基础知识点

1. 从URL在浏览器被输入到页面展现的过程中发生了什么？
2. 键盘或触屏输入URL并回车确认
3. 浏览器先查看浏览器缓存-系统缓存-路由器缓存，如果缓存中有，会直接显示页面内容。没有则跳到第3步操作。
4. 在发送http请求前，需要域名解析（DNS解析），解析获取与URL对应的IP地址
5. 浏览器向服务器发起tcp连接，与浏览器建立tcp三次握手
6. 握手成功后，浏览器向服务器发送http请求，请求数据包
7. 服务器处理收到的请求，将数据返回至浏览器
8. 浏览器收到HTTP响应
9. 浏览器读取页面内容，渲染，解析html源码
10. 生成dom树，解析css样式，js交互，客户端与服务器交互，ajax查询数据

https://www.jianshu.com/p/d616d887953a

1. Git常用指令

下载项目：git clone url[SSH/HTTP]

切换分支：git checkout newBranch

git pull

git add –a

git commit –m “注释”

git push

git merge master  
在自己的分支上打包：gulp mp

创建新分支：git checkout –b dev dev是新创建的分支名称

切换到某个分支上：git checkout master 切换到master分支上

删除某个分支：git branch –d dev 先切换到master主分支上，再删除dev这个分支

提交新创建的分支：git push origin HEAD –u 在当前新创建的分支下提交这个分支

创建新分支以后：

git push --set-upstream origin newBranch

git push origin newBranch

shift+: 点击q 退出

**怎样将自己的分支合并到主分支master上去？**

先切换到master

再拉取最新的master信息

回到自己的分支

再执行git merge master

看是否冲突，解决冲突

在自己的分支上打包：gulp mp

提交更改

查看本地分支：git branch –a

查看GIT日志：git log

查看文件状态：git status

**Git创建新分支并提交到github：**

1. 查看当前所在的分支：git branch
2. 创建本地分支并切换到新创建的分支上去：git checkout –b dev
3. 把新创建的分支信息推送到github：git push origin newBranch
4. 把本地新分支和远程的分支关联：git push --set-upstream origin newBranch

**git rebase：**将分叉的分支重新合并。git merge和git rebase作用相同，都是将一个分支上的代码合并到另一个分支上，只是实现方式不同。

1. 基于VUE的UI框架：element-ui 、 Ant Design Vue 、View UI
2. 闭包

https://www.cnblogs.com/anthong0325/p/9297401.html

函数内的变量在函数外被引用，就形成了一个闭包

闭包最大的2个用处：①读取函数内部的变量 ②让这些变量的值始终保持在内存中，不被污染 ③封装对象的私有属性和私有方法

Javascript中的GC机制：

在Javascript中，如果一个对象不再被引用，那么这个对象就会被GC回收，否则这个对象一直会保存在内存中。

垃圾回收机制---GC

Javascript具有自动垃圾回收机制（GC：Garbage Collection），也就是说，执行环境会负责管理代码执行过程中使用的内存。原理：垃圾收集器会定期（周期性）找出那些不再继续使用的变量，然后释放其内存。找出不再使用的变量，然后释放掉其占用的内存，但是这个过程不是实时的，因为其开销比较大，所以垃圾回收期会按照固定的时间间隔周期性执行。

1. 本地存储

cookie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。

**cookie：**在浏览器和服务器间来回传递；cookie数据不能超过4K；因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如会话标识。**数据有效期：**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。**作用域：**在所有同源窗口中都是共享的。

**sessionStorage：**存储数据可以达到5M或者更大；**数据有效期：**始终有效，窗口或者浏览器关闭前有效。**作用域：**不在不同的浏览器窗口中共享，即使是同一个页面。

**localStorage：**存储数据可以达到5M或者更大；**数据有效期：**始终有效，窗口或者浏览器关闭也一直保存。作用域：在所有同源窗口中共享。

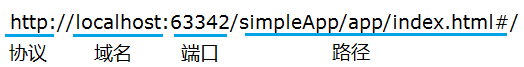
共同点：都保存在浏览器中，且都是同源的

不同点：cookie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。

1. 跨域的原理和方法、浏览器的同源策略、JSONP

域名、协议、端口不同，导致跨域

URL由协议、域名、端口、路径组成。如果2个URL的协议、域名、端口都相同，则表示它们同源。相反，只要协议、域名、端口有任何一个不同，就被当做跨域。



同源策略：浏览器采用同源策略，禁止页面加载或执行与自身来源不同的域的任何脚本。换句话说，浏览器禁止的是来自不同源的“document”或脚本，对当前“document”读取或设置某些属性。

