2023 年珠峰前端架构课

课程简介

本课程主要面向 1 年以上工作经验的前端开发工程师，来自 BAT 等一线互联网大厂的讲师队根据自己的实战、面试和晋级经验，精心打磨出适合前端进阶的成长路径，以摸底分析、直播互动、课后作业、讨论答疑和多人协作项目和简历和面试指导等方式，帮助学员在短时间内晋级高级前端工程师，加入我们，共同开启一段卓越的职业之旅吧!

珠峰教育简介

- `前端老字号` 珠峰教育成立于 9 年，14 年来，珠峰只做前端、做精前端，早在 15 年就开设创了珠峰前端架构课
- `课程体系` 作为行业的领头羊，我们始终关注最前沿的前端技术，从数万名学员中提炼最需要的核心技术，不断升级迭代课程内容和服务
- `顶尖师资` 讲师团队均来自 BAT 等一线互联网大厂，十八年全栈开发经验、十年以上的授课经验的技术大佬领衔授课，，不但有顶尖的技术，更有丰富的教学经验
- `重原理和源码` 不但注重项目实践，更注重前端架构、原理和源码，让你可以轻松手写前端各种主流框架
- `亮点项目` 具备多个重量级前端亮点项目，比如组件库、监控系统、持续集成 CICD 和低代码平台
- `高质量学习社群` 数万学员遍布于各大互联网企业，并形成了一个紧密团结而且互助有爱的前端高质量社群，内推和就业资源丰富

适合人群

- 基础功薄弱，只会使用工具而不知道底层原理
- 缺乏大型的项目架构经验和项目亮点
- 希望进入大厂，但缺乏简历编写和面试技巧，缺乏内推机会

本版更新

前端权限系统设计与实现

- 角色权限设计与概念
- 前端菜单的权限控制
- 前端页面的权限控制
- 前端按钮的权限控制
- 前端接口权限控制
- Vue3 权限控制实现
- React18 权限控制实现

Electron 实战训练营

- Electron 介绍和环境搭建
- Electron 应用协议唤起 —— 桌面掘金
- Electron 集成 Node.js
- Electron 进程通信
- Electron 进程详解
- Electron 自定义窗口 —— 桌面时钟
- Electron 系统菜单
- Electron 系统托盘
- Electron 自定义托盘 —— 托盘计算器
- Electron 中的 webview —— 简易浏览器
- Electron 对话框
- Electron 中的快捷键
- Electron 读取剪贴板
- Electron 网络拦截
- Electron 应用内协议
- Electron 会话管理
- Electron 电源状态管理
- Electron 与其他应用通信
- Electron 中的 asar 文件
- Electron 跨平台打包
- Electron 应用升级
- Electron 崩溃分析

从零实现 Pinia 最新版

- 掌握 Pinia 设计思想，Pinia 及 Vuex 对比
- 掌握 Pinia 的核心使用方法
- 实现 optionsStore 及 setupStore 两种定义方式
- 实现$state、getters、$patch、$reset 等方法
- Pinia 中插件系统的实现，实现持久化插件
- 实现辅助函数 storeToRefs 等

前端跨域解决方案大汇总

- 同源策略和跨域的定义
- JSONP 实现跨域
- CORS 跨域资源共享
- websocket 实现跨域
- postMessage 实现跨域

- Nginx 和 Node 中间层跨域
- iframe 实现跨域

Vue2 & Vue3 diff 算法剖析

- 深入理解 Vue 的虚拟 DOM 和 diff 算法
- 手写 Vue2 的 diff 算法，掌握 diff 算法的实现原理
- 掌握 diff 算法中 key 的关键作用及引发的问题
- Vue2 vs Vue3：深度对比 diff 算法的进化之路
- Vue3 的 diff 算法优化策略：探讨 Block Tree 和 PatchFlags 在 diff 算法中的巧妙应用

手写一个自己的前端脚手架

- 掌握脚手架设计思想和开发流程
- 搭建脚手架项目结构,实现命令行开发与参数解析
- 实现远程拉取项目模板以及模板编译生成项目结构。
- 完善脚手架常见功能：命令行选择、Loading 处理等
- 通过 npm 发布自己的脚手架

实现专业级 Calendar 日历组件

- 利用 TS 定义组件所需属性及事件
- 深度掌握 Vue3 属性传递和事件处理
- 掌握日历组件核心设计思想，实现日历组件
- 实现自定义日历组件内容，可控制日历显示范围及日历头部功能

Webpack5 模块联邦与微前端 EMP2 实践

- Webpack5 模块联邦和微前端原理
- EMP2 架构、核心组件和框架的集成
- EMP2 基座应用、路由管理、加载策略和通信
- EMP2 子应用生命周期管理和协同
- EMP2 性能优化和缓存策略

