1.行元素块元素与行块元素

行级元素：水平分布多个占一行 不可以设置宽高，对margin仅设置左右方向有效，上下无效；padding设置上下左右都有效，即会撑大空间,行内元素尺寸由内含的内容决定，不会自动进行换行

a span i strong em

块级元素：垂直分布 独占一行 可以设置宽高 margin和padding的上下左右均对其有效 可以自动换行 默认排列方式从上到下

div p ul li h1-h6 table

行块元素：不自动换行 可以设置宽高 默认排列方式从左到右

display:inline 转换为行内元素

display:block 转换为块元素

display:inline-block 转换为行内块元素

2.盒子模型

标准盒模型：范围包括margin padding border content，并且content部分不包含其他部分

怪异盒模型：范围包括margin padding border content，与标准盒模型不同的是：怪异盒模型的content部分包含了border和padding

当box-sizing:content-box时，为标准盒模型 内容width=content

当box-sizing:border-box时，为怪异盒模型 内容width=content+border+padding

3. css选择器优先级

!important>行内样式>ID选择器>类选择器>元素>通配符(\* $)>继承>浏览器默认属性

4. 元素的垂直水平居中

(1)方法一：父元素设置 position:relative

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

(2)方法二：calc 计算函数 + 相对定位

父元素设置 position:relative

子元素设置 position:relative; top:calc(50%-子元素高度的一半); left:calc(50%-子元素宽度的一半)

(3)方法三：transform

父元素设置 position:relative

子元素设置 position:absolute; top:50%; left:50%; transform:translate(-50%,-50%)

(4)方法四：flex布局

父元素设置 display:flex; justify-content:center; align-item:center;

5.两边固定中间自适应

(1)方法一：float流体布局，左右两边盒子设置宽度，和左浮动和右浮动，中间盒子设置左右margin，宽度自适应，布局是center放在right之后

(2)方法二：绝对定位布局，左中右两边absolute布局，左边盒子设置绝对定位，left=0,右边盒子设置绝对定位，right=0,中间盒子设置绝对布局，宽度为100%,left为左边盒子的宽度,right为右边盒子的宽度

(3)方法三：flex布局 三个盒子外层包裹一个父盒子，center盒子放中间，父元素设置display:flex; 左右盒子设置固定宽度，中间盒子设置 flex:1

(4)方法四：圣杯布局

先给三个盒子的父级盒子设置padding-left和padding-right，使左右两边留有左右盒子宽度的空白区域，再讲三个盒子同时设置左浮动float:left，并设置中间盒子宽度为100%，这样中间盒子独占一行显示，接着设置左右盒子的margin-left，让三个盒子并排显示，会遮住中间盒子左右两边的一部分内容，最后左右盒子使用相对定位position:relative，并分别配合right和left属性，将盒子定位到左右两边留白的位置。



(5)方法五：双飞翼布局



中间盒子外包裹一个div，接着让三个盒子左浮动，中间盒子宽度设置100%，这样中间盒子会独占一行，然后左右盒子设置自己的固定宽度，在设置左右盒子的margin-left，让三个盒子同一行显示，左盒子的margin-left值和center外层包裹盒子的宽度相等，由于中间盒子包裹了一层div，并设置了左右的margin值，与左右盒子宽度一致，这样中间盒子就不会被遮挡，就不用再使用定位解决，中间盒子center放在最前面.

6. 块级格式化（BFC）的规范理解

块级格式化上下文是一块独立的渲染区域，该区域拥有一套渲染规则来约束块级盒子的布局，且与区域外部无关

如何创建BFC：

1.float的值不为none

2.position不是relative或static

3.display是inline-block或者flex或者inline-flex

4.overflow:hidden

BFC的其他作用：BFC可以取消盒子的margin塌陷，给父盒子加overflow:hidden

BFC可以阻止元素被浮动元素覆盖

7. 实现元素隐藏的方法及区别

(1) 利用opacity透明度来隐藏，opacity设置为0，它只是一种视觉隐藏，元素本身依旧占用它的位置并对网页的布局起到作用，它也同样影响用户交互。在读屏软件中会被识别。

(2) visibility：规定元素 是否可见，一般设置为hidden，它可以很好的隐藏，不会影响用户的交互，在读屏软件中不会被识别。（visibility可能的值）

(3) display：当display设置为none，任何对该元素直接打用户交互操作都不可能生效，被隐藏的元素完全不会占用空间，仿佛元素完全不存在一样。（display可能的值）