情景：比如一个恶意网站的页面通过iframe嵌入了银行的登录页面（二者不同源），如果没有同源限制，恶意网页上的javascript脚本就可以在用户登录银行的时候获取用户名和密码。

浏览器中有哪些不受同源限制：<script>、<img>、<iframe>、<link>这些包含src属性的标签可以加载跨域资源。但浏览器限制了JavaScript的权限使其不能读、写加载的内容。

跨域：指一个域的网页去请求另一个域的资源。比如从http://www.baidu.com/页面去请求http://www.google.com的资源

跨域技术-JSONP：

JSONP是什么？

上面提到的包含src属性的<script>标签可以加载跨域资源。JSONP就是利用<script>标签的跨域能力实现跨域数据的访问。

解决跨域问题常见的方案有2个：

1. JSONP：利用script标签可跨域的特点，在跨域脚本中可以直接回调当前脚本的函数
2. CORS：服务器设置HTTP响应头中Access-Control-Allow-Origin值，解除跨域限制

vue中解决跨域问题：

方法1：后台更改header

header(‘Access-Control-Allow-Origin:\*’); //允许所有来源访问

header(‘Access-Control-Allow-Method:POST,GET’) //允许访问的方式

方法2：打开config/index.js，在proxyTable中填写以下代码

1. 父子组件同时加载时的生命周期怎样？
2. 先加载父组件的beforeCreate、created、beforeMount
3. 再加载子组件的beforeCreate、created、beforeMount、mounted
4. 最后加载父组件的mounted
5. http状态码

200 请求响应成功

301 重定向

400 请求错误

403 服务器接收到请求，但拒绝返回数据

404 请求资源不存在

500 服务器错误

1. 前端性能优化

提高网页响应速度🡪减少请求数、减少资源大小、找最快的服务器

①减少请求数：使用浏览器缓存、减少重定向请求、减少dns的查询次数

②减少资源大小：压缩图片、开启网络压缩（vue的默认压缩和gzip压缩，需要服务器设置）、压缩js和css代码

③找最快服务器：进行cdn加速

项目上线前，你做过哪些性能优化：

* 图片预加载，CSS样式表放在顶部且link链式引入，Javascript放在底部body结束标签前
* 减少http请求次数，如防抖和节流
* API接口数据设置缓存，CSS Sprites/SVG Sprites，JS、CSS源码压缩、图片大小控制合适，使用iconfont或SVG它们比图片更小更清晰，网页Gzip压缩
* 减少DOM操作次数，优化javascript性能
* 减少DOM元素数量，合理使用:after :before等伪类
* 避免重定向，图片懒加载，前后端分离开发，资源按需加载，最好能做到首屏直出（即：服务端渲染）
* 避免使用CSS Expression（CSS表达式）
* 多域名分发划分内容到不同域名，解决浏览器域名请求并发数问题，同时也解决了请求默认携带的cookie问题
* 尽量减少iframe使用，它会阻塞主页面的渲染
* 对所有资源压缩：JS、CSS、字体、图片等，甚至html

1. CSS的优化

* 避免嵌套太多的选择器
* 减少css嵌套，最好不要嵌套3层以上
* 建立公共样式，把相同样式提取出来作为公共类使用，比如常用的清除浮动等
* 拆分出公共css文件，把大部分页面的公共结构的样式提取出来放到单独css文件里
* CSS Sprite / SVG Sprite
* 减少重排，避免不必要的重绘
* 简化css，借助webpack、gulp等进行代码压缩丑化
* 尽量不要使用过多的内联样式
* 异步加载css文件
* 首推的是合并css文件，如果页面加载10个css文件，每个文件1K，那么也要比只加载一个100K的文件慢
* 不要使用@import引入css文件，用link
* less、sass抽取公共部分写成函数