2023 微前端实战源码训练营

- 微前端的概念和优势
- 各种微前端架构实现方案
- single-spa 微前端实战
- 从零实现 single-spa
- 详解 JS 沙箱及 CSS 样式隔离方案
- qiankun 整合 Vue、React 项目
- 落地微前端的问题和解决方案
- 实现 webpack5 模块联邦和 EMP2

Vue3+TS 实现 UpLoad 组件

- 利用 TS 定义组件所需属性及事件
- 深度掌握 Vue3 中属性传递和事件
- 掌握上传组件的设计思想，实现上传功能
- 实现类型和大小限制、支持照片墙、自定义缩略图、图片列表展示、上传文件列表控制等
- 实现手动上传和拖拽上传图片功能

zustand

- zustand 的概念和优势
- zustand 项目实战
- 实现创建仓库、读取状态、更新状态
- 实现状态分享，计算属性和事件处理
- 实现日志和持久化中间件

BFF

- BBF 的应用场景和架构设计
- Node 搭建高性能 BFF 服务
- 使用 RPC 协议提升性能
- 使用 Redis 实现高速缓存
- 使用异步消息队列 MQ 实现海量任务处理

RxJS

- RxJS 中的 Observable、Observer、Subscription、Operators、Subject 等核心概念讲解
- RxJS 中的弹珠图等可视化工具
- RxJS+React Hooks 异步接口请求和防抖节流实战
- 实现 Observable、Observer、Subscription
- 实现防抖节流等 Operators 和 Subject
- mergeMap,switchMap、concatMap、mergerAll 和 concatAll 等
- fromFetch、share、debounceTime、takeUntil 和 withLatestFrom 等
- RxJS 封装 http 请求功能
- RxJS 控制异步任务最大并发数
- RxJS 实现 AutoComplete 组件
- RxJS 实现拖拽功能封装
- observable-hooks 实战和原理

UMI4

- Umi4 新功能和新特性
- Umi4 设计和架构
- Umi4 技术栈选型
- Umi4 开发最佳实践
- Umi4 Max 开发最佳实践

Pinia2

- 掌握 pinia 设计思想，以及和 vuex 对比
- pinia2 实战开发
- 实现 optionsStore 和 setupStore
- 实现$state、getters、$patch、$reset、$subscribe、$onActions、$dispose
- 实现插件系统和持久化插件
- 实现辅助函数 storeToRefs、mapState、mapWritableState 和 mapActions

Vite3+Vue3 工程化

- Vite3 新特性讲解
- Eslint + Prettier + Husky 配置
- 路由和 pinia 状态管理
- tailwindcss 和 UnoCSS 实战
- 使用 Vitest 编写单元测试
- CICD 持续集成

Vue+TS 实现复杂表单组件

- 利用 TS 定义组件所需属性及事件
- 掌握表单组件的设计思想及组件间数据通信处理
- 实现 Form、FormItem 等组件
- 实现表单组件的数据绑定、校验规则处理、错误显示
- 使用 async-validator 实现对表单的校验功能

Vue-Loader

- webpack5+vue-loader15 的实战
- webpack loader 原理的深入分析
- 实现对.vue 文件的编译和处理
- 实现对 template、script、style 的编译和处理
- 实现 CSS Scoped 和 CSSModules

实现 ReactHooks

- 实现 useState
- 实现 useCallback 和 useMemo
- 实现 useReducer 和 useContext
- 实现 useEffect 和 useLayoutEffect
- 实现 forwardRef 和 useImperativeHandle

课程大纲

- 前端基础
- Node.js
- 前端工程化
- Vue.js
- React.js
- 多端开发
- 数据可视化
- 后端开发
- 解决方案
- 就业指导
- 实战训练营
- 计算机基础

前端基础

JavaScript

- 栈和队列
- JS 数据类型
- 执行上下文栈
- 执行上下文生命周期
- 作用域和作用域链
- 闭包
- var 和 let
- this 指针
- 原型链和继承
- ES6 新特性
- let&const
- 解构赋值
- 模板字符串
- 箭头函数
- 展开运算符

- 数组和对象新方法
- 类的定义
- 类的的继承
- 微任务和宏任务
- 事件环

ECMAScript6

- let&const
- 解构赋值
- 模板字符串
- 箭头函数
- 展开运算符
- 数组的方法
- 对象
- 继承
- 类的编译
- 类的继承 br
- 异步发展流程
- promise

Typescript

- 基本数据类型
- 函数和双向协变
- 类的定义和装饰器
- 抽象类和重写重载继承多态
- 接口和泛型
- 类型保护、推断、变换
- 条件类型和工具类型
- Proxy、OverWrite 和 Merge
- 模块和命名空间
- 模块和命名空间
- 类型声明和扩展
- leetcode 面试题

