时被加载，link因为是html标签所以没有兼容性的问题，此外link的优先级高于@import
2. @import是CSS提供的，@import引用的CSS会在页面加载结束后加载 @import只有IE5 以上才能识别

三 transition和 animation的区别

Transition是一个过渡效果，animation是一个动画效果，所以两者都是随时间改变元素的属性的值，transition有from to所以是两帧animation可以一帧一帧播放。Transition需要触发一个事件来调动其过度的效果但是animation不需要 动画的实际表现是由 @keyframes规则实现

四、Flex布局

Flex布局顾名思义是弹性布局，任何一个元素都可以flex布局设置display： flex即可 使用flex布局以后float clear 以及vertical-align都会失效

容器属性

1. flex-deriction : row(默认值)起点左端/row-reverse（起点右端）|| column（垂直方向）起点上/column-reverse起点下
2. Flex-wrap： nowrap（默认）不换行/wrap换行第一行在上/wrap-reverse换行第一行在下
3. Flex-flow是flex-deriction和flex-wrap的简写默认 row nowrap
4. Justify-content项目在主轴上的对其方式

Justify-content：flex-start（默认）左对齐/flex-end右对齐 center剧中/space-bettew两端对齐，项目之间间隔相等/space-around 项目两侧间隔相等，所以项目之间的比边框大一倍

1. Align-item 定义在交叉轴的对齐方式 Align-item：flex-start交叉轴的起点/flex-end交叉轴终点对齐/center中点对齐/base-line项目的第一行文字的基线对齐/ stretch（默认值）如果未设置高度或者auto则填满整个容器
2. Align-conten定义多根轴线的对齐方式同align-item类似

项目的属性

1. Order定义项目的排列顺序从小到大
2. Flex-grow 定义项目的放大比例默认为0，即有剩余空间也不放大
3. Flex-shrink定义项目的缩小比例，默认为1 即如果空间不足则项目缩小，若有一个设置为0则该项目不缩小
4. Flex-basis 默认为 auto计算主轴是否有多余的空间

Flex的默认是 0 1 auto 然后有auto：1 1 auto 可放大缩小none： 0 0 auto 不放大不缩小

1. Align-self的是允许单个项目与其他的项目的不同的对齐方式

五、垂直居中的方法(图片在一个容器中居中)

1. 文字的垂直居中设置height=line-height

2. 盒子的垂直居中是设置margin： 0 auto

五、图片在容器中居中

没有脱离文档流

1. 使用flex布局 即给其父元素添加 display：flex justify-content：center align-items：center

2. 父元素设置 display：table-cell vertical-align:middel可以实现图片的垂直居中然后再加上text-align：center就能实现图片完全居中

脱离标准流的时候

3. 父元素设置position：relative

子元素设置position：absolute top: 0 left: 0 right:0 bottom: 0 margin: auto

4. 设置父元素相对定位，子元素设置 top: 50% left: 50% margin-top: -1/2height margin-left: -1/2width

六、块级元素和行内元素

行内元素：span a b i em

与其他行内元素并排，不能设置宽高默认的宽度是文字的宽度

块级元素：div h li dt p

独占一行不予其他元素并列不设置宽度的话默认是父元素的宽度

七、多行文本一行显示出省略号

1. overflow： hidden

2.white-space：nowrap

3.text-overflow：eclipse

八、visibility = hidden 和 opacity： 0 以及 display ：none的区别

1 opacity=0，该元素隐藏起来了，但不会改变页面布局，并且，如果该元素已经绑定一些事件，如click事件，那么点击该区域，也能触发点击事件的 会被子元素继承

2 visibility=hidden，该元素隐藏起来了，但不会改变页面布局，但是不会触发该元素已经绑定的事件 会被子元素继承

3 display=none，把元素隐藏起来，不占据空间，并且会改变页面布局 --不会被子元素继承

九、双边距重叠问题

多个相邻（兄弟或者父子关系）普通流的块元素垂直方向marigin会重叠

折叠的结果为：

两个相邻的外边距都是正数时，折叠结果是它们两者之间较大的值。

两个相邻的外边距都是负数时，折叠结果是两者绝对值的较大值。

两个外边距一正一负时，折叠结果是两者的相加的和。

十、position属性

固定定位fixed：

元素的位置相对于浏览器窗口是固定位置，即使窗口是滚动的它也不会移动。Fixed定位使元素的位置与文档流无关，因此不占据空间。 Fixed定位的元素和其他元素重叠。

