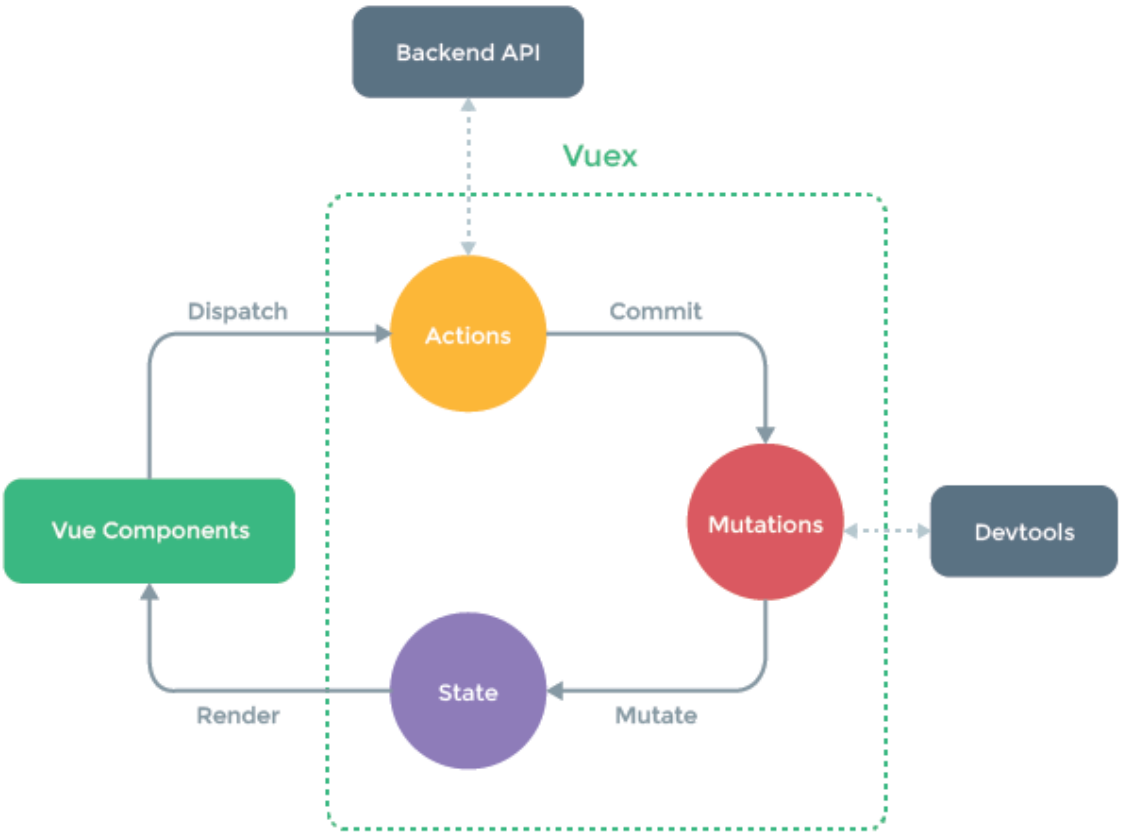
1、OPPO一面

1. 介绍做过的项目
2. 介绍实习经历
3. 监听事件的方法——addEventListener
    - element.addEventListener(*event* , *function* , *useCapture*)三个参数代表的意思：
        - event：必须。字符串，指定事件名
        - function：必须。指定要事件触发时执行的函数
        - useCapture：可选。布尔值，指定事件是否在捕获或冒泡阶段执行
            - true - 事件句柄在捕获阶段执行
            - false- false- 默认，事件句柄在冒泡阶段执行
4. 阻止事件冒泡的方法——stopPropagation
    - stopPropagation也是事件对象(Event)的一个方法，作用是阻止捕获和冒泡阶段中当前事件的进一步传播
5. 事件冒泡有什么作用
    - 事件冒泡允许**多个操作被集中处理**（把事件处理器添加到一个父级元素上，避免把事件处理器添加到多个子级元素上）——event.target.nodeName.toLowerCase()
    - 让**不同的对象同时捕获同一事件**，并调用自己的专属处理程序做自己的事情
6. **介绍Vue的生命周期**
    - Vue 实例从创建到销毁的过程，就是生命周期。也就是从开始创建、初始化数据、编译模板、挂载 Dom→渲染、更新→渲染、卸载等一系列过程，我们称这是 Vue 的生命周期
    - Vue生命周期总共有几个阶段？
        - 总共分为8个阶段：创建前/后，载入前/后，更新前/后，销毁前/销毁后
            - 创建/前后：在beforeCreated阶段，vue实例的挂载元素el还没有；在beforeCreated阶段可以添加loading事件，在created阶段发起后端请求，拿回数据
            - 载入前/后：在beforeMount阶段，vue实例的$el和data都初始化，但是挂载之前为虚拟的dom节点，data.message还未替换，页面无重新渲染。在mounted阶段，vue实例挂载完成，data.message成功渲染
            - 更新前/后：当data变化时，会触发beforeUpdate和updated方法
            - 销毁前/后：在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在
7. 渲染过程是在Vue生命周期的哪个阶段
    - beforeMounted阶段以后就开始进行DOM渲染，而在 mounted 阶段就已经完成了DOM 渲染
8. Vuex的执行流程
    - Vuex 是一个专为 Vue.js 应用程序开发的**状态管理模式。**它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化



9. 做过微信小程序吗——没有
10. PC端和手机端如何自适应大小
    - 采用自适应
11. rem是什么——相对于当前根节点字体的大小
    - px : 最基础的单位（像素）
    - em : 相对于当前父节点字体的大小 ---- 1em = 父节点字体大小
    - rem: 相对于当前根节点字体的大小 ---- 1rem = 根（html）节点字体大小
    - vw: 当前视窗宽度 ---- 1vw = 1%视窗宽度
    - vh: 当前视窗高度 ---- 1vh = 1%视窗宽度
    - vmin: vw和vh中较小的那个
    - vmax: vw和vh中较大的那个
12. 如何识别浏览器内核——window全局对象里的Navigator对象里的userAgent属性
    - 利用navigator.userAgent
        - AppleWebKit——苹果、谷歌内核
        - Trident——IE内核
        - Gecko——火狐内核
        - Presto——opera内核
13. http和https的区别
    - HTTPS 协议需要到 CA （Certificate Authority，证书颁发机构）申请证书，一般免费证书较少，因而需要一定费用
    - 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
    - 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

- 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。(无状态的意思是其数据包的发送、传输和接收都是相互独立的。无连接的意思是指通信双方都不长久的维持对方的任何信息）
    - 注： SSL由从前的网景公司开发，有1、2、3三个版本，但现在只使用版本3；TLS是SSL的标准化后的产物，有1.0、1.1、1.2三个版本，默认使用1.0，TLS1.0和SSL3.0几乎没有区别

14. https如何实现加密的——对称加密+非对称加密+数字证书
15. 自己读过哪些前端书籍——《JavaScript高级程序设计》、《图解HTTP》、《JavaScript DOM编程艺术》
16. Nodejs用过吗——用过
    - 为什么选择express？
        - express既不是最底层的，又不是最抽象的，但是应用广泛，又有典型架构
    - 补充：据说koa2是未来，同样都是express团队开发，支持async、await，处理异步相对express更加优雅
17. webpack使用过吗——学习过，项目中没有使用过

2、猿辅导

一面

1. 自我介绍
2. 求下面代码的运行结果：
    - 代码：

```
Promise.reject(0)
  .then(() => { console.log(1) })
  .catch(() => { console.log(2) })
  .then(() => { console.log(3) })
  .catch(() => { console.log(4) })
  .finally(()=>{ console.log(5)})
  .then(() => { console.log(6) })
```

    - 结果：2 3 5 6
3. **箭头函数的this——箭头函数自身没有this**
    - 默认绑定外层this——普通嵌套函数中的 this 不会从外层函数中继承
    - 不能用call方法修改里面的this
4. 箭头函数与普通函数的区别
    - **语法更加简洁、清晰**
    - **箭头函数没有 prototype (原型)，所以箭头函数本身没有this**
    - **箭头函数不会创建自己的this**
        - 箭头函数中this的指向在它被定义的时候就已经确定了，之后永远不会改变
    - **call | apply | bind 无法改变箭头函数中this的指向**
    - **箭头函数不能作为构造函数使用**
    - **箭头函数不绑定arguments，可以使用扩展运算符来访问箭头函数的参数列表**
    - **箭头函数不能用作Generator函数，不能使用yield关键字**
5. Vue的路由模式

- vue-router 提供了三种模式：
  - hash: 监听浏览器地址hash值变化，执行相应的js切换网页
    - 本身以及它后面的字符称作hash，可通过 `window.location.hash` 属性读取
    - 通过 `hashchange` 事件触发
    - 浏览器前进后退可能导致hash的变化，前提是两个网页地址中的hash值不同
  - history: 利用history API实现url地址改变，网页内容改变
    - History 对象是 window 对象的一部分，可通过 `window.history` 属性对其进行访问
    - History对象基本方法：history.back() 、history.forward()、history.go(n)
    - 新方法：`history.pushState(stateObj, title, url)`
    - 通过 `popstate` 事件触发
  - abstract：使用一个不依赖于浏览器的浏览历史的虚拟管理后端

6. let 暂时性死区
   - 使用 `let、const` 命令声明变量之前，该变量都是不可用的。这在语法上，称为“暂时性死区”
7. **var、const、let区别**
   - **var和const、let区别**
     - 块级作用域
     - 不存在变量提升
     - 暂时性死区
     - 不可重复声明
     - `let、const`声明的全局变量不会挂在顶层对象下面
   - const命令两个注意点：
     - const 声明之后必须马上赋值，否则会报错
     - const 简单类型一旦声明就不能再更改，复杂类型(数组、对象等)指针指向的地址不能更改，内部数据可以更改
8. **作用域链**
   - 应用场景：当使用了一个在当前作用域中不存在的变量时， JavaScript 引擎就需要按照 作用域链 在其他作用域中查找该变量
   - 定义：在 JavaScript 执行过程中，其 作用域链 是由 词法作用域 决定的，而 词法作用域 是根据 代码的位置 来决定的， 通过作用域查找变量的链条称为作用域链
9. **http的强缓存和协商缓存**
   - 强缓存
     - 在早期，也就是HTTP/1.0时期，使用的是**Expires**，而HTTP/1.1使用的是**Cache-Control**
     - Expires：即过期时间，存在于服务端返回的响应头中，告诉浏览器在这个过期时间之前可以直接从缓存里面获取数据，无需再次请求
       - 缺点： 服务器的时间和浏览器的时间可能并不一致 ，那服务器返回的这个过期时间可能就是不准确的
     - Cache-Control：采用过期时长来控制缓存，对应的字段是**max-age**
       - 其他指令：
         - **private** ： 这种情况就是只有浏览器能缓存了，中间的代理服务器不能缓存
         - **no-cache**: 跳过当前的强缓存，发送HTTP请求，即直接进入 协商缓存阶段
       - **no-store** ：非常粗暴，不进行任何形式的缓存