Typescript 实现 Axios

- 实现 GET 和 POST 请求
- 实现错误处理
- 实现拦截器功能
- 实现配置合并

- 实现转换请求与响应
- 实现实现任务

设计模式

UML 模型图

- 用例图
- 类图和对象图
    - 依赖关系(Dependence)
    - 泛化关系(Generalization)
    - 实现关系(Implementation)
    - 关联关系
    - 聚合关系
    - 组合关系
- 活动图
- 时序图
- 协作图
- 组件图
- 部署图

面向对象

- 什么是面向对象
- 封装、继承和多态
- 面向对象和面向过程

设计原则

- 开放封闭原则
- 单一职责原则
- 里氏替换原则
- 依赖倒置原则
- 接口隔离原则
- 迪米特法则
- 合成复用原则

创建型模式

- 抽象工厂模式(Abstract Factory)
- 建造者模式(Builder)
- 工厂方法模式(Factory Method)
- 原型模式(Prototype)
- 单例模式(Singleton)

结构型模式

- 适配器模式(Adapter)
- 桥接模式(Bridge)
- 组合模式(Composite)
- 装饰模式(Decorator)
- 外观模式(Facade)
- 享元模式(Flyweight)
- 代理模式(Proxy)

行为型模式

- 职责链模式(Chain of Responsibility)
- 命令模式(Command)
- 解释器模式(Interpreter)
- 迭代器模式(Iterator)
- 中介者模式(Mediator)
- 备忘录模式(Memento)
- 观察者模式(Observer)
- 状态模式(State)
- 策略模式(Strategy)
- 模板方法模式(Template Method)
- 访问者模式(Visitor)

网络安全

- 反射型和存储型 XSS
- XSS 防御
- Web 编码和转义
- CSRF 攻击
- 验证码防御 CSRF
- refer 验证防御 CSRF
- token 验证防御 CSRF
- xss+csrf(蠕虫)
- DDOS 攻击
- HTTP 劫持

AST 抽象语法树

- 抽象语法树用途
- 抽象语法树定义
- JavaScript Parser
- AST 节点和遍历

- babel 工作原理
- 实现转换箭头函数
- 实现把类编译为函数
- 实现自动埋点插件
- 实现自动日志插件
- 实现 eslint 功能插件
- 实现 uglify 功能插件
- 实现 webpack 按需加载插件

函数式编程

- 函数式编程定义与优势
- 闭包(closure)
- 纯函数
- 柯里化函数
- 组合柯里化
- 函子

JWT(JSON Web Token)

- JWT 的结构
- JWT 实战
- JWT 工作原理

Websocket

- HTTP 的架构模式
- 双向通信
- EventSource 流
- websocket 实战
- 实现 websocket 协议

防抖节流

- 防抖工作模式
- 实现防抖参数传递
- 实现防抖立即执行
- 实现防抖取消任务
- 实现防抖获取返回值
- 节流工作模式
- 实现节流 leading
- 实现节流 trailing
- 实现节流取消和返回值

Lighthouse 和 Performance 性能分析与优化

- 浏览器渲染过程全流程分析
- LightHouse 性能分析和优化方案
- Performance 概览面板 FPS、CPU、网络、内存分析
- 性能指标主线程、合成线程、GPU 线程和 IO 线程分析
- 各种类型性能指标的详情面板

浏览器渲染原理

- 浏览器渲染流程
- 为什么 css 放上面 js 放下面
- 模拟浏览器解析流程
- 优化策略
- V8 的编译过程
- 什么是垃圾
- 实现执行上下文栈
- 实现闭包
- V8 的垃圾回收机制
- V8 的性能优化
- 接收 HTML 内容
- 计算节点的样式
- 页面的绘制
- lighthouse
- 词法环境
- 准备全局执行上下文和绑定初始化
- 开始准备执行函数代码
- 执行函数代码
- 什么是 closure
- this 指向

前端面试

- call 和 apply 的区别
- ES6 转 ES5 的实现思路
- XHR 具体底层原理和 API
- prototype 底层原理
- parse 函数
- 数组扁平化方法
- 实现不可变对象
- 异步实现并发请求

- 防抖和节流
- Reflect Proxy
- 控制最大并发数
- JS 模块化
- promise、async await、Generator 的区别
- 如何让 (a == 1 && a == && a == 3) 的值为 true?
- arrayBuffer 和 Buffer 区别
- 函数柯理化
- 拷浅贝和深拷贝
- console.log(==!);
- +号重载

模块化方案

- 自执行函数
- AMD
- CMD
- CommonJS
- UMD
- ESM

浏览器事件环

- 进程和线程
- Chrome 浏览器进程
- 渲染进程
- Eventloop 使用
- EventLoop 实现
- requestAnimationFrame
- requestIdleCallback
- Node 中的 EventLoop
- EventLoop 面试题