1. @mixin/@extend/@include/ scss语法

https://www.jianshu.com/p/a99764ff3c41

<style rel="stylesheet/scss" lang="scss" scoped>

CSS原生的@import会通过额外的HTTP请求获取引入的样式片段，而SASS的@import则会直接将这些引入的片段合并到CSS文件中并且不会产生新的HTTP请求。

通过$符号去声明一个变量

SCSS使用&关键字在CSS规则中引用父级选择器

**混合—mixin**

@mixin border-radius($radius){

border-radius: $radius;

-ms-border-radius: $radius;

-moz-border-radius: $radius;

-webkit-border-radius: $radius;

}

.box{

@include border-radius(10px);

}

上面的代码建立了一个名为border-radius的mixin，并传递了一个变量$radius作为参数，然后在后续代码中通过@include border-radius(10px)使用该mixin，最终编译结果如下：

.box{

border-radius: 10px;

-ms-border-radius: 10px;

-moz-border-radius: 10px;

-webkit-border-radius: 10px;

}

**继承--@extend**

通过@extend指令在选择器之间复用css属性，并且不会产生冗余的代码

// 这段代码不会被输出到最终生成的CSS文件，因为它没有被任何代码所继承。

%other-styles {

display: flex;

flex-wrap: wrap;

}

// 下面代码会正常输出到生成的CSS文件，因为它被其接下来的代码所继承。

%message-common {

border: 1px solid #ccc;

padding: 10px;

color: #333;

}

.message {

@extend %message-common;

}

.success {

@extend %message-common;

border-color: green;

}

SASS提供了标准的算术运算符，如：+ - \* / %

1. 750px----缩放视口---viewport

在head头部的meta标签中，设置initial-scale、maximum-scale、user-scale为该倍数

获取屏幕宽度，除以750px，得到倍数，以这个倍数来缩放视口

(function (pw) {

var d = document;

var a = ((d.documentElement && d.documentElement.clientWidth) || (screen && screen.width)) / pw;

var vp = document.querySelectorAll("meta[name='viewport']")[0];

if (!vp) {

vp = d.createElement('meta');

d.head.appendChild(vp);

}

vp.setAttribute('name', 'viewport');

vp.setAttribute('content', 'width=device-width,initial-scale=' + a + ',minimum-scale=' + a + ',maximum-scale=' + a + ',user-scalable=no');

})(750);

1. 你对重绘、重排的理解

* 首先网页首次渲染生成时，这个可称为重排
* 修改DOM、样式表、用户事件或行为（鼠标悬停、页面滚动、输入框键入文字、改变窗口大小等等），这些都会导致页面重新渲染，那么重新渲染，就需要重新生成布局和重新绘制节点，前者叫做“重排”，后者“重绘”
* 减少或集中对页面的操作，即多次操作集中在一起执行
* 总之可以简单总结为：重绘不一定重排，但重排必然会重绘

1. Object常用API

掘金网：https://juejin.im/post/5c9257aef265da60e86e0c7b

1. Object.assign()
2. Object.create()
3. Object.defineProperty(obj,prop,descriptor)
4. Object.getOwnPropertyDescriptor(obj,prop)
5. Object.getOwnPropertyDescriptors(obj)
6. Object.getOwnPropertyNames(obj)
7. Object.keys(obj)
8. Object.getPrototypeOf(obj)
9. Object.is(val1,val2)
10. Object.isExtensible(obj)
11. Object.seal(obj)
12. Object.isSealed(obj)
13. Object.freeze(obj)
14. Object.isFrozen(obj)
15. Object.getOwnPropertySymbols(obj)
16. Object.entries(obj)
17. Object.values(obj)
18. CSS命名---BEM命名规范

https://juejin.im/post/5b925e616fb9a05cdd2ce70d

Bem是块（block）、元素（element）、修饰符（modifer）的简写，由Yandex团队提出的一种前端css命名方法论。

BEM是一个简单又非常有用的命名约定。让你的前端代码更容易阅读和理解，更容易协作，更容易控制，更加健壮和明确，而且更加严密。

BEM命名约定的模式是：

.block{}

.block\_element{}

.block-modifier{}

BEM的一个槽点是，命名方式长而难看，书写不雅。相比BEM格式带来的遍历来说，我们应客观看待。而且，一般来说会使用通过less/sass等预处理语言来编写css，利用其语言特性书写起来要简单得多。