        - **s-maxage**：这和 `max-age` 长得比较像，但是区别在于s-maxage是针对代理服务器的缓存时间
        - **must-revalidate**: 是缓存就会有过期的时候，加上这个字段一旦缓存过期，就必须回到源服务器验证
    - 注意：当**Expires**和**Cache-Control**同时存在的时候，**Cache-Control**会优先考虑
- 协商缓存
    - 强缓存失效之后，浏览器在请求头中携带相应的 `缓存tag` 来向服务器发请求，由服务器根据这个tag，来决定是否使用缓存，这就是**协商缓存**。具体来说，这样的缓存tag分为两种: **Last-Modified** 和 **ETag**
    - Last-Modified：即最后修改时间。在浏览器第一次给服务器发送请求后，服务器会在响应头中加上这个字段。浏览器接收到后，如果再次请求，会在请求头中携带 `If-Modified-Since` 字段，这个字段的值也就是服务器传来的最后修改时间。服务器拿到请求头中的 `If-Modified-Since` 的字段后，就会和这个服务器中 `该资源的最后修改时间` 对比：
    - 如果请求头中的这个值小于最后修改时间，说明是时候更新了。返回新的资源，跟常规的HTTP请求响应的流程一样
    - 否则返回304，告诉浏览器直接用缓存
    - ETag： `ETag` 是服务器根据当前文件的内容，给文件生成的唯一标识，只要里面的内容有改动，这个值就会变。服务器会在响应头中加上这个字段。浏览器接收到ETag的值，会在下次请求时，将这个值作为 `If-None-Match` 这个字段的内容，并放到请求头中，然后发给服务器。服务器接收到 `If-None-Match` 后，会跟服务器上 `该资源的ETag` 进行比对：
        - 如果两者不一样，说明要更新了。返回新的资源，跟常规的HTTP请求响应的流程一样
        - 否则返回304，告诉浏览器直接用缓存
    - 两者对比：
        - 在 `精准度` 上， `ETag` 优于 `Last-Modified` 。优于 ETag 是按照内容给资源上标识，因此能准确感知资源的变化。而 Last-Modified 就不一样了，它在一些特殊的情况并不能准确感知资源变化，主要有两种情况：
            - 编辑了资源文件，但是文件内容并没有更改，这样也会造成缓存失效
            - Last-Modified 能够感知的单位时间是秒，如果文件在 1 秒内改变了多次，那么这时候的 Last-Modified 并没有体现出修改了
        - 在性能上， `Last-Modified` 优于 `ETag` ，也很简单理解， `Last-Modified` 仅仅只是记录一个时间点，而 `Etag` 需要根据文件的具体内容生成哈希值
    - 注意：如果两种方式都支持的话，服务器会优先考虑 `ETag`

10. **响应头有哪些**
    - 常见的如下：
        - server：服务器名字
        - date：表示消息发送的时间
        - status：状态码
        - Cache-Control、Expires、Last-Modified：强缓存和协商缓存字段名
        - Connection：浏览器与服务器之间连接的类型
        - Content-Type：表示后面的文档属于什么MIME类型（字符集）
        - Content-Encoding：压缩方式（gzip）
        - Content-Language：支持语言
        - Content-Length: 发送给接收方的消息主体的大小

- 请求头有哪些：
  - Host：当前请求要被发送的目的地，即目标资源的host，仅包括域名和端口号
  - Origin：当前请求资源所在页面的协议和域名，仅仅包括协议和域名
  - Referer：当前请求资源所在页面的完整路径URL，包括：协议+域名+查询参数
  - User-Agent：浏览器的用户代理字符串
  - Cookie：当前页面设置的任何Cookie
  - Connection：浏览器与服务器之间连接的类型
  - Accept：浏览器能够处理的内容类型
  - Accept-Charset：浏览器能够显示的字符集
  - Accept-Encoding：浏览器能够处理的压缩编码
  - Accept-Language：浏览器当前设置的语言

11. js如何设置cookie
    - cookie 以名/值对形式存储，js中可以通过document.cookie属性来创建，读取以及删除cookie
    - 创建cookie：`document.cookie="username=John Doe";`
    - 读取cookie：`var x=document.cookie;`
    - 修改cookie：`document.cookie="username=John Smith; expires=Thu, 18 Dec 2043 12:00:00 GMT; path=/";`
      - 和创建cookie类似
    - 删除cookie：`document.cookie='username=;expires=Thu,01 Jan 1970 00:00:00 GMT'`
      - 只需要设置expires参数为以前时间即可，如Thu ，01 jan 1970 00：00:00GMT
12. 平时怎么用浏览器调试页面的
    - 利用控制台打印结果调试
    - 利用Sources面板，对网页进行断点调试
    - 利用Network面板，查看请求头和响应头
    - 性能检测工具：Performance和Audits
13. git如何合并多个文件为1个——没懂
14. git pull和git fetch的区别
    - 区别：
      - git pull：相当于是从远程获取最新版本并merge到本地
      - git fetch：相当于是从远程获取最新版本到本地，不会自动merge
    - 相同点：功能是大致相同的，都是起到了更新代码的作用
15. 自定义函数实现模板字符串替换
    - 思路：正则匹配替换
    - 代码：

```
let name = 'fhc'
let age = 18
let str = '我叫${name},今年${age}岁了！'
function replaceStr(str){
    return str.replace(/\$\{(\w+)\}/g, (matched,key)=>{
        try{
            return eval(key) //eval() 函数会将传入的字符串当做 JavaScript 代码进行执行
        }catch(e){
            return matched
```

```
        }
    })
}
let ans = replaceStr(str)
console.log(ans)
```

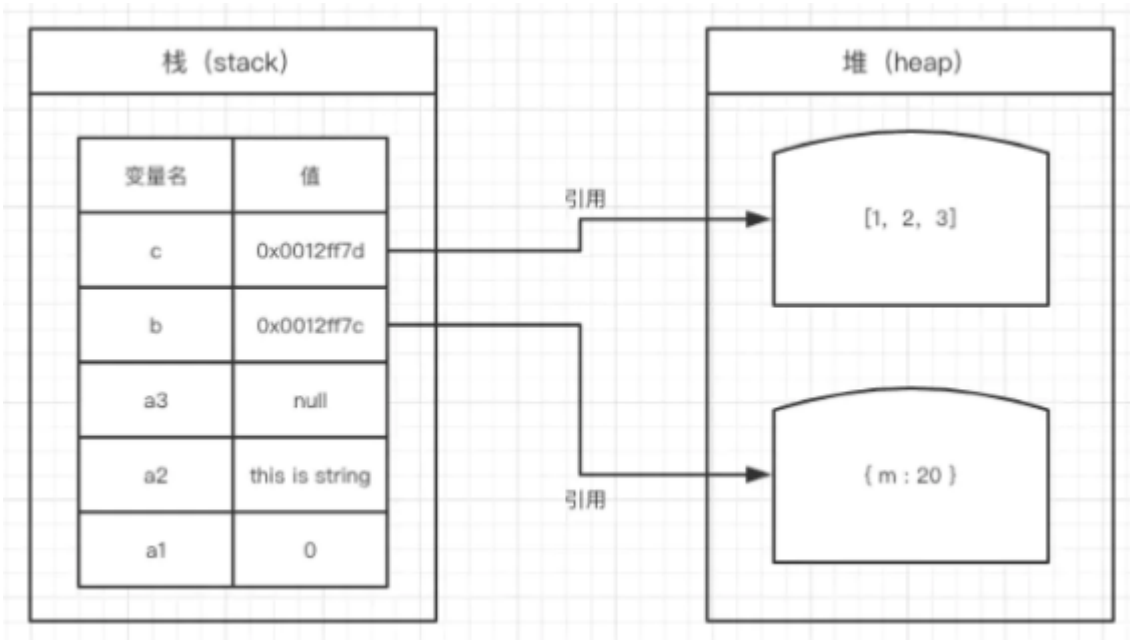
16. 求二叉树的最大宽度
    - 思路：广度优先遍历
    - 代码：

```
function getMaxWidth(root) {
      if (!root) return 0
      let queue = [root] //当前队列
      let res = [] //依次存放所有层节点
      let max = 0
      while (queue.length) {
          res = []
          let len = queue.length
          for (let i = 0; i < len; i++) {
              let cur = queue.shift()
              res.push(cur.val)
              if (cur.left) queue.push(cur.left)
              if (cur.right) queue.push(cur.right)
          }
          max = Math.max(max, res.length)
      }
      return max
  }
```

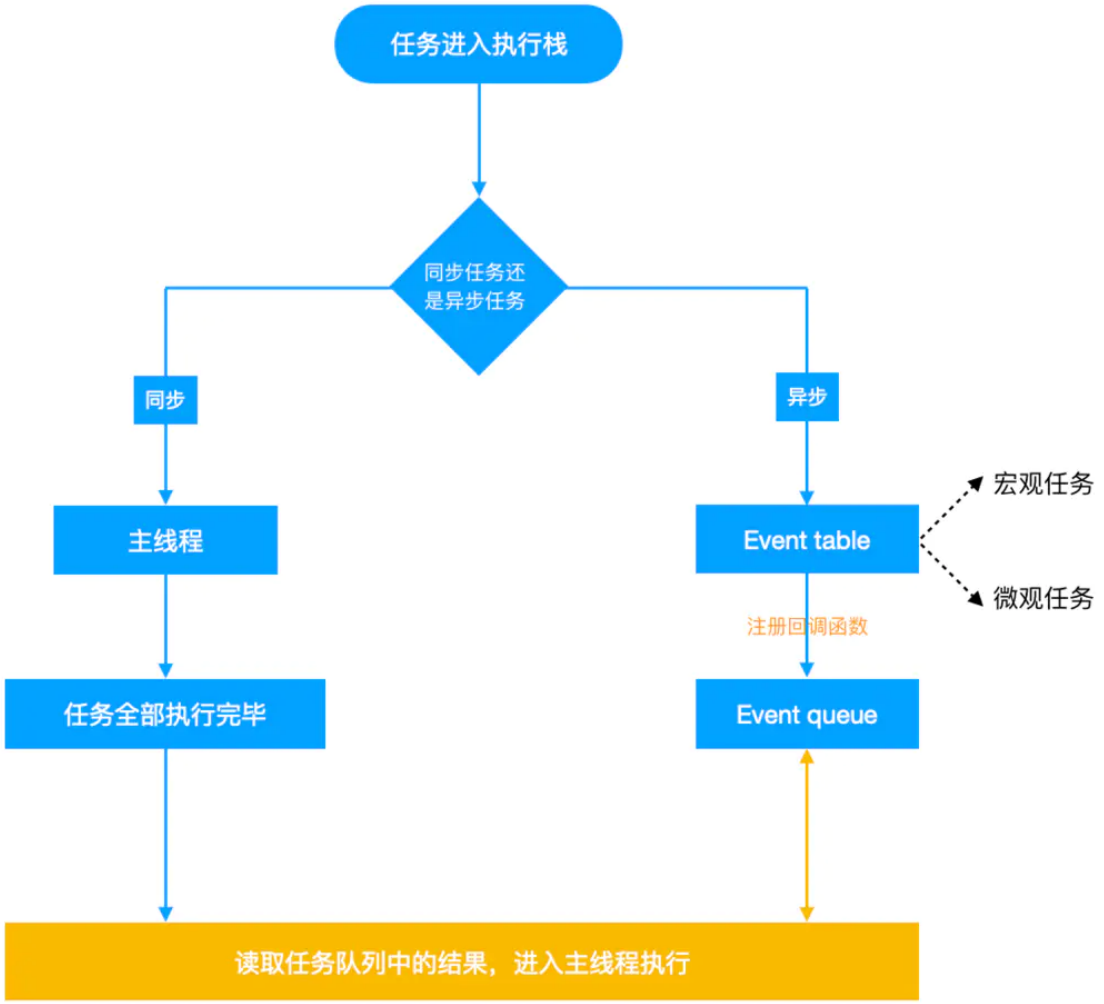
二面

1. 进程和线程的区别
    - 根本区别：进程是**操作系统资源分配的基本单位**，而线程是**任务调度和执行的基本单位**
    - 其他区别：
        - 在开销方面：每个进程都有独立的代码和数据空间（程序上下文），程序之间的切换会有较大的开销；线程可以看做轻量级的进程，同一类线程共享代码和数据空间，每个线程都有自己独立的运行栈和程序计数器（PC），线程之间切换的开销小。
        - 所处环境：在操作系统中能同时运行多个进程（程序）；而在同一个进程（程序）中有多个线程同时执行（通过CPU调度，在每个时间片中只有一个线程执行）
        - 内存分配方面：系统在运行的时候会为每个进程分配不同的内存空间；而对线程而言，除了CPU外，系统不会为线程分配内存（线程所使用的资源来自其所属进程的资源），线程组之间只能共享资源
        - 包含关系：没有线程的进程可以看做是单线程的，如果一个进程内有多个线程，则执行过程不是一条线的，而是多条线（线程）共同完成的；线程是进程的一部分，所以线程也被称为轻权进程或者轻量级进程
2. 一般在http1.1里面，浏览器只支持同时发送6个http请求，有什么办法可以同时发送更多的请求？
    - 使用 CDN 实现域名分片机制
    - http2
    - http3