大文件上传和断点续传

- 整体上传
- 分片上传
- 进度条
- 秒传
- 断点续传(支持单个分片续传和刷新浏览器续传)
- 暂停和恢复

跨域方案

- 什么是同源策略及其限制内容
- 常见跨域场景
- Jsonp
- CORS
- PostMessage
- Websocket
- Node 中间件
- Nginx 反向代理
- Window.name + iframe
- Location.hash + iframe
- Document.domain + iframe

不可变数据

- 共享可变状态
- 不可变数据的解决方案
- 实现 immer
- 实现 produce
- 实现 useImmerState

V8 的垃圾回收机制和内存泄露分析

- V8 的内存分配和管理
- 垃圾回收 GC 常见算法和优劣分析
- V8 中的垃圾回收策略详解
- Timeline 内存快照监控
- Performance 内存泄露分析
- 有效的针对 V8 性能优化方案

V8 的内存管理和性能优化

- V8 内存区域管理
- V8 不同区域的垃圾回收策略
- V8 垃圾回收性能优化方案
- Performance 和 Timeline 性能分析
- 如何编写高性能前端应用

V8 垃圾收集

- V8 的内存垃圾回收
- 计算机模型之寄存器
- 内存和指令
- 解释型和编译型语言
- V8 执行过程

V8 内存管理

- JavaScript 中的垃圾收集
- JavaScript 中的内存管理
- V8 垃圾回收机制分类
- 引用计数、标记清除、标记整理和增量标记

前端数据结构和算法

算法的基础知识

- 输入、输出和数量级 | 计算能力的变革
- CPU、寄存器和内存 | 二分查找 | 插入排序 | 冒泡排序

算法的衡量和优化

- 时间复杂度和空间复杂度 | 复杂度的本质
- 合并排序 | 递归函数复杂度分析
- 递归表达式分析法 | 递归数学归纳法分析
- 主定理

排序算法

- 排序算法介绍
- 基于比较的排序算法
- 合并排序优化
- 快速排序
- 快速排序复杂度和优化
- 计数排序
- 基数排序
- 桶排序
- 外部排序

递归

- 递归的基本概念
- 递归图形的绘制
- 递归和穷举问题

- 组合问题
- 递归空间优化
- 回溯算法
- 重复子问题优化
- 尾递归
- 搜索问题(8 皇后)
- 深度优先搜索和广度优先搜索

数据结构

- 数组
- 双向链表
- 反转单向链表
- 堆
- 栈
- 队列

进阶算法

- 动态规划的概念
- LCS 问题的子结构
- 填表法
- 构造结果

BAT 面试真题

- 反转二叉树
- 解析 Query 字符串
- 取 N 个数字为 M
- 火车排序问题和队列
- 网格走法动态规划
- 两个栈实现一个队列

Node.js

Node.js 核心

- EventLoop 和事件队列
- process 全局对象
- events 事件处理模块
- commonjs 原理解析
- 深入字符编码
- Buffer 对象
- fs 文件模块

- 压缩与解压缩
- 加密和签名算法
- stream 流的原理和应用
- 多进程与集群
- tcp 和 http 服务
- cookie 和 session 原理
- 多语言、防盗链、正向和反向代理服务器

Express

- express 路由配置
- express 处理参数
- express 使用中间件
- express 模板引擎
- express 静态文件服务器
- express 重定向
- cookie 和 session 原理
- 1 比 1 手写 express 框架
- 1 比 1 手写 koa2 框架
- JWT 权限认证原理

进程线程集群

- 进程
- 子进程
- cluster 集群

Egg.js

- 项目架构
- 配置路由
- 静态文件中间件
- 模版引擎
- 远程接口服务
- 计划任务
- 集成 MYSQL
- Restful 接口
- Sequelize 持久化工具
- 国际化
- 扩展工具方法
- 中间件
- 运行环境

- 单元测试
- 服务器部署和运维
- 手写自己的 Egg.js 框架，包括 egg-core、egg-init、egg-cluster
- 自定义插件和框架 ,手写 egg-socket.io 插件

Nest.js 实战

- Nest.js 核心概念
- TypeORM 集成使用
- 文件上传
- 微信支付和支付宝支付
- AntDesign 实现权限用户后台管理系统

Nest.js

- ReflectMetadata
- 控制反转和依赖注入
- Nest.js 入门
- 自定义 Token
- 实现服务的注册
- 实现值的获取
- inject 的 useValue 和 useFactory 实现
- Inject
- param
- 实现 decorate
- 实现 inject
- Injectable

前端工程化

Webpack4

- webpack 基础配置
- webpack 打包出的文件解析
- Html 插件
- 样式处理
- 转化 es6 语法
- 处理 js 语法及校验
- 全局变量引入问题
- 图片处理
- 打包文件分类
- 打包多页应用