相对定位relative：

如果对一个元素进行相对定位，它将出现在它所在的位置上。然后，可以通过设置垂直或水平位置，让这个元素“相对于”它的起点进行移动。 在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。

绝对定位absolute：

绝对定位的元素的位置相对于最近的已定位父元素，如果元素没有已定位的父元素，那么它的位置相对于<html>。 absolute 定位使元素的位置与文档流无关，因此不占据空间。 absolute 定位的元素和其他元素重叠。

粘性定位sticky：

元素先按照普通文档流定位，然后相对于该元素在流中的flow root（BFC）和 containing block（最近的块级祖先元素）定位。而后，元素定位表现为在跨越特定阈值前为相对定位，之后为固定定位。

默认定位Static：

默认值。没有定位，元素出现在正常的流中（忽略top, bottom, left, right 或者 z-index 声明）。

inherit:

规定应该从父元素继承position 属性的值。

十一、清除浮动

1. 在浮动的元素后边加一个空元素，比如div 然后设置clear: both
2. 给浮动的元素加上overflow：hidden 或者 overflow：auto这样浮动的元素又回到了容器层，把高度撑起来了
3. 使用伪元素的方法

给浮动的元素加上一个clearfix类

.clearfix

Display:inline-block

&:after

Display:block

Height:0 line-height:0 content:’’clear:both visibility:hidden

Float position： absolute fixed 元素变成inline-block

十二、三栏布局的方式

三栏布局

如果是三个div宽度已经知道，则可以使用定位去做。

父元素：position:relative

左边 position: absolute;left:0; width: n px;

中间 position: absoluet left: npx; width: npx;

右边 position: absolute right: 0; width: npx;

如果是三个div宽度已经知道，则可以使用浮动去做。

左边 float: left; width: n px;

中间 float:left; width: npx;

右边 float: left; width: npx;

但是这样会带来一个问题，如果三个盒子的高度等于父盒子的高度，那么后边的元素会上移，所以为了保证文档不出现问题，最后的元素最好是清除浮动。

（如果是三栏布局的时候，如果左边左浮动右边右浮动，中间元素不浮动的时候是就会把第三个元素挤下去，所以中间元素设置margin什么的是不对的）

如果三个div的宽度已经知道了，可以使用table-cell做

左边 display:table-cell; width: n px;

中间 display:table-cell; width: npx;

右边 display:table-cell; width: npx;

三栏的宽度不知道可以使用flex布局去做

父元素： display：flex；

左右中：flex：1

十三、重绘和重排

重排： DOM的变化影响浏览器预算几何属性，浏览器需要重新计算元素几何属性，其他元素的几何属性也会受到影响，浏览器需要从新生产渲染书，这个过程称为重排

重绘：浏览器将受到影响的部分重新渲染到屏幕上的过程叫重绘

造成的原因是：

添加或删除可见的DOM元素

元素的尺寸位置发生变化

浏览器页面初始化

减少的重绘和重排的方法有：

不在布局信息改变的时候做DOM的查询

使用csstext和class Name一次性改变属性

对于多次重排的元素比如动画使用绝对定位脱离标准文档流，使其不影响其他元素

十四、CSS布局

六种布局方式：圣杯布局、双飞翼布局、Flex布局、绝对定位布局、表格布局、网格布局。

圣杯布局是指布局从上到下分为header、container、footer，然后container部分定为三栏布局。这种布局方式同样分为header、container、footer。圣杯布局的缺陷在于 center 是在 container 的padding中的，因此宽度小的时候会出现混乱。

双飞翼布局给center 部分包裹了一个 main 通过设置margin主动地把页面撑开。

Flex布局是由CSS3提供的一种方便的布局方式。

绝对定位布局是给container 设置position: relative和overflow: hidden，因为绝对定位的元素的参照物为第一个postion不为static的祖先元素。 left 向左浮动，right 向右浮动。center 使用绝对定位，通过设置left和right并把两边撑开。 center 设置top: 0和bottom: 0使其高度撑开。