3. 服务器怎么知道什么时候要去断开tcp连接（比如浏览器关闭了页面，服务器会怎么做）
    - 根据Connection请求头，如果是**keep-alive**服务器就保持住tcp连接，如果**没有或是close**，则服务器response传输完后主动关闭tcp连接。当然现在浏览器都是http1.1都默认是keep-alive的，在浏览器tab关闭时，tcp连接关闭
4. 堆和栈各自的作用及区别
    - 在 JS 中，每一个数据都需要一个内存空间，内存空间又分为栈内存(stack)与堆内存(heap)
        - 栈主要是用来执行程序的，栈内存一般储存基础数据类型；堆一般用来储存引用数据类型
        - 栈(stack)会自动分配内存空间，会自动释放。堆(heap)动态分配的内存，大小不定也不会自动释放
    - 图示：



5. 讲讲事件循环机制（Event Loop）
    -



- 解释：
  - 同步和异步任务分别进入不同的执行场所，同步的进入主线程，异步的进入Event Table并注册函数
  - 当指定的事情完成时（重点），Event Table会将这个函数移入Event Queue中
  - 主线程内的任务执行完毕为空，会去Event Queue读取对应的函数，进入主线程执行
  - 上述的过程会不断的重复，也就是常常说的Event Loop（事件循环）

6. 实现一个类似于Promise.all的方法
7. 求两个数组的并集
   - 示例：比如数组[1,5]和数组[3,9]，并集为[1,9]；比如数组[1,3]和数组[5,9]，因为交集为空，所以结果为false
   - 代码：

```
function arrMerge(arr1,arr2){
    if(arr1[0]>arr2[0]){
        let temp = [...arr1]
        arr1 = arr2
        arr2 = temp
    }
    if(arr1[1]<arr2[0]){
        return false
    }else{
        return [Math.min(arr1[0],arr2[0]),Math.max(arr1[1],arr2[1])]
    }
}
console.log(arrMerge([7,100],[1,9]))
```

3、网易雷火一面

1. 自我介绍
2. 块级元素、行内块级元素和行内元素的区别？
    - 块级元素：
        - 每个块级元素通常都会 独占一行或者是多行 ，可以对其单独 设置高度、宽度以及对齐 等属性
        - 常见的块级元素有： `<h1>~<h6>` , `<p>` , `<div>` , `<ul>` , `<ol>` , `<li>`
        - 块级元素的特点：
            - 块级元素会独占一行
            - 高度，行高，外边距和内边距都可以单独设置
            - 宽度默认是容器的100%
            - 可以容纳内联元素和其他的块级元素
                - 内联元素就是行内元素
    - 行内元素：
        - 不占有独立的区域， 仅仅依靠自己的字体大小或者是图像大小来支撑结构 。 一般不可以设置宽度，高度以及对齐 等属性
        - 常见的行内元素有： `<a>` , `<strong>` , `<b>` , `<em>` , `<del>` , `<span>` 等
            - `<em>` 标签是一个短语标签，用来呈现为被强调的文本，斜体显示
        - 行内元素的特点：
            - 和相邻的行内元素在一行上
            - 高度和宽度无效，但是水平方向上的padding和margin可以设置，垂直方向上的无效
            - 默认的宽度就是它本身的宽度
            - 行内元素只能容纳纯文本或者是其他的行内元素（a标签除外）
        - 注意：
            - 只有文字才能组成段落，因此类似 `<p>,<h1>~<h6>,<dt>` 等里面不能放块级元素，因为它们都是文字块级标签，里面不能再存放其他的块级标签
            - 链接里面不能再存放链接
    - 行内块级元素：

        - 在行内元素中有几个特殊的标签，`<img/>`，`<input/>`，`<td/>`，可以设置它们的宽高度以及对齐属性
            - `<td>` 标签定义 HTML 表格中的标准单元格
        - 行内块级元素的特点：
            - 和相邻的行内元素（行内块）在一行上，但是中间会有空白的间隙
            - 默认的宽度就是本身内容的宽度
            - 高度，行高，内边距和外边距都可以设置
3. 行内元素里面嵌套块级元素会怎样？
    - 比如下面这行代码：

```
<span id="span1">
    before div
    <div style="font-size:22px">div</div>
    after div
</span>
<span id="span2">new span</span>
```

    - 这段 html 浏览器最终会这样解析为：

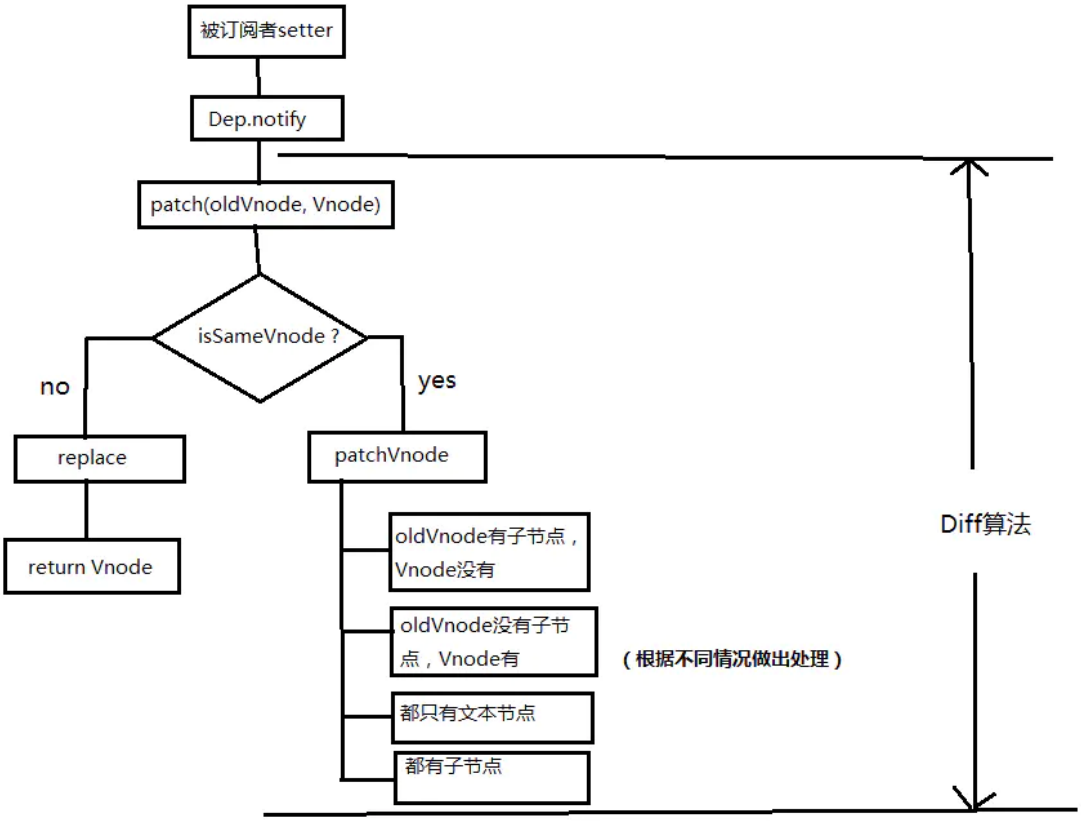
```
<span id="span1">
    before div
</span>
<div style="font-size:22px">div</div>
<span>
    after div
</span>
<span id="span2">new span</span>
```

    - 规矩是不允许在行内元素内写块级元素，但是浏览器是有容错机制的，会自动转换
4. 绝对定位是怎么定位的？
    - position：absolute 元素会被移出正常文档流，并不为元素预留空间，通过指定元素相对于最近的非 static 定位祖先元素的偏移，来确定元素位置。绝对定位的元素可以设置外边距（margins），且不会与其他边距合并
5. 闭包为什么会存在？
    - 个人理解：当 JavaScript 引擎执行到外部函数时，首先会编译，并创建一个空执行上下文。在编译过程中，遇到内部函数，JavaScript 引擎还要对内部函数做一次快速的词法扫描，发现该内部函数引用了外部函数中的某个变量，由于是内部函数引用了外部函数的变量，所以 JavaScript 引擎判断这是一个闭包，于是在堆空间创建换一个“closure(外部函数名)”的对象（这是一个内部对象，JavaScript 是无法访问的），用来保存引用的那个变量
    - 原理：
        - 当某个函数被调用时，会创建一个执行环境（ execution context ）及相应的作用域链。然后，使用 arguments 和其他命名参数的值来初始化函数的活动对象（ activation object ）。但在作用域链中，外部函数的活动对象始终处于第二位，外部函数的外部函数的活动对象处于第三位，……直至作为作用域链终点的全局执行环境。
        - 在另一个函数内部定义的函数会将包含函数（即外部函数）的活动对象添加到它的作用域链中。因此，在外部函数内部定义的匿名函数的作用域链中，实际上将会包含外部函数的活动对象。 外部函数在执行完毕后，其活动对象也不会被销毁，因为匿名函数的作用域链仍然在引用这个活动

对象。换句话说，当外部函数返回后，其执行环境的作用域链会被销毁，但它的活动对象仍然会留在内存中，直到匿名函数被销毁后， 外部函数的活动对象才会被销毁

- 闭包定义：
  - 在 JavaScript 中，根据词法作用域的规则，内部函数总是可以访问其外部函数中声明的变量，当通过调用一个外部函数返回一个内部函数后，即使该外部函数已经执行结束了，但是内部函数引用外部函数的变量依然保存在内存中，我们就把这些变量的集合称为闭包

6. diff算法原理
   - diff的过程就是调用名为 `patch` 的函数，比较新旧节点，一边比较一边给**真实的DOM打补丁**
   - 流程图：



4、百度

一面

1. 自我介绍
2. 项目介绍
3. 项目里如何实现用户验证的？前端如何根据用户权限设置页面的？
4. 说说项目里的一个难点——内存泄漏
5. 有用过typescript吗？——没有
6. require exports 和 import export 的区别
   - require/exports 是 CommonJS 的标准，适用范围如 Node.js；import/export 是 ES6 的标准，适用范围如 Vue、React

    - require/exports 出生在野生规范当中，这种规范是 JavaScript 社区中的开发者自己草拟的规则，得到了大家的承认或者广泛的应用，比如 CommonJS、AMD、CMD 等等；import/export 则是名门正派，在 ES6（ES2015）中包含进来了
    - require/exports 和 import/export 形式不一样
    - require的调用时间为运行时调用，所以require可以出现在文件的任何地方，属于动态加载；import是编译时调用，所以必须放在文件头部，属于静态加载
    - 对于require，如果是module.exports里的基本类型值，会得到该值的副本，如果是module.exports里的对象类型的值，会得到该值的引用；对于import，导入的是值引用，而不是值拷贝
7. 自己设计一个组件时应该考虑到哪些内容？——不知道
    - 就近管理
    - 高复用性
    - 分层设计
    - 灵活扩展
8. vue的双向数据绑定
9. vue router的两种模式
10. vue3用过吗？——没用过
11. react接触过吗？——没接触过
12. 可以说说js里面的闭包吗
13. 闭包的应用
    - 简单回答：
        - 读取函数内部的变量
        - 让这些变量的值始终保持在内存中，不会再f1调用后被自动清除
    - 具体应用场景：
        - 函数节流和防抖
        - 设置私有变量
        - 创建一个计数器
14. http和https的区别
15. 移动端自适应知道吗？——不知道，只做过PC端的
16. 水平垂直居中怎么实现
17. 说说重绘和回流
18. 事件循环机制
19. 读取节点的属性值会引起回流吗？——比如dom.offsetHeigt、dom.width
    - 引起回流的操作有：
        - DOM的增删行为
        - 几何属性的变化
        - 元素位置的变化
        - **获取元素的偏移量属性**
            - 例如获取一个元素的scrollTop、scrollLeft、scrollWidth、offsetTop、offsetLeft、offsetWidth、offsetHeight之类的属性，浏览器为了保证值的正确也会回流取得最新的值，所以如果你要多次操作，最取完做个缓存
        - 页面初次渲染