- 配置 source-map
- watch 的用法
- webpack 小插件应用
- webpack 跨域问题
- resolve 属性的配置
- 定义环境变量
- 区分不同环境
- noParse
- IgnorePlugin
- dllPlugin
- happypack
- webpack 自带优化
- 抽离公共代码
- 懒加载
- 热更新
- tapable 介绍
- tapable
- AsyncParralleHook
- AsyncSeriesHook
- AsyncSeriesWaterfall
- loader
- loader 配置
- babel-loader 实现
- banner-loader 实现
- 实现 file-loader 和 url-loader
- less-loader 和 css-loader
- css-loader
- webpack 流程介绍
- webpack 中的插件
- 文件列表插件
- 内联 webpack 插件
- 打包后自动发布

Webpack5 新特性

- 持久化缓存
- 资源模块
- moduleIds & chunkIds 的优化
- 更智能的 tree shaking

- nodeJs 的 polyfill 脚本被移除
- 支持生成 e6/es2015 的代码
- SplitChunk 和模块大小
- Module Federation

Webpack5 核心

- webpack 基本概念
- webpack 基础使用
- webpack 打包后文件的分析
- 异步加载代码

Webpack5 工作流

- 调试 webpack
- webpack 编译流程
- 实现简版 webpack

实现 AsyncQueue

- 异步队列工作原理
- 实现 AsyncQueue

WebpackLoader

- loader 运行的总体流程
- loader 的分类
- pitch
- 特殊前缀配置
- 实现 loader-runner
- 实现 babel-loader
- 实现 style-loader

css-loader

- 实现 css-loader
- 支持 esModule 参数
- PostcssCSS 语法树
- 支持 url 参数
- 支持 import 参数
- 支持 importLoaders 参数
- 支持 modules 参数

Vue-Loader

- webpack5+vue-loader15 的实战
- webpack loader 原理的深入分析
- 实现对.vue 文件的编译和处理
- 实现对 template、script、style 的编译和处理
- 实现 CSS Scoped 和 CSSModules

px2rem-loader

- 设备物理像素和独立像素
- 移动端适配
- px2rem-loader 实战
- 使用自定义 loader
- CSS 抽象语法树
- 实现 px2rem-loader
- 实现 lib-flexible
- 解决第三方框架样式问题

Tapable

- tapable 分类
- 实现 SyncHook
- 实现 AsyncParallelHook
- 实现 interceptor
- 实现 HookMap
- 实现 stage
- 实现 before

Webpack 插件

- webpack 插件概念
- Compiler 和 Compilation
- 同步和异步插件
- zip 打包插件
- 自动外链插件

DllPlugin

- 使用 DllPlugin
- 使用 DllReferencePlugin
- 生成 dllutils
- 介绍 DllReferencePlugin
- 实现 DllReferencePlugin

SourceMap

- sourcemap 的实现原理,手写实现算法
- webpack 中的 source-map 详细配置和最佳实践
- 生产环境里无 source-map 如何线上调试
- source-map-loader 详解

HMR

- 实现 webpack 开发中间件
- 使用 HMR
- 启动 ws 服务器
- 打包后的模块分析
- 连接 socket 服务器
- 创建父子模块的关系
- 实现热更新
- 文件的生成过程

Webpack 性能优化

- 项目初始化
- 打包的数据分析
- 增加查找速度
- 配置环境
- treeshaking
- scope-hosting
- babel-runtime

Webpack 代码分割

- 入口点分割
- 动态导入和懒加载
- 提取公共代码

Webpack 面试题

- 构建工具选型
- webpack 如何调试
- 有哪些常见的 loader 和 plugin
- 代码分割
- hash
- 优化打包速度
- 如何编写 loader
- webpack 打包的原理

- tree-shaking
- HRM 热更新

从零实现 Webpack

- 实现 Compiler 的 run 方法
- 实现 Compiler 和 make 钩子
- 分析对象之间的关系
- 复习上节课的流程
- 实现 buildModule
- 实现 Stats
- 获得依赖的模块 ID
- 递归编译模块
- 实现 seal 封装 Chunk 代码块
- 实现 emit 功能
- 支持懒加载
- 加载第三方模块
- splitChunks
- 代码分割
- runtime 运行原理
- 实现自己的 splitChunks
- 支持 loader-runner
- 三个 hash 值
- treeshaking
- preload 和 prefetch

Webpack5 中的模块联邦

- Module Federation 工作原理
- Module Federation 实战
- 通过 shared 处理公共依赖
- 处理双向依赖
- 支持多个 remote
- 实现 Module Federation

