# 二面

考察代码能力，解决问题的能力，可根据其项目经历和自己擅长的领域问答

## js

- 如何实现一个倒计时
  - 从后端获取倒计时终点， 通过 setInterval 倒数
  - 但是 setInterval 会不准，如何解决
  - 答： 事件循环， 其他程序占用了线程导致延时， 需要和系统事件对比， 通过 new Date() 矫正
  - 事件循环， 如下代码打印结果及原因， 1,4,6,5,2,3

```
console.log(1);

setTimeout(() => console.log(2), 1)
setTimeout(() => console.log(3))

new Promise((resolve) => {
  console.log(4)
  resolve()
}).then(() => {
  console.log(5)
})

console.log(6)
```

- 实现一个时间格式化函数 timeFormat(timestamp, fmt)
  - 合格： timestamp 转日期，然后用 getYear(), getMonth() 等等来获取， 然后替换掉
  - 考察正则， string 方法
  - 参考 uc-fun timeFormat
- 如何实现队列（先进先出)

```
function Queue(){
  var items = [] ;
  this.enqueue = function(element){}
  ……
}
```

  - enqueue-添加新项，利用push
  - dequeue-移除队列第一个元素 ，利用shift
  - front－获取第一个元素，利用data[0]
  - isEmpty—是否为空，利用data.length == 0

  - size —队列元素数量，利用data.length
- 写一个通用方法将n维数组展平。
- 解析get带参数请求为json串（期间可考察代码质量，如对空值的判断，实现的复杂度）

# vue框架使用

- Vue 的父组件和子组件生命周期钩子执行顺序是什么
  - 加载渲染过程：父（beforeCreate）=> 父（created）=> 父（beforeMount）=> 子（beforeCreate）=> 子（created）=> 子（beforeMount）=> 子（mounted）=> 父（mounted）
  - 父组件更新过程：父（beforeUpdate）=> 父（updated）
  - 子组件更新过程：父（beforeUpdate）=> 子（beforeUpdate）=> 子（updated）=> 父（updated）
  - 销毁过程：父（beforeDestroy）=> 子（beforeDestroy） => 子（destroyed）=> 父（destroyed）
- 使用过 vue 框架的哪些功能
  - data/watch/computed/filter/method/directive 等
  - watch/computed/filter/method 有什么区别
  - 实现一个打点统计的功能，点击某个按钮，或任一标签，向后端接口发送一个请求，请求数据

```
{
  event: '事件名',
  // 参数列表
  params: {
    key: 'value',
  }
}
```

    - 基础： 在待统计标签上绑定 @click 事件， 然后调用接口数据
    - 中级： 写自定义指令， `v-track:event="{ [key]: [value] }"`

```
/**
 * 打点统计
*/
export default {
  // bind 中绑定 click 事件监听
  bind(el, { value: args, arg: event }) {
    el.handler = () => {
      console.log(`statistic track: event = ${event} args =`, args);
    };
    el.addEventListener('click', el.handler);

    // touch for mobile
    // el.addEventListener('touch', el.handler)
  },
  // unbind 移除事件监听
  unbind(el) {
```

```
    el.removeEventListener('click', el.handler);
    delete el.handler;
  }
};
```

- 实现一个组件 `<Title :level="1|2|3|4|5">text<Title>` 最终输出 `h1, h2, h3, h4, h5` 及其内容
  - 基础： 多个 v-if 来实现
  - 高级： 用 jsx 实现

# react 框架使用

- 类组件(Class Component)和函数式组件(Functional Component)有什么区别
- 什么是高阶组件，使用场景有哪些
- 为什么setState不会同步更新组件的状态
- 循环遍历渲染时为什么要绑定key值，引出react性能优化方案（shouldComponentUpdate，使用key来帮助React识别列表中所有子组件的最小变化）
- react hooks是什么，常用的有哪些，分别应用于什么场景
- 实现一个权限校验的功能， 有权限显示此代码块内容， 无权限不显示
  - 基础： 写一个权限校验函数， 通过 if else 实现

```
// check function
const checkPermission = (code) => { // todo check permission and return
true|false  }

// render
render() {
  return (
    <div>
      {checkPermission && <div>something</div>}
    </div>
  )
}
```

  - 高级： 封装一个组件，在需要做校验的地方套一层组件

```
// Permission
// get permission list from somewhere
import { permission } from 'somewhere';
const Permission = React.memo(({ code, children }) => {
  if (code in permission) {
    return children;
  }
  return null;
```

```
})

// render
render() {
  return (
    <div>
      <Permission code={permissionCode}>
        something
      </Permission>
    </div>
  )
}
```

- 实现一个组件，当用户登录后才能显示此组件，登录再其他组件中实现
  - 当前组件获取登录状态
  - 高阶组件统一处理
- 如果用过antd，可以问这个比较经典的问题：Antd实现自定义form组件，如何传递组件变量给form表单