        - 浏览器窗口尺寸改变

20. 对象的浅拷贝和深拷贝
    - 比如对象`a = {b:'s',c:{d:'e'}}`，设置a.b='v'，浅拷贝的变量会改变吗?

```
let obj = {b:'s',c:{d:'e'}}
let obj_copy = Object.assign({},obj)
// Object.assign的拷贝，是对于第一层属性的拷贝，所以是浅拷贝
obj_copy.b = 'v'
console.log(obj)  //{b:'s',c:{d:'e'}}
let new_obj = obj
console.log(new_obj)  //{b:'s',c:{d:'e'}}
```

21. 编程——多维数组转一维数组
    - 代码：

```
let arr = [1,[2,[3,4,5]]]
function flatten(arr) {
    let result = []
    for(let item of arr){
        if(Array.isArray(item)){
            result = result.concat(flatten(item))
        }else{
            result.push(item)
        }
    }
    return result
}
var result = flatten(arr)
console.log(result)
```

22. Promise.all和Promise.race的区别
    - Promise函数对象的all方法：返回一个promise，只有当所有promise都成功时才成功，否则只要有一个失败的就失败
    - Promise函数对象的race方法：返回一个promise，其结果由第一个完成的promise决定
23. Promise.allSettled：Promise.allSettled跟Promise.all类似，其参数接受一个Promise的数组，返回一个新的Promise， 唯一的不同在于，其不会进行短路， 也就是说当Promise全部处理完成后我们可以拿到每个Promise的状态， 而不管其是否处理成功。
24. 编程——爬楼梯
    - 假设你正在爬楼梯。需要 n 阶你才能到达楼顶。每次你可以爬 1 或 2 个台阶。你有多少种不同的方法可以爬到楼顶呢？
    - 注意：给定 n 是一个正整数
    - 测试样例：
        - 示例 1：
            - 输入： 2
            - 输出： 2
        - 示例 2：
            - 输入： 3

        - 输出：3
    - 思路：动态规划，类似于斐波那契数列
    - 代码：

```
function jumpFloor(number){
      if(number<=1) {
          return number
      }
      let f0 = 1,
          f1 = 1,
          f2
      for(var i=2;i<=number;i++){
          f2=f0+f1
          f0=f1
          f1=f2
      }
      return f2
}
```

二面——电话面试

> 心态很崩的一次面试，卒

1. 自我介绍
2. 项目介绍
3. 实习做的内容
4. 自己项目做的内容
5. 前端优化了解吗
6. 后端node优化了解吗
7. node有哪些应用场景
8. 你觉得你以后的职业发展方向是什么？
9. node底层原理知道吗
10. 前端构建工具知道哪些？webpack会吗？
11. webpack底层原理知道吗？
12. Three.js底层实现原理知道吗？
13. 你说你用过Echarts，那么它的底层原理你知道吗？
14. jquery的原理你知道吗？

5、网易互联网一面

1. 自我介绍
2. 实习项目介绍
3. **CSS实现鼠标放在div上，点击会有圆扩展显示的动画效果**
    - **先实现点击圆扩大并渐变**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
```

```
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>点击圆扩大并渐变</title>
            <style type="text/css">
            html,body{
                /* 保证body里面的div水平垂直居中 */
                height: 100%;
                width: 100;
                display: flex;
                justify-content: center;
                align-items: center;
            }
            .box_1 {
                height: 100px;
                width: 100px;
                background: red;
                /* 设置为圆形 */
                border-radius: 50%;
                /* 光标呈现为指示链接的指针（一只手） */
                cursor: pointer;

                /* 过渡效果 完成过渡效果需要2s 规定以慢速开始和结束的过渡效果*/
                transition: all 3s ease-in-out;
            }
            .box_2 {
                transform: scale(20);
                opacity: 0;
            }
        </style>
    </head>
    <body>
        <div id="box" class="box_1" onclick="myfunc()"></div>
        <script type="text/javascript">
            let divdom  = document.getElementById("box")
            function myfunc(){
                divdom.classList.add("box_2")
            }
        </script>

    </body>
</html>
```

- 最终实现效果：

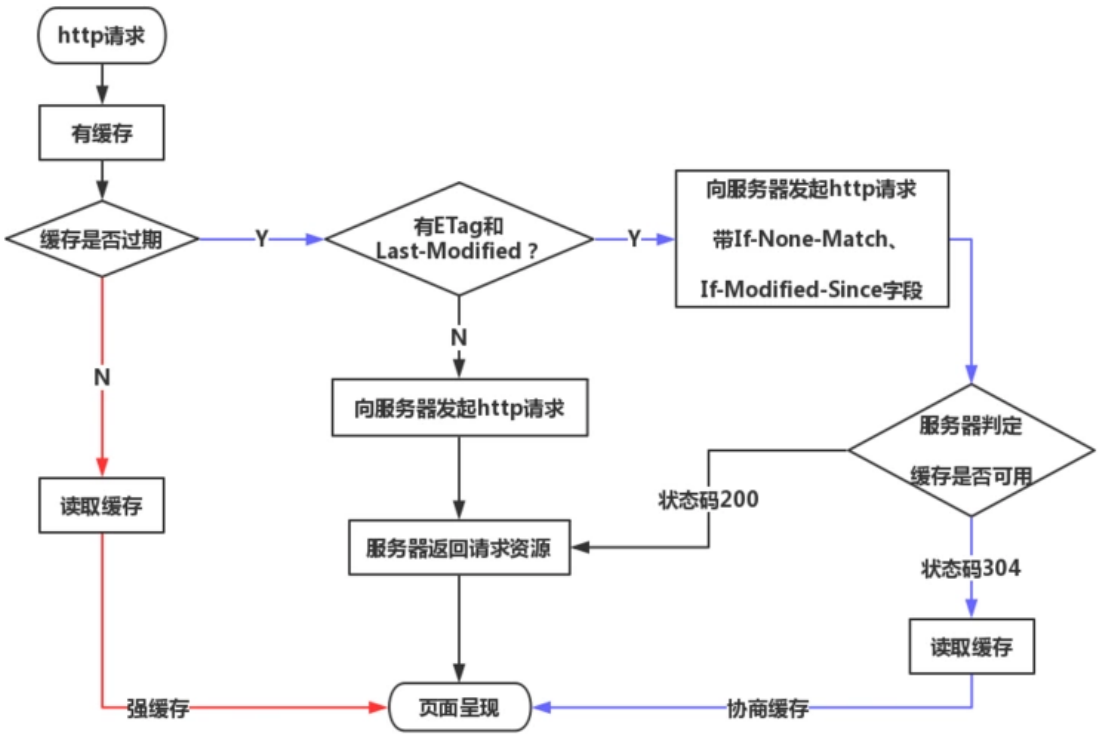
```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>点击圆扩大并渐变</title>
        <style type="text/css">
            html,
            body {
                /* 保证body里面的div水平垂直居中 */
                height: 100%;
```

```
                width: 100;
                display: flex;
                justify-content: center;
                align-items: center;
                overflow: hidden;
            }

            #btn {
                height: 100px;
                width: 200px;
                background: red;
                line-height: 100px;
                text-align: center;
                font-size: 30px;
                color: white;
                cursor: pointer;
            }

            .circle_1 {
                height: 25px;
                width: 25px;
                background: gray;
                opacity: 1;
                /* 设置为圆形 */
                border-radius: 50%;
                position: absolute;
            }