表格布局的好处是能使三栏的高度统一。

网格布局可能是最强大的布局方式了，使用起来极其方便，但目前而言，兼容性并不好。网格布局，可以将页面分割成多个区域，或者用来定义内部元素的大小，位置，图层关系。

十五、sticky footer 布局

常见的网页布局一般分为header content footer 部分。

当页面的头部和内容的部分内容比较少的时候，页脚区不是随着内容区排布而是始终显示在屏幕的最下方。当内容区的内容较多的时，页脚能随着文档流撑开始终显示在页面的最下方。

代码主要分为 三部分

<div class=”wrapper clearfix”>

<div class=”title”></div>

<div class=”main”></div>

<div class=” footer”></div>

</div>

.wrapper{ min-height: 100% width: 100%}

.main{margin-top:64px padding-bottom:64px底部的文字的高度}

.footer{ margin: -64px auto 0 auto}

.clearfix::after{

display: block content: ‘’ height:0 line-height:0 clear:both visibility: hidden

}

JavaScript篇

一、get和post请求

Http协议从来没有规定过post和get请求的长度限制，Get显示最大长度是因为浏览器和web服务器限制了URL的长度。且不同的浏览器和服务器限制的最大长度不一样，IE浏览器的最大长度为2083byte Chrome最大时8182byte

Get请求是向服务器请求指定的内容，并返回请求的实体部分，类似于查找的过程，用户获取数据，可以不用每次都与数据库连接，因此可以使用缓存

Post不同，post一般是修改和删除的工作，所以必须与数据库交互，不能使用缓存。

get 是从服务器上获取数据，post是向服务器传送数据

get是将数据放在URL中，在URL中可以看到数据， Post是将表单各个字段放到Header内，用户看不到

get传送的数据量小安全性低，post传送的数据量大安全性高

二、各种高度clientHeight，scrollHeight，offsetHeight

ClientHeight 指元素。容器的高度，不包括border和滚动条的高度。clientHeight与height值差不多，如果没有边框和滚动条的话，两者值相等，都是指容器的高度。火狐与IE下的值是一样的。

scrollHeight 指元素实际内容的高度，而不是容器的高度。当元素内容的高度大于容器高度时，指元素内容的高度。

offsetHeight 指元素容器的高度，如果有设置boder和滚动条的话，加上边框和滚动条的高度， 样式的height+上下padding+上下border

clientTop，scrollTop，offsetTop

ClientTop 容器内部相对于容器本身的top偏移，实际上是border-width即上边框border的厚度

ScrollTop 元素中垂直滚动条滚动的距离。向上卷出去的距离

OffsetTop 元素距离父元素顶端的距离， 子元素的外边框到offsetParent内边框的距离

**clientX:** 相对文档的水平坐标

**clientY:** 相对文档的垂直坐标

**offsetX:** 相对容器的水平坐标

**offsetY:** 相对容器的垂直坐标

三 JavaScript继承与原型链

JavaScript只有一种结构： 对象

每一个实例对象都有一个私有属性（\_proto\_）指向他的构造函数的原型对象(prototype)，该原型对象也有一个自己的原型对象（\_proto\_）,层层向上直到一个对象的原型为null。Null没有原型，且是原型链的最后一个环节。

使用不同的方法来创建对象和生成原型链

1. 使用语法结构创建对象

Var o ={a:1}其原型链为o-🡪object.prototype-->null

Var a =[“hhh”]其原型链为 a 🡪 Array.prototype🡪Objext.prototype🡪 null

Function f(){return 2} 其原型链为 f🡪Function.prototype🡪Object.prototype🡪null

2、 使用构造器创建对象

在JavaScript中，构造器其实就是一个普通的函数。当使用new操作符来操作这个函数的时候，它就被称为构造方法（构造函数）

Function Graph（）{this.edge=[]}这就创建了一个Graph的构造函数

var g =new Graph()-----创建了g对象，他有属性edge g🡪Graph.prototype

* 1. 使用 Object.create()来创建对象 ES5

Var a= {a:1}

Var b = Object.create(a)