(4) position：元素的定位。把position：absolutely；top：-4555px；left：-45545px；top和left设置成足够大的负数，相当于把元素放到可视区域外，它不会影响布局，能够让元素保持可操作性，在读屏软件上可以被识别。(position可能的值）

(5) clip-path：可以创建一个只有元素的部分区域可以显示的剪切区域。

8. 定位position详解

(1) static   默认值

(2) fixed    固定定位

相对于浏览器窗口是固定位置，无论滚动条如何滚动，其位置不会改变。

(3) relative  相对定位

相对定位相对的位置是自己原来的位置。即定位原点为该元素本身所在的位置。进行相对定位后，该元素仍然占有原来的位置。

(4) absolute   绝对定位

定位原点是离自己元素最近一级并且position属性为relative或者absolute的父级元素，如果没有则相对于浏览器定位。和relative属性不同的是，进行绝对定位后，元素不会继续占有原来的位置。

(5) inherit：规定应该从父元素继承 position 属性的值

9.文档流

文档流有三种：标准流，浮动流，定位流；标准流的层级最低，浮动流的层级第二，定位流的层级最高(可以使用z-index调整定位盒子的层级)

position：relative相对于自身原始位置进行定位 不脱离文档流

position：absolute以定位父级为原点进行定位 脱离文档流，若没有定位父级以document文档定位，绝对定位即使没有初始值，也一定要设置值left:0;top:0;

position：fixed以窗口左上角为原点定位，定位之后相对于窗口的位置始终不变 脱离文档流

浮动脱离文档流

脱离文档流的两点影响：

1.后面同父级元素会顶上来

2.父级元素检测不到脱离文档流的元素，从而导致父级高度无法由内容撑开

10. css3、html5的新特性

Html5:

添加了web存储功能，localStorage和sessionStorage

添加了一些列语义化标签header,footer,main,section,aside,nav等

添加了video，radio标签

添加了地理位置定位功能

Input中的type添加了很多属性，如url email number search color

css3添加的新特性：

媒体查询

(1)transform，transition，translate,scale,skelw，rotate等相关动画效果

(2)box-shadow,text-shadow等特效

(3)CSS3 @keyframes 规则，可以自己创建一些动画等

(4)2D、3D转换

(5)添加了border-radius，border-image等属性

(6)CSS3 @font-face 规则,设计师可以引入任意的字体了

11. css3实现动画的属性及区别

通过transition和animation实现动画

主要区别在于：transition需要触发一个事件才会随着时间改变其css属性，animation在不需要触发任何事件的情况下，也可以显式的随时间变化来改变元素的css属性，达到一种动画效果。

12. 实现本地缓存的方法及区别

(1)cookie 指一般网站为了辨别用户身份、进行session跟踪而储存在用户本地终端上的数据（通常经过加密）活跃在浏览器和服务器之间 在设置的有效期之前数据都存在 存储空间为4K 般用于保存用户的认证信息。

(2)localStorage是用来存储ajax返回的数据，解决了cookie存储空间不足的问题，核心API有setItem getItem removeItem clear 浏览器关闭，存储的数据还在，除非手动删除，否则数据不会消失

(3)sessionStorage也是用来存储ajax返回的数据，不是一种永久化存储，浏览器关闭后，数据就会消失。

13. 实现两列布局 (左边宽度固定，右边自适应)

方法一：左侧浮动，设置固定宽度，右侧设置overflow:auto

方法二：利用margin边距和浮动实现两列布局 左侧盒子设置左浮动和固定宽度，右 侧盒子设置margin-left值

方法三：利用margin边距和定位实现两列布局

方法四：父元素设display:flex 左侧盒子设置固定宽度 右侧盒子设置flex:1

14. css中常用单位及区别

(1)px 是一个绝对长度单位

(2)em是相对长度单位。相对于当前对象内文本的字体尺寸。

(3)rem是相对长度单位，相对的只是HTML根元素

15. 如何实现一条0.5px的线

(1)采用meta viewport

<meta name="viewport" content="width=device-width, initial-scale=0.5, minimum-scale=0.5, maximum-scale=0.5"/>

(2)采用transform:scale(0.5,0.5)的方式

16. 什么是css预处理器和后处理器

css预处理器：CSS 预处理器定义了一种新的语言，其基本思想是，用一种专门的编程语言，为 CSS 增加了一些编程的特性，将 CSS 作为目标生成文件，然后开发者就只要使用这种语言进行web页面样式设计