            .circle_2 {
                /* 过渡效果 完成过渡效果需要2s 规定以慢速开始和结束的过渡效果*/
                transition: all 0.8s ease-in-out;
                transform: scale(20);
                opacity: 0;
            }
        </style>
    </head>
    <body>
        <div id="btn" onclick="myfunc()">点击</div>
        <div id="circle" class="circle_1" style="visibility: hidden;"></div>
        <script type="text/javascript">
            function myfunc() {
                document.getElementById("circle").classList.add("circle_2")
                document.getElementById("circle").style.visibility = 'visible'
                setTimeout(function() {
                    document.getElementById("circle").classList.remove("circle_2")
                    document.getElementById("circle").style.visibility = 'hidden'
                }, 800)
            }
        </script>
    </body>
</html>
```

4. border-radius是改变圆角还是什么？

- border-radius用于设置元素的外边框圆角。当使用一个半径时确定一个圆形，当使用两个半径时确定一个椭圆。

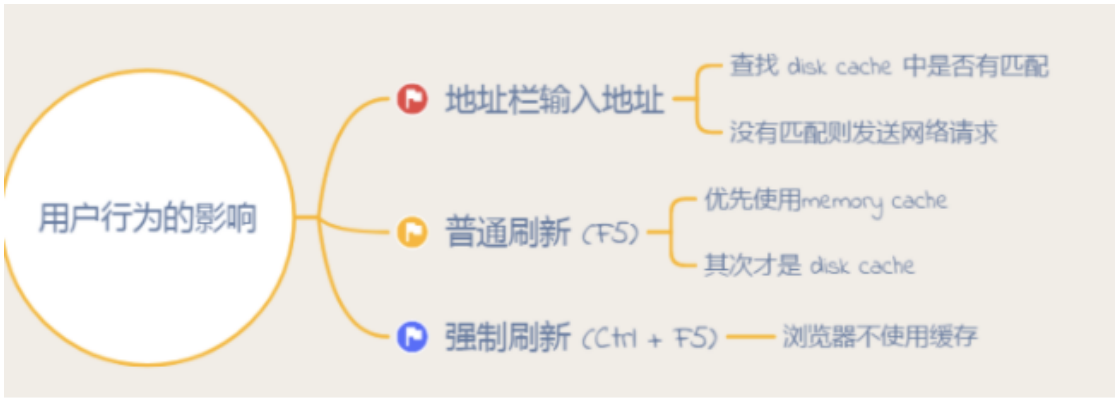
5. 盒模型介绍
6. **一张图片背景的显示范围**——使用到background-clip属性
   - 在CSS3中可以使用 `background-clip` 来修改背景的显示范围，background-clip有三种取值：
     - border-box：背景范围包括边框区域
     - padding-box：背景范围仅包括内部补白区域
     - content-box：背景区域仅包括内容区域
     - text：背景被裁剪成文字的前景色
7. **一张图片缓存的过程（强缓存和协商缓存）**
   - 浏览器第一次向一个web服务器发起 `http` 请求后，服务器会返回请求的资源，并且在响应头中添加一些有关缓存的字段如： `Cache-Control` 、 `Expires` 、 `Last-Modified` 、 `ETag` 、 `Date` 等等。之后浏览器再向该服务器请求该资源就可以视情况使用**强缓存**和**协商缓存**。
   - 具体图示：



   - **缓存分类**：
     - 宏观上：
       - 私有缓存：私有缓存就是用户专享的，各级代理不能缓存的缓存
       - 共享缓存：共享缓存就是那些能被各级代理缓存的缓存
     - 微观上：
       - 浏览器缓存
       - 代理服务器缓存
       - CDN缓存
       - 数据库缓存
       - 应用层缓存

- **缓存位置**：
  - 图示：

- **用户操作对缓存的影响**
  - 图示：



8. 防抖和节流的应用，防抖函数里怎么传参
   - 防抖应用场景：
     - 在进行搜索的时候，当用户停止输入后调用方法，节约请求资源
   - 节流应用场景：
     - 典型的案例就是鼠标不断点击触发，规定在n秒内多次点击只有一次生效
   - 防抖函数里直接传入 arguments 即可
9. 实现Promise.all的类似方法
10. **前端常用到的一些设计模式？**
    - 工厂模式
      - 定义：**工厂模式抽象了创建具体对象的过程，即：发明一种函数，用函数来封装以特定接口创建对象的细节**
    - 单例模式
      - 定义：最简单的一个模式，指的是**在系统中保存一个实例，相当于定义了一个全局变量**，如果要使用可以直接调用，但是如果更改，可能会影响其他调用时场景
    - 观察者模式\发布-订阅模式
      - 定义：发布-订阅模式，他定义了一种一对多的依赖关系，即当一个对象的状态发生改变的时候，所有依赖他的对象都会得到通知

    - 适配器模式
        - 定义：将一个类（对象）的接口（方法或属性）转化成客户希望的另外一个接口（方法或属性），适配器模式使得原本由于接口不兼容而不能一起工作的那些类（对象）可以一些工作
        - 适配器模式在前端项目中一般会用于做数据接口的转换处理，**比如把一个有序的数组转化成我们需要的对象格式**
    - 策略模式
        - 定义：策略模式，定义一系列的算法，把它们一个个封装起来，并且使它们可以相互替换
        - 简单来讲，就是完成一个方法过程中，可能会用到一系列的工具，通过外部传入区分类别的参数来达到使用不同方法的封装
    - 命令模式
- 用来对方法调用进行参数化处理和传送，经过这样处理过的方法调用可以在任何需要的时候执行
    - 职责链模式
        - 使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系，将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理它为止。（大函数分割成一个个小函数，清晰，各司其职）

11. 数组方法有哪些
    - **改变原数组的方法**
        - pop()：删除数组最后一个元素，并返回该元素
        - push()：在数组尾部添加元素，并返回更新后的数组长度
        - shift()：删除数组的第一个元素，并返回该元素
        - unshift()：在数组第一位添加元素，并返回更新后的数组长度
        - sort()：对数组排序（按字符ASCII进行排序），也可添加回调函数按照想要的规则排序
        - reverse()：数组反转
        - splice(index, howmany, 新数据)：返回被删除元素所组成的数组
        - fill()：将一个固定值替换数组的元素
        - copyWithin(target[, start[, end]])：复制数组的一部分到同一数组中的另一个位置，并返回它，不会改变原数组的长度，但是会改变原数组里的值
    - 不改变数组的方法
        - slice()
        - map()
        - forEach()
        - every()
        - some()
        - filter()
        - reduce()
        - find()
        - concat()
        - entries()：返回一个数组的迭代对象，该对象包含数组的键值对 (key/value)
12. 将两个数组合并排序并去重

6、贝壳

一面

1. 自我介绍

2. 实习+项目介绍
3. 防抖函数的实现
4. 实习用到了Three.js这个库，它是怎么实现鼠标点击事件的
5. 在Vue的created阶段还是mounted阶段发送请求合适？
6. localstorage和sessionstorage的区别？localstorage里面的值可以设置过期时间吗？
    - localStorage和sessionStorage一样都是用来存储客户端临时信息的对象。他们均只能存储字符串类型的对象（虽然规范中可以存储其他原生类型的对象，但是目前为止没有浏览器对其进行实现）
    - 区别：
        - localStorage生命周期是永久，这意味着除非用户显示在浏览器提供的UI上清除localStorage信息，否则这些信息将永远存在
        - sessionStorage生命周期为当前窗口或标签页，一旦窗口或标签页被永久关闭了，那么所有通过sessionStorage存储的数据也就被清空了
7. 字符串反转的方法
    - 方法一——reverse()+join()
        - 代码：

```
function reverseStr(str){
    return str.split("").reverse().join("")
}
```

    - 方法二
        - 代码：

```
function reverseStr(str){
    let temp = []
    for(let i=0;i<str.length;i++){
        temp.unshift(str[i])
    }
    return temp.join("")
}
```

    - 方法三——charAt()
        - 代码：

```
function reverseStr(str) {
    let ans = ""
    for (let i = str.length-1; i >= 0; i--) {
        ans += str.charAt(i)
    }
    return ans
}
```

8. 有什么想问的？

二面

1. 自我介绍
2. 觉得前端的发展怎么样

3. 会webpack吗？
4. webpack打包css、less怎么实现的
5. babel有什么作用？
    - `Babel`是一个`JavaScript`编译器，很多浏览器目前还不支持ES6的代码，`Babel`的作用就是把浏览器不支持的代码编译成支持的代码
    - 注意很重要的一点就是，`Babel`只是转译新标准引入的语法，比如ES6的箭头函数转译成ES5的函数，但是对于新标准引入的新的原生对象，部分原生对象新增的原型方法，新增的API等（如`Set`、`Promise`），这些`Babel`是不会转译的，需要引入`polyfill`来解决
6. babel/presets-env有什么作用？——用于兼容？
    - 示例：

```
module: {
    rules: [
        /*
        js兼容性处理：babel-loader  @babel/core
          1.基本js兼容性处理 -->  @babel/preset-env
            问题：只能转换基本语法，如promise高级语法不能转换
          2.全部js兼容性处理 --> @babel/polyfill
            问题：我只要解决部分兼容性问题，但是将所有兼容性代码全部引入，体积太大
          3.需要做兼容性处理的就做：按需加载 --> core-js
      */
        {
            test:/\.js$/,
            exclude: /node_modules/,
            loader: 'babel-loader',
            options: {
                //  预设：指示babel做怎么样的兼容性处理
                presets: [
                    [
                        '@babel/preset-env',
                        {
                            // 按需加载
                            useBuiltIns: 'usage',
                            // 指定core-js版本
                            corejs:{
                                version: 3
                            },
                            // 指定兼容性支持哪个版本以上的浏览器
                            targets:{
                                chrome:'60',
                                firefox:'60',
                                ie:'9',
                                safari:'10',
                                edge:'17'
                            }
                        }
                    ]
                ],
            }
        }
    ]
```

```
}
```

7. **如何实现在一个窗口内通过鼠标拖动div滑动？**
    - 思路：
        - 设置div为absolute，通过top和left的改变来确定div位置
        - 监听鼠标事件
            - 监听mousedown事件
            - 监听mousemove事件
                - 通过鼠标偏移量计算出新的left和top，然后通过更改style.left和style.top即可
            - 监听onmouseup事件
8. 监听的事件Event有哪些属性和方法，鼠标位置怎么获取
9. vue实践经验
10. 如何检测一个字符串是不是手机号

- 答案：
    - 方法一——不推荐
        - 思路：比较笨的方法，去判断字符串里的每一个字符
        - 代码：

```
function isLocaleMobileNumber(str){
    if(str.length!==11){
        return false
    }else{
      for(let i=0;i<str.length;i++){
            if(!(str[i]>='0' && str[i]<='9')){
              return false
            }
        }
    }
    return true
}
let ans = isLocaleMobileNumber('18717889293')
console.log(ans)
```

        - 另一种思路——不推荐

```
function isLocaleMobileNumber(str){
    if(!Number(str)){
        return false
    }
    if(str.length === 11){
        return true
    }else{
        return false
    }

}
let ans = isLocaleMobileNumber('18717889293')
console.log(ans)
```

- 方法二——推荐
  - 思路：正则表达式
  - 代码：