1. call/apply/bind

b.call(a,b1,b2…,b3) 将this的指向改为a，b1、b2是传递的参数

b.apply(a,[b1,b2,…],c1,c2,…) 将this的指向改为a，第2个参数必须为数组

var c=b.bind(a) 返回一个函数，其它和call一样

1. function Child()继承function Parent()

方法一：function Child(){Parent.apply(this,arguments)}

方法二：Child.prototype=new Parent(); Child.prototype.constructor=Child;

1. 非构造函数的继承

http://www.ruanyifeng.com/blog/2010/05/object-oriented\_javascript\_inheritance\_continued.html

var Chinese={nation:’中国’};

var Doctor={career:’医生’};

怎样让“医生”去继承“中国人”

function object(o){

function F(){}

F.prototype=o;

return new F();

}

var Doctor=object(Chinese);

1. prototype和\_\_proto\_\_的区别

Javascript中所有的对象都是Object的实例，并继承Object.prototype的属性和方法。

在对象创建时，就会有一些预定义的属性，其中定义函数的时候，这个预定义属性就是prototype，这个prototype是一个普通的对象。

而定义普通的对象的时候，就会产生一个\_ \_proto\_ \_，这个\_ \_proto\_ \_指向的是这个对象的构造函数的prototype

1. 判断是否为数组的方法
2. var arr=[1]; var bool=arr instanceof Array; //array instanceof Array
3. if(arr.constructor===Array)
4. var bool=Array.isArray(arr) //Array.isArray(array)
5. 数组排序的方法

https://yuzmb.github.io/2016/11/03/笔试经常出现的js数组排序与去重算法/

var arr=[1,9,4,50,49,6,3,2];

function sortNumber(a,b){return a-b;}

function test(){return arr.sort(sortNumber)}

1. 数组去重的方法
2. Array.from(new Set(arr))
3. […new Set(arr)]
4. 用indexOf或者includes循环
5. 垂直居中的方式

设置父级元素：.parent{display:flex;justify-content:center;align-items:center;} 或者display:grid;

justify-content:center; /\* 水平居中 \*/

align-items:center; /\* 垂直居中 \*/

1. slice / splice

slice() 从某个已有的数组返回选定的元素

arrayObject.slice(start,end) 返回一个新数组，包含从start到end（不包括该元素）的arrayObject中的元素

start，必需，规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置，也就是说，-1指最后一个元素，-2指倒数第二个元素，以此类推

end，可选，规定从何处结束选取。该参数是数组片段结束处的数组下标。如果没有指定该参数，那么切分的数组包含从start到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。

splice() 删除元素，并向数组添加新元素

arrayObject.splice(index,howmany,item1,...,itemX);

index 必需。整数，规定添加/删除项目的位置，使用负数可从数组结尾处规定位置

howmany 必需。要删除的项目数量。如果设置为0，则不会删除项目

item1,...,itemX 可选。向数组添加的新项目

注意：splice()方法和slice()方法的作用是不同的，splice()方法会直接对数组进行修改

1. arr.includes(searchElement, fromIndex)

searchElement 必须。需要查找的元素值。

fromIndex 可选。从该索引处开始查找 searchElement。如果为负值，则按升序从 array.length + fromIndex 的索引开始搜索。默认为 0。

var bool2=arr1.includes(1,-1);

console.log(bool2);

// 从第4位开始搜索，到第5位

1. JS中的DOM与BOM

Javascript组成：

1. ECMAScript基本语法
2. BOM（浏览器对象模型）
3. DOM（文档对象模型）

一）BOM（browser Object Model）

浏览器对象模型：使用对象描述了浏览器的各个部分的内容

1. window：当前的窗口

window常用的方法：

open() 打开一个新的资源

moveTo() 将窗口左上角的屏幕位置移动到指定的x和y位置

moveBy() 相对于当前的窗口移动指定的x和y偏移值（左上角）

setInterval() 每隔指定的毫秒数执行代码（无限次）

setTimeout() 经过指定毫秒数执行一次代码（一次）

open()方法接收4个参数，分别是：打开资源名、打开方式、打开的窗口大小位置、资源找不到是否用其它代替

window.open(“image.html”,”\_blank”,”height=500pt,width=350pt,top=50pt,left=450pt”,false);