实现 CreateReactApp

- create-react-app 源码调试
- 实现 init 方法
- 实现 createApp 方法
- 实现 run 方法

实现 react-scripts

- react-scripts 源码调试
- 实现 build 命令
- 实现 start 方法

VueCli V1

- commander 的用法
- 拉取模板
- 下载资源

VueCli V4

- @vue/cli4.0 源码调试
- 实现参数解析
- 实现脚手架的插件系统
- 实现 create 命令
- 实现@vue/cli-service 插件

Create-vite

- 配置执行命令
- 编写配置命令
- 实现子进程执行配置命令
- 实现 create 命令
- 完成 create 命令
- 实现文件的拷贝
- 实现插件机制

Lerna

- 搭建 npm 私服
- 编写单元测试
- 支持 eslint 和 prettier
- 支持 git hooks
- 发布上线和安装命令
- 实现 init 和 create 命令

Rollup

- rollup 项目实战
- AST 和作用域分析
- 实现 rollup
- 实现 tree-shaking

- 实现变量重命名

Vite V1

- lerna 创建项目
- 启动并调试
- 实现静态服务
- 重写导入路径
- 解析 vue 文件
- 编译 vue 模板
- 支持样式

Vite V4

- 配置开发环境
- 静态资源处理
- 配置别名
- 样式处理
- typescript
- 配置代理
- 配置 mock
- 配置 ESLint
- 配置 Prettier
- 配置单元测试
- 配置 git hook
- Vite 的 HMR 热更新
- Vite 的 SSR 服务器端渲染
- 实现命令行
- 实现 HTTP 服务器
- 实现静态文件中间件
- 分析第三方依赖
- 预编译并保存 metadata
- 修改导入路径
- 支持 vue 插件
- 支持 style
- 支持环境变量
- 支持 HMR

Gulp

- gulp 工程化实战和核心 API
- 样式编译 、JS 和 TS 编译 、模板编译 、图片和字体转换、文件压缩和删除、开发服务器和监听等工程化流程
- 从零实现 gulp 和核心 API
- 从零实现组合任务、异步任务和文件操作
- 从零实现 gulp 常用核心插件

Polyfill

- babel-polyfill
- useBuiltIns
- babel-runtime
- babel-plugin-transform-runtime
- 最佳实践

ESLint

- AST 抽象语法树的概念和基本原理
- AST 抽象语法树的遍历和生成
- ESLint 引擎和规则的运行原理
- ESLint 插件的基本工作原理
- 实现实用的 ESLint 插件

Esbuild

- Esbuild 安装和使用
- Esbuild 内容类型
- 编写 esbuild 的 plugin
- 命名空间
- 过滤器
- Resolve 参数和回调
- onLoad 参数和回调

Vue.js

Vue2 应用

- Vue2 概念介绍
- 模板的采用方式
- 表单组件
- 实例方法
- 内置指令

- 指令用法
- 自定义指令
- v-lazy 的用法
- lazyload 的实现
- 组件常规通信
- 在线运行组件
- 响应式的规则

Vue2 源码

- 手写 Vue 响应式原理
- 手写 Vue 模板编译原理
- 手写 Vue 组件化机制原理
- 手写 Vue 生命周期原理
- 手写 Vue 虚拟 DOM 和 Diff 算法原理
- 手写 Vue 计算属性、watch 等原理
- 手把手剖析 Vue2 源码（剖析 props、slot、directive...）

Vue2 路由

- 手写 Vue-Router 中 Hash 模式和 History 模式的实现
- 手写动态路由 addRoutes 原理
- 手写$route及$router 实现
- 手写 router-link 及 router-view 组件
- 手写多级路由实现原理
- 手写 Vue-router 中路由钩子原理

Vue2Vuex

- 手写 state、getters、mutation、actions 原理
- 手写 vuex 中的 modules 原理及 namespaced 原理
- 手写 vuex 中插件机制 replaceState、subscribe...
- 手写 vuex 中辅助函数 mapSate、mapGetters...

Vue2 测试

- 单元测试概念
- 单元测试 jest 应用

Vue2SSR

- SSR 实现原理剖析及使用场景剖析
- 通过 Webpack 构建 Vue 项目，实现客户端打包和服务端打包
- 使用 koa 实现服务端渲染

- 集成 Vue-Router 及 Vuex

Vue2PWA

- 使用 webcomponent 实现折叠菜单组件
- pwa 开始
- pushApi-notification

Vue2 珠峰课堂项目

- 项目生成
- 路由的模块化
- 容器组件的使用
- ajax 请求配置
- vuex 模块化操作
- 实现轮播图功能
- 调用接口存储数据
- 注册实现
- 实现获取验证码
- 登录实现
- 封装路由钩子
- 路由钩子校验用户登录状态
- 路由权限动态添加
- 菜单递归渲染
- websocket 封装