Sass、LESS、cssNext、Stylus 是目前最主流的 CSS 预处理器

css后处理器：CSS 后处理器 是对 CSS 进行处理，并最终生成 CSS 的 预处理器，它属于广义上的 CSS 预处理器。  
我们很久以前就在用 CSS 后处理器 了，最典型的例子是 CSS 压缩工具（如 clean-css），只不过以前没单独拿出来说过。  
还有最近比较火的 Autoprefixer，以 Can I Use 上的 浏览器支持数据 为基础，自动处理兼容性问题。

17. js数据类型介绍

分为基本数据类型和引用数据类型

基本数据类型：string number boolean null undefined symbol

引用数据类型:object 里面包含 Array function Date

typeof “123” =>”string”

typeof 123 =>”number”

typeof true =>”boolean”

typeof null =>”object”

typeof undefined =>” undefined”

typeof [1,2,3] =>”object”

typeof {name:”aa”} =>”object”

typeof 函数 =>”function”

判断是数组还是对象：

1. Array.isArray() (2) instanceof 实例

(3) constructor 构造函数 (4) Object.prototype.toString

18. 如何判断数据类型

typeof instanceof Object.prototype.toString.call()

19. null 和 undefined 的区别

null表示没有对象，此处不应该有值

undefined表示缺少值，就是此处应该有值，但是还没有定义

20. js的事件机制

(1) js中事件的发生包括捕获和冒泡两个阶段，两个阶段的传播顺序为：

捕获:从最外边父元素节点传递至发生事件的元素节点，即由外到内

冒泡:从事件发生节点传递至最外边父元素节点，即有内到外

(2)事件监听

在HTML中对事件进行绑定；

在js中对DOM元素进行事件绑定；

利用监听函数addEventListener("事件名",function(){},false)；

(3) 如何阻止浏览器的默认事件的发生

event.preventDefault()；或者 return false

21. 什么是事件委托

‘事件代理’就是把原本需要绑定的事件委托给父元素，让父元素负责事件监听。事件代理的原理是**DOM元素的事件冒泡**。使用事件代理的好处是可以提高性能，解决新生成元素的事件绑定问题

22. ES6中都新增的哪些常用的方法

let const 解构赋值 模板字符串 扩展运算符 symbol map与set数据结构 箭头函数 rest参数 for of 数组中的方法如reduce filter forEach

proxy **Proxy 可以对目标对象的读取、函数调用等操作进行拦截，然后进行操作处理**

reflect Reflect 可以用于获取目标对象的行为，它与 Object 类似，但是更易读，为操作对象提供了一种更优雅的方式**。**

**Number.parseInt()：**  
用于将给定字符串转化为指定进制的整数。

23. var、let、var的区别



24. 箭头函数与普通函数的区别

(1)箭头函数不能用于构造函数，不能使用new

(2)箭头函数里面没有this，如果要用，this指向定义时它的上下文

(3)箭头函数不能使用arguments，可以使用rest参数解决

(4)不具有prototype原型对象 不具有super

25. 说说对promise的理解

Promise是一个构造函数，可以返回一个promise实例

Promise是为了解决异步优化问题的，

内部有pending fulfilled rejected三种状态 pending表示程序正在执行未得到结果

fulfilled表示成功 rejected表示失败

使用axios发送请求时，返回的就是一个promise对象

26. js中常用的继承方式

Es5:

1.原型链实现继承 2.借用构造函数 3.组合式继承 4.原型式继承 5.寄生式继承 6.组合寄生式继承

Es6:

Extends实现继承

27. 浏览器的渲染原理

第一步：向服务器发请求，将html返回给客户端

第二步：解析html，生成DOM树

第三步：解析CSS，生成CSS规则树

第四步：合并DOM树和CSS规则树，生成render树

第五步：布局render树(layout)，计算各元素的尺寸和位置

第六步：绘制render树(paint)，

28. CSS 能阻塞文档解析吗？

Css加载不会阻塞DOM树的解析

Css加载会阻塞DOM树的渲染

Css加载会阻塞后面js语句的执行

29. 重绘与回流

重绘：对DOM的修改导致了样式的变化，却并未影响几何属性，比如改了颜色或者背景色，浏览器不需要重新计算几何属性，直接为该元素绘制新的样式

回流：对DOM的修改导致了几何尺寸的变化，比如修改了元素的宽高，隐藏元素等，浏览器需要重新计算元素的几何属性，再讲结果绘制出来。

30. 常见的前端性能优化都有哪些

(1)减少http请求次数

(2)使用cnd, 网站上静态资源即css、js全都使用cdn分发，图片亦然

(3) 避免空的src和hre

(4) 为文件头指定Expires

(5) 使用gzip压缩内容

(6) 把JS放到底部

(7) 避免使用CSS表达式

(8) 使用GET来完成AJAX请求

31. JavaScript的垃圾回收机制

垃圾收集器会定期（周期性）找出那些不在继续使用的变量，然后释放其内存

JavaScript垃圾回收的机制很简单：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，所以垃圾回收器会按照固定的时间间隔周期性的执行。