b🡪a🡪Object.prototype🡪null

这样console。Log（b.a）=1 继承而而来的属性

* 1. 使用class关键字来创建对象

ES6 引入了一套新的关键字来实现class。这写新的关键字包括 class, constructor static extends super

四、闭包（阿里和腾讯

闭包可以从内部函数访问外部函数的作用域

闭包是指有权访问另外一个函数作用域中的变量的函数。

闭包就是函数的局部变量集合，只是这些局部变量在函数返回后会继续存在。闭包就是就是函数的“堆栈”在函数返回后并不释放，我们也可以理解为这些函数堆栈并不在栈上分配而是在堆上分配。当在一个函数内定义另外一个函数就会产生闭包。

闭包就是函数定义的方法或者回调函数在外部调用的时候仍然可以访问父函数的变量

JavaScript的函数可以形成闭包。闭包是由函数以及创建该函数的语法环境组合而成。 这个环境包含了创建这个闭包创建时所能访问的所有局部变量。

通常使用只有一个方法的地方都可以使用闭包，返回的函数可以被接受，我们成返回的函数为回调函数

闭包同样可以访问私有函数和变量，闭包有它自己独立的语法环境

五、Javascript的继承

怎样获取原型

1、p.constructor.prototype

2、Object.getPrototypeOf(p)

JavaScript的继承不通过类class(ES6)，而是通过“原型对象（prototype）”实现。在JS中常见的继承

在JS中是使用构造函数来创建一个对象的，每一个构造函数都一个prototype指针也是一个对象，也是该构造函数的原型对象，该原型对象包含了由该构造函数所有实例所共享的属性和方法。当使用该构造函数初始化一个对象的时候，该对象的内部就会有一个指针（\_\_proto\_\_），指向构造函数原型对象.

当我们访问一个对象的属性的时候，如果这个对象中不包含这个属性就会查找其指向的原型对象中查找这个属性，然后这个原型对象也会有自己的原型，就这样一级一级查找，就形成了一个原型链，也是实现继承的一种方式。

1.原型链

子类的原型\_\_proto\_\_指向父类的实例以实现继承。将父类的私有共有方法和属性都当作子类的公共有属性。缺点是当有多个实例的时候，都是指向同一个父类的实例，数据共享，一个子类操作属性就会影响另一个子类。

其实就是将子类的实例指向父类的实例。

而且在子类中添加方法或者重写父类的时候要放到替换原型的语句之后

Student.prototype = new Person()

1. 构造函数继承

就是在子类的构造函数中通过call()调用父类型构造函数。它只是子类的实例，只能继承父类实例的属性和方法，不能继承原型属性和方法

Function Student (args) {

Person.call(this, args)

}

1. 组合继承

在子类构造函数中通过call()调用父类构造器，继承父类的属性并保留传参的优点，然后再将父类的实例作为子类的原型，实现函数复用

组合继承需要修复构造函数的构造器的指向的 prototype.constructor=()

这种的缺点是需要两次调用构造函数。第一次是在创建子类原型的时候，另一次是在子类构造函数的内部

Function Student (args) {

Person.call(this, args)

}

Student.prototype = new Person()

4 组合继承的优化。

为了不两次调用构造函数，最好的方法是让子类原型（prototype）和父类的原型（prototype）指向同一个对象。这样子类继承父类的公有方法作为自己的共有方法

为了区分实例时子类的实例还是父类的实例，使用Object.creat()方法，然后再修复子类的构造函数，将子类的构造函数赋值给子类的原型的构造函数

Function Student (args) {

Person.call(this, args)

}

Student.prototype = Object.create(Person.prototype)//核心代码

Student.prototype.constructor = Student//核心代码

（其中Person 是f父类构造函数 Student是子类构造函数）

4、ES6的class继承

Class 定义的类可以使用extends关键字实现继承。而且可以使用static关键字定义静态的方法。

再使用的时候需要使用super()方法将父类的实例对象和方法加到this上去

比如 class Students extends Person {

Constructor(name, age, …arg){

super(name, age)//其中name和age是父类的属性，通过super调用了父类的构造方法和属性

this.arg = arg //(子类自己的属性)