Vue2 组件库

- button 的基本用法
- 单元测试
- 文档配置
- 布局组件
- 容器组件
- input
- 文件上传
- time-picker
- infiniteScroll
- popover
- 轮播图实现
- 分页和表格渲染

Vue2 面试题

- 响应式原理
- 数组更新问题
- 模板编译原理
- 生命周期和 mixin
- diff 算法原理
- set 方法的实现
- vue 的生命周期
- 组件间的数据传递
- Vue2 面试题

Vue3 实战

- Vue3 Composition API 组件库开发
- Vue3 中的自定义 canvas 渲染器
- Vue3+vuex 中的 typescript 复杂类型推断
- Vue3 中的 SSR 服务器端渲染

Vue3 源码

- 手写 Vue3.0 模板编译原理
- 手写 Vue3.0 响应式原理(实现 reactive ref computed effect)
- 手写 Vue3.0 虚拟 DOM 及组件渲染原理
- 手写 Vue3.0 中生命周期原理及异步渲染原理
- 手写 Vue3.0 中的 DOM-DIFF 算法
- 手写 Vite 工具实现原理及热更新原理
- 手写 Vue3 中 diff 算法属性比对及子元素比对流程
- 手写 Vue3 中最长递增子序列算法
- 手把手剖析 Vue3 源码

Vue3 路由

- 核心路由系统的实现
- vue-router4 基本结构实现
- VueRouter 中响应式原理
- RouterView 的实现
- 路由导航守卫的实现

Vue3Vuex

- vuex4 基本使用及原理解析
- 实现基本的 vuex 的功能
- vuex4 中模块的状态的实现
- vuex 中的模块实现原理
- vuex 中的命名空间的实现
- 实现 vuex 中的严格模式
- vuex 中插件系统的实现
- Vuex 中注册模块的实现
- vuex4 源码预览

Pinia2

- 掌握 pinia 设计思想，以及和 vuex 对比
- pinia2 实战开发
- 实现 optionsStore 和 setupStore
- 实现$state、getters、$patch、$reset、$subscribe、$onActions、$dispose
- 实现插件系统和持久化插件
- 实现辅助函数 storeToRefs、mapState、mapWritableState 和 mapActions

Vite3+Vue3 工程化

- Vite3 新特性讲解
- Eslint + Prettier + Husky 配置
- 路由和 pinia 状态管理
- tailwindcss 和 UnoCSS 实战
- 使用 Vitest 编写单元测试
- CICD 持续集成

Vite2+Vue3+Typescript 实战

- Vite2+Vue3+Typescript 开发环境
- ESlint+Prettier+git hooks 配置实战
- vite2 中的样式处理和静态资源
- 接口 mock 和 axios 封装 JWT
- pinia+路由、naive-ui 组件库实战
- Jest 单元测试和 cypress E2E 测试
- HMR 热更新 API 全实战
- vite2 实现 SSR 服务器端渲染
- vite2 静态站点生成
- esbuild、rollup 和 vite2 插件实战

- rollup 和 vite2 插件工作流和钩子
- rollup 和 vite2 插件 API 详解
- 实现 rollup 的 babel、commonjs、node-resolve、typescript、terser、postcss、serve@vitejs/plugin-vue-jsx 插件
- 实现@vitejs/plugin-vue、@vitejs/plugin-vue-jsx 等插件
- 从零实现 vite2

Vue3 组件库

- 从 0 到 1 实现自己的组件库
- 使用 VitePress 搭建组件库文档
- 基于 VueCli4 编写组件测试，Karma、Mocha、Chai
- 实现组件库的按需加载，组件库主题定制化
- 从零封装树形组件
- 从零封装日历组件
- 从零封装表单组件
- 从零封装模态窗口组件
- 从零封装轮播图组件
- 从零封装表格组件
- 从零封装上拉加载和下拉刷新组件
- 从零封装异步加载的省市级联组件
- 组件的单元测试和集成测试

Vue+TS 实现复杂表单组件

- 利用 TS 定义组件所需属性及事件
- 掌握表单组件的设计思想及组件间数据通信处理
- 实现 Form、FormItem 等组件
- 实现表单组件的数据绑定、校验规则处理、错误显示
- 使用 async-validator 实现对表单的校验功能

Vue3+TS 实现虚拟树组件

- 利用 TS 定义组件所需属性及事件
- 掌握树组件的设计思想及树中扁平化数据的处理
- 实现树的展开、节点选择、节点半选、全选、及自定义节点内容、封装 checkbox 组件、节点组件
- 虚拟滚动组件，实现 Tree 组件的虚拟滚动
- 实现树中数据懒加载

Vue3+TS 实现 UpLoad 组件

- 利用 TS 定义组件所需属性及事件
- 深度掌握 Vue3 中属性传递和事件
- 掌握上传组件的设计思想，实现上传功能
- 实现类型和大小限制、支持照片墙、自定义缩略图、图片列表展示、上传文件列表控制等
- 实现手动上传和拖拽上传图片功能