32. 如何加快首屏加载？

减小入口文件体积 静态资源本地缓存 UI框架按需加载 图片资源的压缩 组件重复打包 开启Gzip压缩 使用SSR

33. 什么是JavaScript的执行上下文

执行上下文就是指代码执行的作用域，js中常见的有全局上下文、函数上下文

JS引擎使用执行上下文栈来管理执行上下文。上下文栈确保了JS的执行过程，我们知道JS是单线程的，所以每次需要创建新的上下文就会对应一个上下文栈，遵循先进后出的原则。

34. 如何理解JavaScript中的闭包

一个函数可以访问到另一个函数的内部变量，就形成一个闭包，可以避免垃圾回收机制。

35. 如何理解JavaScript中的作用域链

比如一个变量a，在当前作用域下，如果没有定义，就向父级作用域找，如果父级也没有定义a，就向父级上一层找，一层一层的往上找，就形成了作用域链。

36. js中new一个对象的过程

(1)创建一个空对象，构造函数中的this指向这个空对象

(2)这个新对象被执行原型连接

(3)执行构造方法，属性和方法添加到this引用的对象中，

(4)如果构造函数没有返回其他函数，那么返回this，即创建的新对象，否则返回构造函数的返回对象

37. 当输入一个网址后浏览器都做了什么

(1)DNS解析

(2)TCP连接(三次握手)

(3)发送HTTP请求

(4)服务器处理请求并返回HTTP报文

(5)浏览器解析渲染页面

(6)连接结束

38. 三次握手

(1)客户端向服务器发送SYNJ标志位，并进入SYN\_Sent状态(等待服务器确认)

(2)服务器接收到来自客户端的SYNJ，服务器会确认收到该数据包并发送ACK标志位和SYN k+1标志位，服务器进入SYN\_recvd状态（确认接收并等待客户端确认状态）

(3)客户端进入连接建立状态后，向服务器发送ACK标志位确认客户端已收到连接建立确认，服务器收到ACK标志位后，服务器端进入连接已建立状态

39四次挥手

1. 客户端发送一个报文，表示即将断开连接
2. 服务器回复确认，然后进入一个半关闭状态，如果这个时候服务器没发完的话，继续发，而且客户端需要继续接受
3. 服务器发完数据后，就发送确认释放的报文
4. 客户端收到信息，关闭连接

40.DNS解析过程

(1) 主机先向本地域名服务器进行递归查询

(2) 本地域名服务器采用迭代查询，他先向根域名服务器查询

(3) 根域名服务器告诉本地域名服务器，下一次查询的顶级域名服务器的IP地址

(4) 本地域名服务器向顶级域名服务器进行查询

(5) 顶级域名服务器告诉本地域名服务器下一步应查询的权限域名服务器的IP地址

(6) 本地域名服务器向权限域名服务器进行查询

(7) 权限域名服务器告诉本地域名服务器，所查询的主机的IP地址

(8) 本地域名服务器最后把查询结果告诉主机

整个查询过程共用到了8个UDP报文

41.vue优化

(1)v-if v-show区分使用场景

(2)computed watch区分使用场景

当我们需要进行数值计算，并且依赖于其它数据时，应该使用 computed，因为可以利用 computed 的缓存特性，避免每次获取值时，都要重新计算；

当我们需要在数据变化时执行异步或开销较大的操作时，应该使用 watch，使用 watch 选项允许我们执行异步操作 ( 访问一个 API )，限制我们执行该操作的频率，并在我们得到最终结果前，设置中间状态。这些都是计算属性无法做到的。