}

}

**什么是 DOM 和 BOM？**

```

DOM 指的是文档对象模型，它指的是把文档当做一个对象来对待，这个对象主要定义了处理网页内容的方法和接口。

BOM 指的是浏览器对象模型，它指的是把浏览器当做一个对象来对待，这个对象主要定义了与浏览器进行交互的法和接口。BOM

的核心是 window，而 window 对象具有双重角色，它既是通过 js 访问浏览器窗口的一个接口，又是一个 Global（全局）

对象。这意味着在网页中定义的任何对象，变量和函数，都作为全局对象的一个属性或者方法存在。window 对象含有 locati

on 对象、navigator 对象、screen 对象等子对象，并且 DOM 的最根本的对象 document 对象也是 BOM 的 window 对

象的子对象。

五、前端中的事件流

事件流就是用户触发事件后，事件的传递。DOM中的事件包括事件捕获阶段、处于目标阶段、事件冒泡阶段。其中事件从内到外是事件的冒泡阶段，事件从外到内是事件的捕获阶段

其中addEventlistener(要处理事件类型，事件处理函数，布尔值)

其中 true是捕获阶段，false 是冒泡阶段

事件冒泡：多个元素有嵌套有层次关系，都注册相同的事件，如果里面元素触发了，外面元素的事件就自动触发了。 阻止冒泡： window.event.cancelBubble = true // e.stopPropagation

1. 事件委托

事件委托就是不在是事件的发生地上设置监听函数，而是在其父元素上设置监听函数，通过事件冒泡，父元素可以监听到子元素的事件触发，然后可以通过判断不同的DOM类型做出不同的响应。

八、图片的懒加载和预加载

预加载： 提前加载图片，当用户需要查看的时候直接从本地渲染

懒加载： 延迟加载图片或者当符合一定的条件的时候才加载图片，就是在图片的src路径下放一个1px\*1px的图片路径，在data-src 存放真正的需要加载的图片路径当满足一定的条件的时候js将data-src的值赋值给src

九、mouseover和mouseenter的区别

Mouseover 当鼠标移入元素和子元素都会触发事件，所以有一个触发过程，冒泡的过程。对应的事件是mouseout

Mouseenter是鼠标移入元素自身才会触发事件，也就是不会冒泡，对应的移除事件是mouseleave

十、apply 和 call bind 的区别

JavaScript中函数是对象，可以改变其上下文，函数中的this的指向也是可以变化的。可以使用call apply bind 方法修改上下文，函数中的this指针被转换为 call apply的第一个参数

Apply、call中的第一个参数如果传入的是null 那么调用该方法的函数对象的指向就是window， call和apply的区别就是call内的参数是arg1,,,argn的方式但是apply的是传入的数组。

Bind是将传入的对象赋值一份，返回一个新的函数，所以使用的时候需要调用

十一、如何解决回调地狱

1. 拆分function 将各步拆解为单个的function
2. 通过Promise链式调用的方式 (Promise.resolve()方法会将这个对象转为Promise对象, 然后立即执行该对象的then方法,如果是没有任何参数或则没有then方法则返回一个状态为resolve的Promise对象)
3. 使用generator暂停执行效果方式 （function\* 可将函数变为generator。在异步任务处使用yield关键字， 此时generator就会将程序的执行权交给其他代码，在异步任务完成后调用next方法来恢复yield下方代码的执行）
4. 使用async/await的方法, 使用async创建一个async函数，await只能用在这个函数内

Async 的用法如下

async function foo () {} const foo = async function() {}

当执行到await的时候相当于函数挂起等待 await后的Promise的返回再执行后面的语句

十二、异步加载JS的方法

一般把script的标签放在尾端是因为，渲染引擎当遇到JS脚本的时候就会停下来执行脚本，等执行完脚本才会继续向下渲染，如果JS中有输出document的内容、修改DOM、重定向等行为，就会造成页面阻塞

异步加载又称非阻塞加载，就是在浏览器加载JS脚本的同时还会继续渲染页面。

常用的异步加载JS的办法有：

defer和async的不同是他们开始执行脚本的时机不同。defer比async要先引入，他的执行在解析完全完成之后再开始，它处在DOMContentLoaded事件之前。它保证脚本会按照它在HTML中出现的顺序执行，并且不会阻塞解析。