```
// 手机号有如下规则:
// (1)必须全为数字
// (2)必须是11位.(有人说还有10位的手机号,这里先不考虑)
// (3)必须以1开头(有人见过以2开头的手机号吗?)
// (4)第2位是34578中的一个
function isLocaleMobileNumber(str){
    var pattern = /^1[34578]\d{9}$/
    if(pattern.test(str)) {
        return true
    }else{
        return false
    }
}
let ans = isLocaleMobileNumber('18717889293')
console.log(ans)
```

11. webgl和three.js的关系
12. shader和gsls是什么
13. DNS是什么
14. three.js里面的相机有哪些
    - OrthographicCamera
    - PerspectiveCamera
15. 实习项目Three.js这个具体实现了什么样的效果？
    - 里面的文字是怎么渲染的？
    - 鼠标点击事件怎么捕获？
      - rayCuster原理
    - Material Geometry Mesh三者是什么？
16. git操作用过哪些？
17. linux常见命令有哪些？

18. 比如要提交某些文件，但是要忽略其中一些文件，该用什么git命令？——gitignore
19. 访问一个网址，返回一个html，如果用node，大致要怎么实现？
    - node里面通过fs去读文件，然后返回给客户端

三面

1. 自我介绍（不要说简历上有的，主要是讲一些生活学习相关的）
2. 实习学到了什么东西？
3. 对城市有什么倾向吗？
4. 做过最有成就感的事情
5. 研电赛你在里面担任什么角色？数学建模比赛你在里面担任什么角色？
6. 你觉得自己的优势是什么？
7. 对贝壳有了解吗？为什么要投递贝壳？有师兄师姐在贝壳吗？
8. 你现在手上有offer吗，如果有，你会怎么选择offer？
9. 有什么想问我的吗？

7、腾讯

一面

1. 自我介绍
2. 实习项目介绍
3. git merge 操作冲突了怎么办
    - 方法一——很傻的办法
        - 将 `<<<<<<< <分支名>` `========` `>>>>>>>>><分支名>` 删掉，`冲突代码自行合并修改`
        - git add <已修改冲突文件>

          ```
          17412@LAPTOP-86IC3VGF MINGW64 /d/git/gitmerge (first|MERGING)
          $ git add hello.txt
          ```

        - git commit -m "合并日志"

          ```
          17412@LAPTOP-86IC3VGF MINGW64 /d/git/gitmerge (first|MERGING)
          $ git commit -m '合并first分支与second分支'
          [first 6e76696] 合并first分支与second分支

          17412@LAPTOP-86IC3VGF MINGW64 /d/git/gitmerge (first)
          $
          ```

            - 注意：分支名由（first|MERGING）变成（first）说明合并成功
    - 方法二——暴力办法
        - 先确定需要的是哪个仓库的文件：
            - git checkout --theirs 文件名 //使用版本库的里版本覆盖本地，相当于放弃本地修改
            - git checkout --ours 文件名 //使用自己修改的内容覆盖本地（因为目前本地已经是merge过了的），相当于放弃服务器的内容
        - 上面操作执行完成后，还需要 add 、commit 操作，才算完成。
            - git add -A
            - git commit -m "update conflict
4. vue的生命周期

5. vuex的作用以及用法，包含啥
6. vue3的新特性有哪些
7. vue的computed和data有什么区别
8. 双向数据绑定时，如果data里面新增一个属性，页面会进行重新渲染吗
    - `Vue2.x` 对新增属性是无感知的，依赖收集发生在初始化组件的过程中，`Vue` 不能响应后来新增的属性，这是 `Object.defineProperty` 天生的特性
9. vue的路由模式
10. vue的路由守卫有些啥
11. 前端项目是怎么部署的？
12. webpack使用过吗？
13. vue的组件传值
14. js里的instanceOf怎么确定是父类还是子类
    - instanceOf 运算符用来 测试一个对象在其原型链中是否存在一个构造函数的 `prototype` 属性
15. 事件循环机制

一面

1. js造成内存泄漏的原因有哪些？
2. 闭包的定义、使用场景、优缺点
3. 为什么一般情境下都是事件冒泡触发事件，而不是事件捕获阶段触发事件？
    - 支持W3C标准的浏览器在添加事件时用addEventListener(event,fn,useCapture)方法，基中第3个参数useCapture是一个Boolean值，用来设置事件是在事件捕获时执行，还是事件冒泡时执行。而不兼容W3C的浏览器(IE)用attachEvent()方法，此方法没有相关设置，不过IE的事件模型默认是在事件冒泡时执行的，也就是在useCapture等于false的时候执行，所以**在处理事件时把useCapture设置为false是比较安全，也实现兼容浏览器的效果**
4. 事件委托的好处？
    - 提高JavaScript性能。事件委托可以显著的提高事件的处理速度，减少内存的占用
    - 动态的添加DOM元素，不需要因为元素的改动而修改事件绑定
5. DOM0 DOM1 DOM2 DOM3各代表什么事件
    - DOM0 事件：**通过 onclick 绑定到 html上的事件**
        - DOM0级事件具有极好的跨浏览器优势，会以最快的速度绑定
        - 同一个元素的同一种事件只能绑定一个函数，否则后面的函数会覆盖之前的函数
        - 清理DOM0事件时，只需给该事件赋值为 null
        - 不存在兼容性问题
    - DOM1事件：DOM1一般只有设计规范没有具体实现，所以一般跳过
    - DOM2事件：**通过 addEventListener 绑定的事件**
        - 同一个元素的同种事件可以绑定多个函数，按照绑定顺序执行
        - 清除 DOM2事件时，使用 removeEventListener
        - DOM2 包含3个事件：事件捕获阶段、处于目标阶段和事件冒泡阶段
        - DOM2级事件有三个参数：
            - 第一个参数是事件名（如click）

        - 第二个参数是事件处理程序函数
        - 第三个参数如果是true则表示在捕获阶段调用，为false表示在冒泡阶段调用
    - DOM3事件：在DOM2级事件的基础上添加了更多的事件类型
        - 全部类型如下：
            - UI事件，当用户与页面上的元素交互时触发，如：load、scroll
            - 焦点事件，当元素获得或失去焦点时触发，如：blur、focus
            - 鼠标事件，当用户通过鼠标在页面执行操作时触发如：dbclick、mouseup
            - 滚轮事件，当使用鼠标滚轮或类似设备时触发，如：mousewheel
            - 文本事件，当在文档中输入文本时触发，如：textInput
            - 键盘事件，当用户通过键盘在页面上执行操作时触发，如：keydown、keypress
            - 合成事件，当为IME（输入法编辑器）输入字符时触发，如：compositionstart
            - 变动事件，当底层DOM结构发生变化时触发，如：DOMsubtreeModified
            - 同时DOM3级事件也允许使用者自定义一些事件
6. 如果一个页面里，有js文件、css文件和图片，浏览器会优先处理哪个文件？
    - 一般是先下载的html文件，然后并行加载css和js，最后下载图片等资源
7. 事件循环机制
8. 如果A页面和B页面同域，B页面可以获取A页面的SessionStorage的内容吗？
    - 不可以！
9. 浏览器缓存有哪些？存储的都是些什么？
10. indexDB存储的是些什么
    - **将大量数据储存在客户端**，这样可以减少从服务器获取数据，直接从本地获取数据。通俗地讲，IndexedDB 就是浏览器提供的本地数据库，它可以被网页脚本创建和操作。IndexedDB 允许储存大量数据，提供查找接口，还能建立索引。这些都是 LocalStorage 所不具备的。就数据库类型而言，IndexedDB 不属于关系型数据库（不支持 SQL 查询语句），更接近 NoSQL 数据库。
11. Vue里的nextTick是什么意思，工作原理
    - Vue 实现响应式并不是数据发生变化之后 DOM 立即变化，而是按一定的策略进行 DOM 的更新。简单来说，Vue 在修改数据后，视图不会立刻更新，而是等**同一事件循环**中的所有数据变化完成之后，再统一进行视图更新。
    - 官方解释：Vue 异步执行 DOM 更新。只要观察到数据变化，Vue 将开启一个队列，并缓冲在同一事件循环中发生的所有数据改变。如果同一个 watcher 被多次触发，只会被推入到队列中一次。这种在缓冲时去除重复数据对于避免不必要的计算和 DOM 操作上非常重要。然后，在下一个的事件循环“tick”中，Vue 刷新队列并执行实际 (已去重的) 工作。Vue 在内部尝试对异步队列使用原生的 Promise.then 和 MessageChannel，如果执行环境不支持，会采用 setTimeout(fn, 0) 代替
    - 应用场景：需要在视图更新之后，基于新的视图进行操作。
12. 如何防范XSS攻击
13. 如何防范CSRF攻击

8、58同城

一面

1. 自我介绍
2. 实习项目介绍
3. vue的生命周期
4. Vue3的新特性了解吗

> 总体来说：1. 更快 2. 更小 3. 更容易维护 4. 更加友好 5. 更容易使用

- **响应系统的变动**
  - 由原来的Object.defineProperty 的getter 和 setter，改变成为了ES2015 Proxy 作为其观察机制。
  - Proxy的优势：消除了以前存在的警告，使速度加倍，并节省了一半的内存开销
- **虚拟DOM重写（Virtual DOM Rewrite）**
  - 虚拟 DOM 从头开始重写，我们可以期待更多的编译时提示来减少运行时开销。重写将包括更有效的代码来创建虚拟节点
- **组件渲染的优化（优化插槽生成）**
  - Vue2当中在父组件渲染同时，子组件也会渲染。 Vue3就可以单独渲染父组件、子组件
- **静态树提升（Static Tree Hoisting）**
  - 使用静态树提升，这意味着 Vue 3 的编译器将能够检测到什么是静态组件，然后将其提升，从而降低了渲染成本。它将能够跳过未整个树结构打补丁的过程
- **静态属性提升（Static Props Hoisting）**
  - Vue 3 将跳过不会改变节点的打补丁过程

5. 目标检测和跟踪用的都是什么算法，可以自己搞一个算法出来实现目标跟踪吗
6. es6的新语法有哪些
7. https和http的区别
8. webpack里面的loader和plugin的区别是什么
9. 其他忘了.....

二面

1. 自我介绍
2. 几乎随便聊天、聊职业规划，没问什么技术问题

三面

1. 自我介绍
2. 实习学到了什么
3. 为什么搞前端开发
4. 比赛中学到了什么
5. 自己性格啥的
6. 意向工作城市
7. 手里有其他offer吗
8. 反问

9、拼多多

一面

1. 自我介绍
2. 有两个input输入和一个按钮，点击按钮，间隔500ms后打印输出两个input相加的值（考虑input错误输入、数据过大）
3. 宏任务是什么，宏任务和微任务有哪些

4. 给一个乱序数组，求第几大的某个数（用冒泡，复杂度过大，考虑其他）
    - Top K 问题，堆排序——不会
5. 问实习内容
6. 反问

10、虾皮

一面

1. 自我介绍
2. 实习介绍
3. 项目介绍
4. 问了上机笔试做的三道选择题
5. 其他忘了……

二面

1. 自我介绍
2. 实习介绍
3. 项目介绍
4. 做项目时遇到的一些难点——讲的内存泄漏的问题
5. 对于实习做的内容问的很深
6. 实习时用到了Three.js这个库，面试官就疯狂深入问这个底层原理，卒
    - WebGL是什么？
        - `WebGL(Web图形库)是一种JavaScript API`，用于在任何兼容的Web浏览器中呈现交互式3D和2D图形，而无需使用插件。`WebGL通过引入一个与OpenGL ES 2.0紧密相符合的API，该API可以在HTML5<canvas>元素中使用`。 在我的理解，WebGL给我们提供了一系列的图形接口，能够让我们通过js去使用GPU来进行浏览器图形渲染的工具
        - WebGL 是大部分浏览器直接支持的一种3D绘图标准，three.js在它的基础上进行了进一步的封装和简化开发开发过程
    - Three.js是什么？
        - `Three.js是一款webGL框架`，由于其易用性被广泛应用。Three.js在WebGL的api接口基础上，又进行的一层封装。它是由居住在西班牙巴塞罗那的程序员Ricardo Cabbello Miguel开发的。Three.js以简单、直观的方式封装了3D图形编程中常用的对象。Three.js在开发中使用了很多图形引擎的高级技巧，极大地提高了性能。另外，由于内置了很多常用对象和极易上手的工具，Three.js的功能也非常强大。最后，Three.js还是完全开源的，你可以在GitHub上找到它的源代码，并且有很多人贡献代码，帮助Mr.doob一起维护这个框架
    - WebGL和Three.js的关系
        - WebGL原生的api是一种非常低级的接口，而且还需要一些数学和图形学的相关技术。对于没有相关基础的人来说，入门真的很难，Three.js将入门的门槛降低了整整的一大截，对WebGL进行封装，简化我们创建三维动画场景的过程
        - 补充
            - canvas和WebGL的关系：canvas就是画布，只要浏览器支持，可以在canvas上获取2D上下文和3D上下文，其中3D上下文一般就是WebGL

        - WebGL和OpenGL的关系：`OpenGL是底层的驱动级的图形接口`（是显卡有直接关系的），但是这种底层的OpenGL是寄生于浏览器的JavaScript无法涉及的，为了让Web拥有更强大的图形处理能力，2010年的时候WebGL被推出来了。`WebGL`允许工程师`使用JS`去调用`部分封装过的 OpenGL ES2.0 标准接口`去 提供硬件级别的3D图形加速功能