1. 地址栏对象

常用的方法有：

href：设置或获取整个URL为字符串

reload()：重新加载页面地址

location.href=”http://www.baidu.com”; 设置地址栏地址

location.reload(); 重新加载地址栏地址

1. 屏幕对象

screen（屏幕对象）：获取电脑的屏幕的一些数据

常用方法：

availHeight 获取系统屏幕的工作区域高度，排除Microsoft® Windows® 任务栏(可用的屏幕高度)

availWidth 获取系统屏幕的工作区域宽度，排除Windows任务栏（可用的屏幕宽度）

height width 表示的是显示器的宽和高，也就是电脑的分辨率

二）DOM编程（Document Object Model）

文档对象模型：当一个html页面加载到浏览器的时候，那么浏览器会为每个标签都创建一个对应的对象描述该标签的所有信息，那么我们看到的网页信息实际上就是看到了这些标签对象的信息，如果我们需要操作页面的数据，那么我们就可以通过这些标签对象进行操作。

document.getElementById(“id1”);

document.getElementsByTagName(“div”);

document.getElementsByName(“Button”);

var trNode=document.createElement(“tr”); //创建tr标签节点

1. css有哪些实现布局的方式

* table布局

display:table; =>相当于“table”标签

display:table-row; =>相当于”tr”标签

display:table-cell; =>相当于”td”标签

* flex布局

如果需要两边定宽，中间自适应，可以这样写:

<div class=”container”>

<div class=”left”></div>

<div class=”middle”></div>

<div class=”right”></div>

</div>

.left{width:20px;}

.middle{flex:1};

.right{width:20px;}

* float布局

解决父元素高度塌陷的方法：

父元素设置：overflow:auto; 或者 overflow:hidden;

给父元素加一个after伪类：

.container::after{

content:””;

clear:both;

display:block;

visibility:hidden;

height:0;

}

* 响应式布局

1. meta标签

最简单的处理方式是加上一个meta标签

<meta name=”viewport” content=”width=device-width,initial-scale=1”>

其中，width=device-width这一句的意思是让页面的宽度等于屏幕的宽度

1. 使用rem

rem指的是html元素的font-size，html元素的font-size默认是16px，所以1rem=16px；然后根据rem来计算各个元素的宽高

1. media query

css2允许用户根据特定的media类型定制样式，基本语法如下：

@media screen and (max-width:360px){

html{font-size:12px;}

}

意思就是：当屏幕的宽度小于等于360px的时候html的字体大小为12px。

width:calc(100% - 10px);

height:calc(100% - 10px);

* append在元素内部的尾部加上元素
* prepend在元素内部的前部加上元素
* after在元素外部的尾部加上元素
* before在元素外部的头部加上元素
* html用当前内容替换掉原内容

1. v-if和v-show的区别

v-show用于频繁判断显示和隐藏

1. 普通函数和箭头函数的区别
2. AMD和CMD的区别

AMD推崇依赖前置，在定义模块的时候就要声明其依赖的模块。即：预加载所有需要的依赖。

CMD推崇就近依赖，只有在用到某个模块的时候再去require。

所以：AMD用户体验好，因为没有延迟，依赖模块提前执行了；CMD性能好，因为只有用户需要的时候才执行。

1. HTTP缓存

Web缓存大致分为：数据库缓存、服务器端缓存（代理服务器缓存、CDN缓存）、浏览器缓存等。

1. VUE虚拟DOM的原理

虚拟DOM就是放在JS和HTML中间的一个层。它可以通过对比新旧DOM，来获取差异，然后只把差异部分真正渲染到页面上，从而减少实际DOM操作，提升网页性能。

1. vue传参的方式有哪些
2. vue组件化开发
3. webpack

模式（mode）：开发、测试或生产环境的切换

1. vue中$nextTick的实现原理