async脚本在他们完成下载以后就立即执行，它处在window的load事件之前，这也意味着设置了async的脚本可能不会按照他们在HTML中出现的顺序执行。设置了async的脚本通常是在其他的脚本加载以后才加载，而不阻碍DOM的构建，但是如果一个ａｓｙｎｃ的脚本很快完成了下载，那么他的立即执行阻碍DOM构建以及在之后才能下载的同步脚本。

对于页面很重要的资源，可以使用preload来告诉浏览器要尽快的加载他们

<link **rel**="preload" **href**="very\_important.js" **as**="script">

几乎可以链接到任何东西上。ａｓ属性告诉浏览器要下载的是什么。

1. Script 标签的 async 属性<script type=””,src=””async=”async”>

Async属性是HTML5新增的属性，Chrome、FireFox 、IE9+浏览器都支持，但是该方法只支持外部的JS脚本。但是该方法不能保证脚本按顺序执行。

1. Script标签的defer属性 <script type=”text/javascript”defer=”defer”>

该方法兼容所有的浏览器，可以保证所有设置了defer属性的脚本按顺序执行

1. ES6模块的type=“module”属性。浏览器对于type=”module”属性的浏览器都异步加载，不会造成阻塞浏览器，等到整个页面渲染完，再执行模块脚本。<script type=”module” src=””>
2. 创建script，插入到DOM中，加载完的时候回调可以使用load事件监听什么时候js文件异步下载完毕。

十三、JS拖拽功能的实现

JS的托拽需要使用三个DOM事件，

onmousedown :鼠标按下事件 onmousemove 鼠标移动事件 onmouseup: 鼠标抬起事件

拖拽的原理就是根据鼠标的移动来移动被拖动的元素。

元素的position属性一定是relative和absolute。移动其实是元素绝对定位的top 和 left的改变

当鼠标点击被拖拽的元素时触发onmousedown事件，然后设置元素可以拖拽，可以使用clientX和clientY来获取鼠标的x,y坐标。

然后鼠标移动触发onmousemove的事件 放开鼠标触发onmouseup的事件

元素的坐标 =鼠标移动的横向距离+元素本来的坐标=鼠标之前的坐标-鼠标现在的坐标+元素本来的坐标

HTML5提供了拖放属性

主要分为一下几步

1. 设置元素可以拖放 draggable=”true”
2. 拖动什么 1）ondragstart属性调用一个函数，该函数规定被拖动的内容

2）DataTransfer.setData(‘Text/类型’，’可拖动的元素的id’)该方法设置被拖动的数据的数据类型和值

1. 放到何处 ondragover 属性规定元素被放到何处，但是需要使用e.preventDefault来阻止元素默认处理方式。
2. 进行放置，当放置被拖动数据的时候，ondrop属性调用事件

十四、ajax解决浏览器缓存的问题

Ajax能够提交页面的载入速度主要是ajax能实现局部刷新，通过局部刷新减少了重复数据的载入。也就是说再载入数据的同时将数据放在内存中，只要没有刷新页面这些数据就会一直被缓存在内存中，当提交的URL与历史的URL一致的时候就不会提交给服务器，得到的还是之前缓存在浏览器中的数据。为了保证我们获得数据是最新的还是要禁止浏览器缓存功能

1. 在发生ajax的请求上加上anyAjaxObj.setRequestHeader(“if-Modified-Since”，“0”) (If-Modefied-Since:0是让缓存过期)
2. 在ajax发送请求前加上anyAjaxObj.setRequestHeader(“Cache-Control”,”no-cache”) （Cache-Control：no-cache）禁用缓存
3. 在URL后面加上时间戳 ‘nowtime=’+new Date().getTime() (强行让每次的请求地址不同)
4. 在URL后面加上一个随机数：“fresh=”+Math.random()(强行让每次的请求地址不同)
5. 如果是使用的jQuery，这可以$ajaxSetup({cache:false})(不设置Ajax缓存)

十五、JS中的垃圾回收机制

Javascript创建的字符串、数组、对象等，解释器都必须分配内存来储存那个实体，只要是动态的分配了内存，最终都要释放这些内存以方便可以能够再备用，否则JavaScript的解析器会消耗完全部的内存造成系统崩溃。

垃圾回收的方法：标记清除、计数引用

标记清除：当变量引入环境时、就标记这个变量为进入环境、从逻辑上讲永远不能释放进入环境的变量所占的内存，当离开环境的时候就标记为离开环境。

垃圾回收器在运行的时候会给存储在内存中的变量都加上标记，然后去掉环境变量中的变量以及被环境变量所引用的变量，删除所有被标记的变量（删除的变量无法在环境变量中被访问所以被删除）最后垃圾回收完成内存清除工作，回收他们所占的内存。