            - 三者的关系是`JavaScript -> WebGL -> OpenGL ->.... -> 显卡`，并把最终渲染出来图形呈现到`Canvas`
    - Three.js的优缺点
        - 优点
            - Three.js掩盖了3D渲染的细节：Three.js将WebGL原生API的细节抽象化，将3D场景拆解为`网格、材质和光源`(即它内置了图形编程常用的一些对象种类)
            - 面向对象：开发者可以使用上层的JavaScript对象，而不是仅仅调用JavaScript函数
            - 功能非常丰富：Three.js除了封装了WebGL原始API之外，Three.js还包含了许多实用的内置对象，可以方便地应用于游戏开发、动画制作、幻灯片制作、高分辨率模型和一些特殊的视觉效果制作
            - 速度很快：Three.js采用了3D图形最佳实践来保证在不失可用性的前提下，保持极高的性能
            - 支持交互：WebGL本身并不提供拾取（picking)功能（即是否知道鼠标正处于某个物体上）。而Three.js则固化了拾取支持，这就使得你可以轻松为你的应用添加交互功能
            - 包含数学库：Three.js拥有一个强大易用的数学库，你可以在其中进行矩阵、投影和矢量运算
            - 内置文件格式支持：可以使用流行的3D建模软件导出文本格式的文件，然后使用Three.js加载；也可以使用Three.js自己的JSON格式或二进制格式
            - 扩展性很强：为Three.js添加新的特性或进行自定义优化是很容易的事情。如果你需要某个特殊的数据结构，那么只需要封装到Three.js即可
            - `支持HTML5 canvas：Three.js不但支持WebGL，而且还支持使用Canvas2D、Css3D和SVG进行渲染`。在未兼容WebGL的环境中可以回退到其它的解决方案
        - 缺点：
            - 官网文档非常粗糙，对于新手极度不友好
            - 国内的相关资源匮乏
            - Three.js所有的资料都是以英文格式存在，对国内的朋友来说又提高了门槛
            - Three.js不是游戏引擎，一些游戏相关的功能没有封装在里面，如果需要相关的功能需要进行二次开发
            - Three.js与其他库的对比
            - 随着WebGL的迅速发展，相关的WebGL库也丰富起来
7. Canvas是什么
    - **Canvas API** 提供了一个通过JavaScript 和 HTML的元素来绘制图形的方式。它可以用于动画、游戏画面、数据可视化、图片编辑以及实时视频处理等方面

    - `Canvas API`主要聚焦于2D图形，而同样使用
8. Canvas和SVG的区别是什么
    - 一句话总结：都是2D做图，svg是矢量图，canvas是位图。Canvas 是逐像素进行渲染的，适合游戏

- SVG
  - SVG 是一种基于 XML 语法的图像格式，全称是`可缩放矢量图`（Scalable Vector Graphics）。`其他图像格式都是基于像素处理的`，`SVG 则是属于对图像的形状描述`，所以它本质上是文本文件，体积较小，且不管放大多少倍都不会失真
  - 因为是矢量图，缩放不影响显示
- 区别：
  - SVG 是一种使用 XML 描述 2D 图形的语言。SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。
    - 特点：
      - 不依赖分辨率
      - 支持事件处理器
      - 最适合带有大型渲染区域的应用程序（比如谷歌地图）
      - 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快）
      - 不适合游戏应用
  - **Canvas 通过 JavaScript 来绘制 2D 图形。Canvas 是逐像素进行渲染的。**在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。
    - 特点：
      - 依赖分辨率
      - 不支持事件处理器
      - 弱的文本渲染能力
      - 能够以 .png 或 .jpg 格式保存结果图像
      - 最适合图像密集型的游戏，其中的许多对象会被频繁重绘

9. 画一个三角形
10. CSS实现动画效果的方法有哪些
    - `transition：定义了元素在变化过程中是怎么样的`
      - transition-property：接要改变的属性，如width、height、all等
      - transition-duration：定义了过渡效果花费的时间
      - transition-timing-function：规定过渡效果的时间曲线。默认是 “ease”
      - transition-delay：规定过渡效果何时开始。默认是 0
    - `transform：定义了元素的变化的最终结果`
      - translate(x,y)：定义 2D 转换，沿着 X 和 Y 轴移动元素
      - translateX(n)：定义 2D 转换，沿着 X 轴移动元素
      - translateY(n)：定义 2D 转换，沿着 Y 轴移动元素
      - scale(x,y)：定义 2D 缩放转换，改变元素的宽度和高度
      - rotate(angle)：定义 2D 旋转，在参数中规定角度
      - translate3d(x,y,z)：定义 3D 转
      - scale3d(x,y,z)：定义 3D 缩放转换
    - `animation：动画定义了动作的每一帧(@keyframes)有什么效果`
      - animation-name：规定@keyframes 动画的名称
      - animation-duration：规定动画完成一个周期所花费的时间（秒或毫秒），默认是0
      - animation-timing-function：规定动画的速度曲线，默认是’ease’
      - animation-delay：规定动画何时开始，默认是0

        - animation-iteration-count：规定动画播放的次数，默认是1
        - animation-direction：规定动画是否在下一周期逆向地播放，默认是 “normal”
11. XSS攻击和CSRF攻击的区别
    - XSS 攻击是指黑客往 HTML 文件中或者 DOM 中注入恶意脚本，从而在用户浏览页面时利用注入的恶意脚本对用户实施攻击的一种手段
    - CSRF 攻击就是黑客利用了用户的登录状态，并通过第三方的站点来做一些坏事
12. 如何防止XSS攻击
    - 服务器对输入脚本进行过滤或转码
    - 使用 HttpOnly 属性
13. es6有哪些新特性
14. 问了上机笔试做的一道选择题，关于Promise的
15. 反问
16. 还有些忘了

三面

1. 成绩排名
2. 地点选择
3. 毕业胡选择第一家公司最重要的评判标准
4. 拿了几个offer
5. 预期薪资
6. 反问
7. 还有一些忘了.....

11、京东

一面

1. 实习项目介绍
2. CSS如何实现垂直水平居中——见博客
3. Vue如何实现组件传值（父子、兄弟）
4. get和post的区别
    - get重点在从服务器上获取资源；post重点在向服务器发送数据
    - get传输数据是通过URL请求，以key= value的形式，置于URL后，并用"?"连接，多个请求数据间用"&"连接，如 **http://127.0.0.1/Test/login.action?name=admin&password=admin** ，这个过程用户是可见的；post传输数据通过Http的post机制，将字段与对应值封存在请求实体中发送给服务器，这个过程对用户是不可见的
    - get传输的数据量小，因为受url长度限制，但效率较高；post可以传输大量数据，所以上传文件时只能用post方式
    - get是不安全的，因为url是可见的，可能会泄露私密信息，如密码等；post较get安全性较高
    - get方式只能支持ASCII字符，向服务器传的中文字符可能会乱码；post支持标准字符集，可以正确传递中文字符，支持多种编码方式
        - get是按照某种编码方式（如：utf-8，gbk等）编码成二进制的字节码
        - post支持application/x-www-form-urlencoded、multipart/form-data、application/json、text/xml等编码方式

    - get请求，浏览器会主动cache，参数会被完整保留在浏览历史记录里；post中的参数不会被保留
5. http的缓存——强缓存和协商缓存
6. var和let的区别
7. 解决跨域的一些方法
8. CORS解决跨域的响应头有哪些
    - 简单请求：
        - `Access-Control-Allow-Origin`：如果Origin不在这个字段的范围中，那么浏览器就会将响应拦截
        - Access-Control-Allow-Credentials：表示是否允许发送 Cookie，对于跨域请求，浏览器对这个字段默认值设为 false
        - Access-Control-Expose-Headers：不仅可以拿到基本的 6 个响应头字段（包括Cache-Control、Content-Language、 Content-Type、 Expires、 Last-Modified和Pragma）, 还能拿到这个字段声明的响应头字段
    - 非简单请求：
        - 预检请求的响应：
            - Access-Control-Allow-Origin: 表示可以允许请求的源，可以填具体的源名，也可以填`*`表示允许任意源请求
            - Access-Control-Allow-Methods: 表示允许的请求方法列表
            - Access-Control-Allow-Credentials: 简单请求中已经介绍
            - Access-Control-Allow-Headers: 表示允许发送的请求头字段
            - Access-Control-Max-Age: 预检请求的有效期，在此期间，不用发出另外一条预检请求
        - CORS 请求的响应：
            - 它和简单请求的情况是一样的。浏览器自动加上Origin字段，服务端响应头返回Access-Control-Allow-Origin。可以参考以上简单请求部分的内容

二面（电话面）

1. 自我介绍
2. 实习项目介绍
3. 自己项目介绍
4. Vue生命周期
5. Vue和jquery的区别
6. webpack了解吗
7. 自己做的项目有进行webpack打包吗
8. 还有些忘了……

三面——挂

> 严重怀疑刷KPI

12、360

一面

1. 自我介绍
2. display有哪些属性

3. position定位有哪些
4. css如何实现垂直水平居中
5. css的弹性布局
6. js的数据类型
7. es6有哪些新语法
8. Promise有哪些状态
9. React用过吗
10. Vue的生命周期
11. VueX介绍一下
12. 如何调试页面的，定位bug
13. webpack用过吗？简单介绍一下
14. 反问

13、奇安信

1. 自我介绍
2. 用过哪些可视化的库，可以大致讲讲吗
3. String和Object的区别
4. js的作用域是什么
5. 深拷贝和浅拷贝的区别
6. 如何实现一个深拷贝，**递归和JSON实现深拷贝的区别是什么**
    - 在使用 `JSON.parse(JSON.stringify(xxx))` 时应该注意一下几点：
        - 如果obj里面有**时间对象**，则JSON.stringify后再JSON.parse的结果，时间将只是**字符串的形式**，而不是时间对象
        - 如果obj里有**RegExp、Error对象**，则序列化的结果将只得到**空对象**
        - 如果obj里有**函数，undefined**，则序列化的结果会把函数或 undefined**丢失**
        - 如果obj里有**NaN、Infinity和-Infinity**，则序列化的结果会变成**null**
        - JSON.stringify()只能序列化对象的可枚举的自有属性，如果**obj中的对象是有构造函数生成的**，使用JSON.parse(JSON.stringify(obj))深拷贝后，**会丢弃对象的constructor**
        - 如果对象中存在循环引用的情况也无法正确实现深拷贝
7. ES6有哪些特性
8. Promise简单介绍
9. 闭包是什么，有什么作用
10. 原型和原型链
11. Vue使用的设计模式有哪些
12. Vue组件是什么，使用组件的好处，组件传值
13. Http协议简单介绍一下
14. Http有哪些方法
15. patch和put的区别
16. 手写EventBus(事件总线)，on、emit、off和once
    - 如何传参
17. 反问

14、顺丰科技

一面

1. 自我介绍
2. Vue的双向数据绑定，watcher是什么
3. Vue组件传参方式
4. Vue路由原理，路由模式
5. Node你学过，可以讲讲Node吗
    - Node.js 就是运行在服务端的 JavaScript，是一个基于Chrome JavaScript 运行时建立的一个平台。
    - Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好
6. http缓存是什么
7. https和http的区别
8. webpack怎么使用，有哪些作用
9. 防抖和节流是什么，防抖函数怎么写
10. 反问
11. 其他忘了……

二面

1. 自我介绍
2. 实习内容介绍
3. 项目介绍
4. 深挖第一个项目，包括路由守卫、token处理、登录验证、token存储、token如何发送这些
5. 登录的时候，token怎么处理的
6. 项目里Vuex存储了什么内容
7. 实习内容中最难的一个点是什么
8. Jquery和Vue的区别是什么
9. http缓存是什么
10. 事件循环机制
11. 对象深拷贝和浅拷贝区别
12. 数组去重
13. 反问
14. 其他忘了...

三面

1. 自我介绍
2. 实习内容介绍
3. 这段实习经历最大的收获是什么
4. 项目中遇到的难点
5. 平时如何去解决遇到的难题
6. 如何去学习前端的
7. 学习前端多久了
8. 希望什么样的工作环境和工作氛围

9. 对工作地点有要求吗
10. 其他忘了……

15、滴滴

> 由于是连着面，有些问的内容忘了

一面

1. 自我介绍
2. 实习内容介绍
3. 项目介绍
4. token是怎么生成的，机制
5. Vue的双向数据绑定原理