(3) v-for 遍历必须为 item 添加 key，且避免同时使用 v-if

(4)图片资源懒加载 安装插件vue-lazyload

(5)路由懒加载

(6)第三方插件的按需引入 借助 babel-plugin-component

(7)避免反复渲染 使用<keep-alive></keep-alive>

42.computed与watch的区别

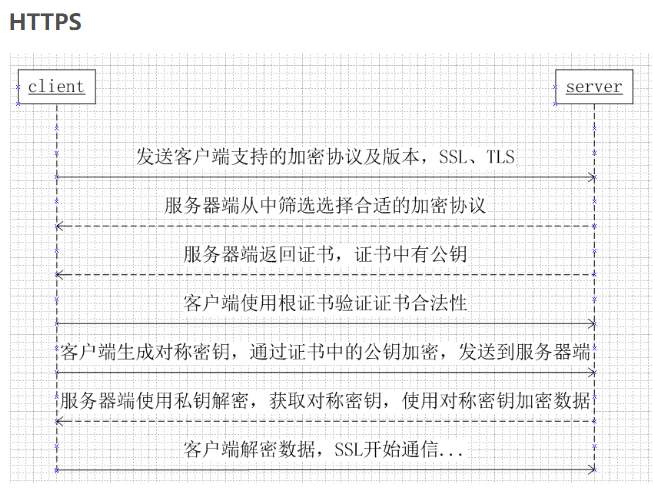
computed：当页面中有某些数据依赖其他数据进行变动的时候，可以使用计算属性

watch：当需要在数据变化时执行异步或开销较大的操作时，这个方式是最有用的。

43.get与post的区别

Get请求是将参数写在url中问号的后面，并用&符号分隔不同的参数，而post是将信息存放在message body中传送，参数不会显示在url中，但post有需要时可以把参数放在url中；get方式需要使用Request.Query.String来取得变量的值，而post方式使用Request.from来获取变量的值，也就是说get通过地址栏传值，而post通过提交表单来传值，get请求提交的数据有长度限制，post请求没有内容限制，http请求本身没有限制url及正文的长度，对url的限制大多是浏览器和服务器的原因，get请求的内容会被浏览器缓存起来，而每次提交post请求，浏览器不会缓存post请求返回的内容，get主要对数据进行查询，post请求主要对数据进行增删改，关于安全性，get请求在url中可以看见参数，所以post更安全。

44.HTTPS连接过程，使用(SSL/TLS)



44.jsonp的原理

从服务器端返回的是一个函数调用，而函数的实参就是给客户端的结果的数据，这个函数要在前端提前声明

第一步：创建script标签

第二步：设置标签的src属性

第三步：将script标签插入文档中

45.nextTick

nextTick在下次 DOM 更新循环结束之后执行延迟回调。

46请实现如下布局，要求自适应各种移动端浏览器，且让内容水平垂直居中，用Vue来实现



flex布局

flex-direction:设置主轴方向 row column row-reverse column-reverse

justify-content:设置主轴上的子元素排列方式 flex-start flex-end center space-around(平分剩余空间) space-between(先贴两边，再平分剩余空间)

flex-wrap:设置子元素是否换行 nowrap(不换行) wrap(换行)

align-content:设置侧轴上的子元素排列方式(多行) flex-start flex-end center space-around(平分剩余空间) space-between(先贴两边，再平分剩余空间) stretch 拉伸

align-items: 设置侧轴上的子元素排列方式(单行) flex-start flex-end center stretch 拉伸

flex-flow:复合属性，相当于同时设置了flex-direction和flex-wrap

47请写出以下console.log输出的结果

var A = function(){};

var a = new A();

console.log(a.\_\_proto\_\_);

console.log(a.\_\_proto\_\_.\_\_proto\_\_);

console.log(a.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_);

48实现一个函数 find(obj, str)，满足:

如var obj = {a:{b:{c:1}}};

find(obj,'a.b.c') //1

find(obj,'a.d.c') //undefined

46.数组相关方法

(1) join方法

array.join(separator) 将数组拼接为字符串

var arr =[1,3,4]

arr.join() 不传参，默认“，”进行拼接

arr.join(‘-’) 按’-’进行拼接

(2) 数组的增删操作

arr.push() 从后面添加元素，返回新数组的length

arr.pop() 从数组的后面删除元素，返回删除的那个元素

arr.unshift() 从数组前面添加元素，返回新数组的长度

arr.shift() 从数组最前面删除元素，返回删除的那个元素

(3)数组的拼接与截取

arr1.concat(arr2) concat不改变原数组

arr1.slice(start,end) 包括start不包括end 不改变原数组

arr1.splice(a,b,c) a表示从哪开始删，b表示删除几个 c表示替换删除的那些元素 改变原数组 返回的是删除的那些元素

(4)清空数组

arr[]=0

arr.length =0

arr.splice(0,arr.length)