十六、前端模块化

将一个复杂的程序按照一定的规则封装成几个块，块的内部数据与实现是私有的，只是向外部暴漏一些接口与其他模块通信。模块化以后就可以减少命名空间污染更好的分离代码便于维护和复用。

模块化规范： CommonJS

CommonJS是同步加载模块，主要是通过module.exports来导出要暴漏的接口，使用require方法来同步加载所依赖的其他模块。（浏览器中不适合使用浏览器环境中使用，因为浏览器同步加载就有可能造成加载阻塞）

AMD（异步加载模块）模块的加载不会影响后面语句的运行，所有依赖模块的语句都定义在一个回调函数中，等到加载完成之后回调函数才执行

requireJS实现了AMD规范，主要用于解决下述两个问题。

1. 多个文件有依赖关系，被依赖的文件需要早于依赖它的文件加载到浏览器
2. 加载的时候浏览器会停止页面渲染，加载文件越多，页面失去响应的时间越长。
3. 使用define定义暴漏的模块 define(‘module’,function(){})
4. Require定义加载模块 require([‘module’],function(){})
5. Module是一个数组，里面的成员就是要加载的模块，第二个参数是callback是加载成功之后的回调函数。

CMD规范

CMD规范专门用于浏览器端，模块的加载是异步的，模块使用时才会加载执行，Sea.js中，所有的JavaScript模块都准寻CMD模块定义规范。

Define(function({exports.暴漏对象=暴漏对象}))

define(function(require){require.async(‘引入的模块’,function(m3) )})

十七、将原生的Ajax封装成一个Ajax对象

Function myAjax(url){

Return new promise(function(resolve,reject){

Var Xhr = new XMLHttpRequest();

Xhr.open(‘get’,url);

Xhr.send(data)

Xhr.onreadystatechange=function(){

If(Xhr.status==200&&readyState==4){

var json = JSON.parseInt(Xhr.responseText);

resolve(json)

}else if(Xhr.readyState==4&&xhr.status!==200){

Reject(‘error’)

}

}

})

}

十八、setTimeout、setInterval和requestAnimationFrame之间的区别

二十、数组去重

1、双层for循环加splice去重

function unique(arr) {

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

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

if(arr[i]===arr[j]){

arr.splice(j,1);

j--;

}

}

}

return arr;

}

这种方法NaN 和 {} 没法去重

2、 使用indexOf()去重

function unique (arr) {

if(!Array.isArray(arr)){

console.log(‘type error’)

return

}

var array = []

arr.forEach((value) => {

if(array.indexOf(value)===-1){ //(arr.indexOf(value)===index)

//！array.includes(value)

// 如果新素组中没有value 或者 原来数组中value第一次出现的地方等于index //如果新数组中不包含value则在新数组中加入value

//includes（value,index）方法的使用同indexOf类似 indexOf是查找数组中value的值并返回第一次的索引。includes是查找有没有该元素，返回true或者false

array.push(value)

}

})

return array

}

二十一、JS数组迭代方法

1、forEach()为每个数组元素调用一次回调函数

Array.forEach(function(value,index,arr){})最常用的方法，可以代替for循环

2、map()该方法会通过调用（传入的也是一个回调函数）每个素组的元素执行函数来创建新的数组，不会更改原数组，会返回一个新的数组

map(function(value,index,arr){})