6. Vue3的proxy代理和Vue2的Object.defineProperty区别是什么
7. Vue前端路由的模式有啥
8. history模式下，同一个url地址不变，为什么不会触发新的强求
9. 浏览器的渲染机制
10. 介绍一下https，加密原理
11. 你说到进程间通信了，具体的进程间通信方式有哪些
12. 同一页面的不同站点的网址为什么要开不同的渲染进程
13. 如何实现并发请求（域名分片、HTTP2多路复用、HTTP3）
14. 闭包的含义、有什么优缺点、应用场景
15. BFC是什么，有什么作用
16. 如何实现BFC
17. 其他忘了……
18. 反问：技术栈、前端氛围

二面

1. 自我介绍
2. 实习内容介绍
3. 项目里挑个有难度的讲
4. 对于用户权限，页面要怎么处理？
5. 路由守卫是干什么的
6. Vuex原理，存储的是些什么
7. ES6的新特性有哪些，箭头函数知道吗？
8. http缓存
9. 其他忘了……
10. 反问

三面

1. 自我介绍
2. 实习内容介绍
3. 自己认为哪个项目最有难度
4. 技术难度最大的一个项目——都没有难度……
5. 项目里面非前端的也挺多的，以后的职业规划
6. 对于大量的数据的展示，该怎么去处理——利用webworker去计算

7. 服务端渲染知道吗？
8. 如何提升首屏渲染速度
9. ES6的新特性有哪些
10. Promise.all是怎么实现的
11. 你觉得自己的优缺点是什么
12. 其他忘了……
13. 反问：新人培养体系

电话面

1. 口头通知offer
2. 是否有其他offer
3. 愿意去滴滴吗
4. 所属部门——**小桔车服——前端研发部**

16、东方财富

> 大前端，天天基金部门

一面

1. 自我介绍
2. 问了一点vue和其他框架的区别
3. 全程聊天

二面——拒了

17、百度

一面

1. 自我介绍
2. 兄弟div：第一个div设置margin-bottom，第二个div设置margin-top，结果会怎么样？如果两个div是包含关系（父子div）呢，结果又会如何？
3. 外层div没有高度，里面包含一个有高度的div，会出现什么情况，如何解决？
4. BFC是什么，实现BFC有哪些方法
5. 水平垂直居中，要把样式自己写出来
6. relative和absolute的区别，定位是相对于自身还是父元素
7. 浏览器渲染过程
8. js的数据类型
9. 基本数据类型和引用数据类型的区别是什么
10. 数组方法有哪些
11. some和every的区别
12. 事件循环机制
13. 原型链是什么
14. 编程：获取字符串里各个字符重复的次数
15. 编程：获取cookie字符串里指定键名对应的值
16. 其他忘了……

二面

1. 自我介绍
2. 如何学习前端知识的
3. js代码执行分为哪几个阶段
4. 上来直接编程相关
    - 字符加减

      ```
      1+"1"
      "1"+1
      "2"-1
      ```

    - 字符数组反转，不能用api
        - 题目：不用api，把["h","e","l","l","o"]转换为["o","l","l","e","h"]
        - 思路：不能用reverse()、push()、unshift()这些方法，因此考虑双指针，前面和后面的交换
    - home页面和login页面，如果进入home页面没有登录就要去login页面，如何实现这个跳转
        - 用到路由守卫
        - 如果有很多的页面，该如何去优化
    - 跳台阶
        - 看规律，知道是斐波那契数列，然后计算，但是面试官说还需要优化，可以通过算法来
    - 切黄金
        - 题目：假如你让一个工人为你工作7天，7天工资是一整根金条，今日工资是1/7根金条，工资每日现结，如果只许切两刀，应该怎么切
5. 反问
6. 其他忘了.....

三面

1. 自我介绍
2. 做的项目中，印象最深的一个项目
3. 为什么要学前端，是如何学习前端知识的
4. 职业规划是什么
5. Vue
6. 跨域是什么，如何实现跨域
7. http缓存
8. 浏览器渲染机制
9. 编程题
    - 题目：两个人拿球，交换着拿，每次一个人最多只能拿5个球（1-5个球）。如果你是第一个拿球的，要如何拿，才能保证最先拿到第99个球？
    - 想的是：倒着推，我要拿第99个球，别人可能要拿98、97、96、95、94位置的球，所以我一定要拿到93位置的球，.....
10. 反问，只能问一个问题

18、字节

一面

1. 自我介绍
2. CSS画斜线，旋转斜线
3. https具体加密过程，如何验证证书
4. 判断一个节点下面div的个数
    - 获取第一层div个数：

      ```
      let dom = document.getElementById("app")
      let num = 0
      for(let item of dom.children){
          if(item.nodeName.toLowerCase() === "div"){
              num++
          }
      }
      console.log(num)
      ```

    - 获取所有div（包括子节点div）个数

      ```
      let dom = document.getElementById("app")
      let num = 0
      function dfs(dom){
          for(let item of dom.children){
              if(item.nodeName.toLowerCase() === "div"){
                  num++
              }
              dfs(item)
          }
      }
      dfs(dom)
      console.log(num)
      ```

5. 介绍一下position定位方式
6. symbol是什么
    - Symbol是由ES6规范引入的一项新特性，它的功能类似于一种标识唯一性的ID。由于Symbol是一种基础数据类型，所以当我们使用typeof去检查它的类型的时候，它会返回一个属于自己的类型symbol
7. parseInt的第二个参数是什么
8. 常见的数组方法
9. reduce的参数有哪些
    - reducer 函数接收4个参数：
        - Accumulator (acc) (累计器)
        - Current Value (cur) (当前值)
        - Current Index (idx) (当前索引)
        - Source Array (src) (源数组)
10. localStorage如何监听变化
11. localStorage如何验证是否过期
12. webpack使用过吗，如何部署webpack

13. 编程：正则验证手机号
14. 编程：和最大的最长子序列——命名很简单，居然没写出来，害，太菜了
15. 反问
16. 其他忘了……

19、哔哩哔哩

一面

1. 自我介绍
2. 为什么学前端
3. 画一个正方形里面包含一个正方形（两个正方形仅包含边框）
4. Vue路由原理
5. 请求拦截是什么
6. token处理是什么
7. 可以简单介绍一下Vue+Element+Three.js这部分内容吗
8. canvas知道吗
9. 还知道什么图形渲染框架
10. 用过webpack吗，webpack的热更新原理是什么
11. 懒加载原理
12. package.json和package-lock.json的区别
13. git merge和git rebase
14. 反问
15. 还有些忘了……

二面

1. 自我介绍
2. 实现div里面嵌套div，并且都水平垂直居中
3. 关于Promise的打印——不难
4. 编程：防抖的实现
5. 编程：取消防抖事件
6. 编程：实现一个简单的洗牌算法
7. webpack的配置可以简单说说吗
8. CommonJS和ES module的区别
9. require(a)和require(./a)
10. require和import的区别
11. git：如何把多个commit信息合并为一个commit信息
12. git cherry-pick
13. 反问

三面

1. 自我介绍
2. 怎么学习的前端
3. 读过的前端书籍有哪些
4. 职业发展规划是什么
5. 在研究生电子设计竞赛中，你的角色是啥
6. 在数学建模比赛中，你的角色又是啥，主要负责什么工作

7. 为什么想到去**公司实习，当时都投了哪些公司
8. 实习公司的人对你的评价是什么
9. 你觉得自己是一个什么样的人
10. 服务端渲染的好处是什么
11. 反问

四面

1. 自我介绍
2. 为什么选择前端
3. 你觉得前端的发展前景会如何
4. 对于前端，你是更看重平台还是个人技术提升什么的？
5. 毕设是做什么的，现在到什么进度了
6. 一个思考题：（不断深入，寻找不同解决思路）
    - 页面上显示了5个点，鼠标从任意位置开始，拖动鼠标形成一个矩形框，判断哪些点在矩形框里？如果有10万个数据点，这样判断是不是有问题？如果这些点在x方向是有序的，有没有什么新的思路？——有序就想到二分法，如果无序呢？
7. 前端offer有几个了
8. 反问

20、海康威视

一面（电话面）

1. 自我介绍
2. 实现水平垂直居中
3. 盒模型知道吗
4. 盒模型分类
5. 实现对象的深拷贝
6. JSON实现深拷贝的缺点
7. Jquery和Vue的区别
8. Vue的生命周期
9. 要在节点渲染后处理事件，该怎么做？
    - Vue.nextTick
10. Promise的优点是什么
11. async、await了解吗
12. for var 定义的变量 实现循环 打印1,2,3,4
13. 页面登录时，服务器会怎么处理——token
14. 使用过webpack吗
15. 反问

二面——没面了

> 不想去了，就没面了

21、阿里巴巴

一面（电话面）

1. 自我介绍
2. CSS权重概念
    - 每一个CSS的选择器都有一个相对的重要程度值，也就是权重的值，简称“权值”。CSS通过CSS选择器的权重占比，来计算CSS选择规则的总权值，从而确定定义样式规则的优先级次序
3. 讲一讲flex布局的主轴和副轴的概念
    - 弹性容器中弹性子元素的排列方式
    - 主轴和侧轴是通过flex-direction确定的：
        - 如果flex-direction是row或者row-reverse，那么主轴就是justify-content
        - 如果flex-direction是column或者column-reverse,那么主轴就是align-items
4. 深入一下，比如三个元素是垂直居中的，现在想要把中间那个元素独立于其他两个元素，设置顶部对齐，该怎么做？
5. js里面闭包的概念
6. 闭包的使用场景
7. js原型链的继承
8. 原型链的最顶端是什么
9. get和post的区别
10. 有用过Vue吗，有用过React吗
11. 讲一下Vue的生命周期是什么样的吗
12. Vue的双向绑定原理
13. Vue数组改变，视图更新原理
14. Vue用了多久
15. 高阶组件的概念
    - 一个函数接受一个组件为参数，返回一个包装后的组件。在 Vue 的世界里，组件是一个对象，所以高阶组件就是一个函数接受一个对象，返回一个新的包装好的对象。类比到 Vue 的世界里，高阶组件就是 f(object) -> 新的object
16. 前端项目构建的工具
17. 简单介绍一下loader下面会用的一些loader
18. 对于js的处理一般用什么loader，比如把es6的内容转换为es5的loader
19. 能简单讲一下babel-loader里面的具体实现
20. 现在正在学习什么前端技术
21. 后面有时间，希望学习前端哪方面的内容
22. 一般的学习途径有哪些
23. 反问

二面

1. 自我介绍
2. 怎么学习前端的
3. 讲一个最难的项目
4. 目标跟踪项目是做什么的
5. http请求发送到响应结束的整个过程（其实就是浏览器渲染机制）
6. 客户端如何获取整个请求过程的时长？（包括请求发起到结束，一共花的时间，怎么去获取）
7. js垃圾回收机制
8. 除了express框架，还用过其他框架嘛，比如koa
9. 介绍一下鹰眼平台是做什么的
10. 3D可视化这块可以介绍一下吗
11. 当有大量数据渲染的时候，3D可视化显示的时候会卡顿吗，如何优化
12. 实习的最大收获是什么
13. 坚持最久的一件事
14. 反问
15. 还有几个问题忘了……

22、网易互联网

一面

1. 自我介绍
2. Vue生命周期
3. JS的垃圾回收
4. Three.js库的原理知道嘛，OPENGL如何实现绘制的？
5. Promise的作用是什么
6. Promsie里为什么用异步setTimeout
7. 反问
8. 其他忘了……

二面

1. 自我介绍
2. 组件的好处，在设计一个组件时应该考虑哪些问题
3. 写一个组件，用于实现下拉框作用——很久不写组件了，瞎写了一下
4. 组件传参时该怎么传，会怎么考虑
5. 父子组件传递参数时，怎么操作
6. 组件的slot知道吗——知道，研一用过，现在忘了，好像是插槽、内容分发作用
7. token处理是怎么处理的
8. token验证的具体原理是什么
9. 会React吗——不会，没学过
10. 编程：二叉树的层次遍历——easy
11. 反问

三面

1. 自我介绍
2. 为什么学习前端

3. 你们学习的课程和科班的课程相同吗
4. 学习前端的路径是什么
5. 讲讲你觉得难度最大的一个项目
6. 手上有哪些offer了
7. 反问
    - 网易的新人培养体系

23、触宝——现场面试

一面

1. 自我介绍
2. js基本数据类型
3. es6的新语法
4. 事件循环机制
5. 垃圾回收
6. 实习内容
7. 如何学的前端
8. 遇到困难如何解决
9. 项目难点是什么
10. 还有好些，忘了.....

二面

1. 自我介绍
2. 排序算法有哪些
3. 冒泡排序的时间复杂度，最好情况和最坏情况
4. 手写一个冒泡排序
5. 手写代码：[{"ID":1001,"SCORE":100},{"ID":1002,"SCORE":95},{"ID":1001,"SCORE":80}]，统计数组里各ID分组得分之和——用哈希表法
6. 做的项目难点
7. get和post的区别
8. http1.0和http2.0的区别
9. http和https的区别
10. 闭包的含义，闭包的使用场景、优缺点
11. 如果与产品经理分歧很大，该怎么解决
12. 对加班怎么看
13. 反问
14. 其他也忘了.....

三面

1. 自我介绍
2. 为什么学习前端
3. 学习前端多久了
4. 对前端感兴趣嘛
5. 职业规划是什么
6. 手里有几个offer了

7. 期望薪资是多少
8. 你对触宝了解吗
9. 为什么选择上海，不去北京了嘛
10. 反问

四面——视频加面

1. 自我介绍
2. 实习内容详细介绍
3. 为什么选择前端
4. 怎么学习前端的
5. 项目具体怎么部署的
6. 你自己的优缺点是什么
7. 如果有一个各方面比你差，但是业务能力比你强的人，你会是什么感受
8. 以后的职业规划是什么
9. 反问
10. 还有些忘了……