# react VS vue； 区别，优劣势； 讲的合理就好

- react 更加自由，贴近 JavaScript 写法，更加灵活；但是代码看着脱离 html 原本的书写方式，jsx 看着比较丑；对于使用者刚开始适应起来较恶心，同时要求使用者能够很好的组织代码逻辑，划分代码块
- vue 作者做的很贴心，帮助开发者做了很多友好的东西，包括模块划分、书写方式、devtool、社区；开发者只要根据作者提供的这些东西就能快速上手开发

# 浏览器性能优化

- 减少资源加载体积；
  - 压缩图片、js、css；
    - 如何压缩图片，压缩到什么程度是比较合适的； 图片格式有什么区别
    - 答： 通过lighthouse or pagespeed 检测，然后找个工具压缩到指定尺寸； jpg/jgeg/png/webp/svg 区别，如何选择；
    - js/css 通过webpack uglify/mincss 等可以实现
  - 开启gzip； gzip 需要压缩哪些资源，为什么，如何配置
    - 压缩js、css、svg； 图片不要使用gzip，尺寸会变大
  - 开启缓存； Etag, last modified, cache-control； 哪些资源要加缓存，哪些不加缓存
  - 分离静态资源和接口资源； 接口资源通常会在header里携带 cookie/Authentication 等数据，这些信息是静态资源不需要的
  - 按需加载，尽量减少首屏加载需要的资源；
- 浏览器渲染
  - 少操作dom
  - 避免浏览器重绘与回流
    - 重绘与回流的概念，如何避免

    - 浏览器分层渲染， 将容易出现重绘或回流的标签单独成层去渲染
    - css 放在 header, js 放在 body 最后
  - 减少 dom/css 嵌套写法
- 回流（重排）和重绘是什么
  - **回流**：当渲染树中的一部分（或全部）因为元素的尺寸、布局、显隐发生改变而需要重新构建，就是回流。回流后会进行重绘。
  - **重绘**：当只是元素的外观、风格变化，不影响布局的，重新渲染的过程就叫重绘。
  - 回流必将引起重绘，而重绘不一定会引起回流。每个页面至少回流一次，就是在页面第一次加载的时候。
  - 回流何时发生
    - 添加或者删除可见的DOM元素
    - 元素的位置发生改变
    - 元素的尺寸发生改变--边距、填充、边框、宽高
    - 内容改变--比如文本改变或者图片大小改变而引起的计算值宽高的改变
    - 页面初始化渲染
    - 浏览器窗口尺寸改变
  - 如何减少回流和重绘
    - 使用cssText或者className一次性改变属性
    - 使用document fragment
    - 对于多次重排的元素，如动画，使用绝对定位脱离文档流，使其改变不影响其他元素。
  - 性能优化
    - 怎样测试浏览器运行内存，判断是否有内存溢出。
    - 是否进行资源压缩合并以减少http请求，非核心代码异步加载，异步加载的方式是什么（import（））
    - 可利用浏览器缓存，引出缓存的分类，缓存的原理（亦可引出http状态码相关问题，如：304状态码代表什么，什么是强制缓存（expires，cache-control，max-age），协商缓存（last-modified，etag））
    - 什么是dns预解析

# 网络相关

- http协议主要特点（无状态，简单快速，无连接，灵活）
- http报文的组成部分，请求报文，响应报文包含什么
- http状态码，几种方法，post和get区别
- 持久连接，keep-alive模式，非keep-alive模式含义。管线化的含义。
- 安全
  - CSRF是什么，基本概念，攻击原理，如何防御
  - XSS是什么，概念，如何防御

# 可视化方向

- 由于可视化的图表插件太多，所以得问他用过哪些。
  - 用过图表插件没（类似echarts）？调研过哪些插件呢？当时为什么选它，说下的自己的观点？
  - 回答的有道理即可。
- 像百度高德这样的地图的api用过吗？有没有想过如果我要从高德切换到百度的gis是不是很麻烦，以前的代码基本用不上了？考验对方能否说出leaflet。

- 用过leaflet吗，能否说下他的工作原理，他有哪些优势?
- 有没有遇到过，如果一个大屏的东西太多（比如有好多图表，还有带动效的），然后导致这个大屏很卡，有没有解决方案?
- 有做过3D可视化吗？说下你当时的实现过程，都用了哪些东西实现的，有遇到什么问题吗？怎么解决的?

# git flow

- 命令了解， add, commit [--amend], merge [--no-ff], pull, rebase, push, reset、remote [add|set-url|remove]、cherry-pick
- 版本控制
- 多人协作
- cicd: 通常处理哪些流程，如何实现
  - gitlab-runner
  - travisci,
- merge request/pull request 机制
- subtree