3、filter()方法创建一个包含通过测试的数组元素的新数组

filter(function(value,index,arr){})

4、 reduce()方法在每个数组元素上运行函数，以生成（减少）单个值

reduce(function(total,value,index,arr){})

1. every()方法检测所有数组的值是否通过测试，全都通过则返回true否则返回false every(function(value,index,arr){})
2. some()方法检测数组中某些值是不是通过了测试。有一个通过测试就返回true否则返回false some(function(value, index, arr){})
3. indexOf()在数组中搜索元素的值并返回其位置，如果没找到就返回-1

Array.indexOf(value,start) value 要检索的值， start开始的位置

1. find()方法返回通过测试函数的第一个元素的值

Array.find(function(reduce, index, arr){})

10． findIndex()方法返回通过测试函数的第一个数组元素的索引

Array.findIndex(function(index, value, ))

二十二、深拷贝和浅拷贝

深拷贝与浅拷贝，假设B复制了A，当修改A时，判断B是否会发生变化，如果B也跟着变了，则说明是浅拷贝，否则是深拷贝。深拷贝是基于比较复杂的object类型的数据，基本类型的数据—名值存储在栈内存中，而引用类型的数据—名存在栈内存中，值存在堆内存中，同时栈内存中会提供一个引用的地址指向堆内存中的值。浅拷贝拷贝的只是栈内存中的指向堆内存中的引用地址，而并非堆内存中的值，故两个引用地址指向同一个堆内存中的值，一个变量修改了堆内存中的值就会影响另一个变量。

数组的浅拷贝

如果是数组，可以使用一些数组方法比如slice和concat返回一个新的数组的特性实现拷贝，但是concat不能拷贝数组中嵌套的对象或者方法，这种复制引用的方式叫做浅拷贝

深拷贝是指完全拷贝一个对象即使嵌套了对象或者方法两者也相互分离，即使改变了一个也不会影响另一个

适用于数组和对象的拷贝方式是JSON.parse(JSON.stringify(arr))

jQuery的extend方法

$.extend([deep], target, object1[, objectN]);

该方法可以实现深复制

二十三、箭头函数

1. 箭头函数没有this，所以需要通过查找作用域链来确定this的值
2. 箭头函数没有自己的arguments对象，但是可以访问外围函数的arguments对象。使用reset代替 。。。
3. 不能用new关键字调用、同样也没有new.target值和原型

二十三、JavaScript 的对象类型

基本对象类型、主要有：number、string、undefined、Boolean、null、symbol

引用类型、当复制保存着某个对象的变量的时候，操作的是对象的引用，但在为对象添加属性的时候，操作的是实际对象。

引用类型有：Object、Array、RegExp、Data、Function以及单体内置对象（Math、Global）特殊基本包装类型（String、Number、Boolean）

二十四、new的操作原理

1. 创建一个类的实例、创建一个空对象，然后将这个空对象的\_\_proto\_\_设置为构造函数的prototype
2. 初始化实例：构造函数被传入参数被调用、关键字this被设定指向该实例obj
3. 返回实例obj

二十五、bootsharp清除浮动的方法

.clearfix:before,.clearfix:after{display:table;content:””}

.clearfix:after{clear:both} .clearfix{zoom:1}

Ajax

一、Ajax返回的状态

0 .未初始化还没调用send()方法

1. 载入。已经调用send（）方法，正在发送请求
2. 载入完成。已经得到全部响应内容
3. 正在解析相应
4. 响应解析完成

二、原生ajax

var xhr = new XMLHttpRequest()

xhr.open(‘get’,’’)

xhr.send()

xhr.onreadystateChange = function(){

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

console.log(xhr.responseText)

}

}

1、匿名函数的应用场景

匿名函数可以用来封装局部作用域内的代码，以使其声明的变量不会暴漏到全局作用域。

匿名函数可以作为只用一次但不在其他地方使用的回调函数。当处理函数在调用他们的程序的内部被定义的时候，代码具有更好的自闭性和可读性。