Vue3 珠峰课堂项目

- Vue3 概念
- 掌握 ts
- Vue3 全家桶
- 头部编写
- 轮播图实现
- 列表实现
- 更新逻辑

VueSSR

- SSR 原理和设计理念
- 集成 KOA 实现服务器端渲染
- webpack 构建 Vue SSR 项目
- 集成路由及代码分割
- 集成 VueSSR 和 Vuex 实现数据同步

Vue 项目优化篇

- 路由懒加载
- 页面预渲染
- SSR 之 Nuxt 实战
- Vue 骨架屏
- Vue-devtools 开发插件
- Vue 动画原理

Vue 虚拟列表

- 虚拟列表的概念和用途
- 理解虚拟列表工作原理
- 创建虚拟列表组件
- 优化虚拟列表性能

Vue3.0 + TS 全家桶项目

- Vue-cli4 + Vant 项目搭建
- 服务器构建 Typescript + Express + Mongodb
- 路由配置及 Vuex 最佳实践
- 进阶 Vue3 中逻辑封装及 CompositionAPI 使用
- 数据获取和 axios 应用拦截器
- 基于 JWT 的注册登录权限管理
- 公共组件封装
- 上拉刷新、下拉加载、图片懒加载
- 项目布署和上线

Vue2.0 + ElementUI 管理系统全栈项目

- 后端 Koa + Mongodb + Redis
- 项目构建流程及最佳实践
- 前后端分离登录权限
- 路由权限控制
- 基于角色的菜单权限及按钮权限
- Vue 中动态管理路由及 Vuex
- Vue 中 axios 二次封装
- Vue 中 websocket 封装(心跳检测，断线重连等)
- 项目打包及上线流程

Vue2/3 面试题

- 谈谈你对 Vue 的理解
- 谈谈你对 spa 的理解
- vue 为什么需要虚拟 DOM
- 谈一谈对 Vue 组件化的理解
- 既然 Vue 通过数据劫持可以精准探测数据变化，为什么还需要虚拟 DOM 进行 diff 检测差异?
- 请说一下你对响应式数据的理解
- Vue 中如何检测数组变化
- Vue 中如何进行依赖收集
- Vue.set 方法是如何实现的
- v-if 和 v-show 的优先级
- watch&computed
- 解释 ref 和 reactive 区别
- watch 和 watchEffect 的区别
- 如何将 template 转换 render 函数
- new Vue()过程中做了些什么
- Vue.observable 你有了解过吗?

- v-if 和 v-for 哪个优先级更高
- 生命周期有哪些
- diff 算法
- 请说明 Vue 中 key 的作用和原理，谈谈你对它的理解
- Vue.use 是干什么的?
- Vue.extend 方法的作用?
- Vue 组件 data 为什么必须是个函数?
- 函数组件的优势
- Vue 中的过滤器了解吗？过滤器的应用场景有哪些？1
- v-once 的使用场景有哪些
- Vue.mixin 的使用场景和原理
- Vue 中 slot 是如何实现的？什么时候使用它?
- 说说你对双向绑定的理解，以及它的实现原理吗?
- Vue 中.sync 修饰符的作用?
- Vue 中递归组件理解
- 组件中写 name 选项有哪些好处及作用?
- Vue 常用的修饰符有哪些有什么应用场景?
- Vue 中异步组件的作用及原理
- 说说你对 nextTick 的理解?
- keep-alive 平时在哪里使用?
- 自定义指令的应用场景
- Vue 中使用了哪些设计模式
- Vue 中的性能优化有哪些?
- 单页应用首屏加载速度慢的怎么解决?
- Vue 项目中你是如何解决跨域的呢?
- Vue 项目中有封装过 axios 吗？主要是封装哪方面的?
- vue 要做权限管理该怎么做？如果控制到按钮级别的权限怎么做?
- Vue-Router 有几种钩子函数，具体是什么及执行流程是怎样的
- Vue-Router 几种模式的区别?
- vue 项目本地开发完成后部署到服务器后报 404 是什么原
- 谈一下你对 vuex 的个人理解
- 如何监听 vuex 中数据的变化
- 页面刷新后 vuex 的数据丢失怎么解决?
- mutation 和 action 的区别
- 有使用过 vuex 的 module 吗？在什么情况下会使用？fbr
- Vue3 中 CompositionAPI 的优势是?
- Vue3 有了解过吗？能说说跟 Vue2 的区别吗?
- Vue 项目中的错误如何处理的?

- Vue3 中模板编译优化
- 你知道那些 Vue3 新特性

React.js

React0.3

- 渲染文本组件
- 渲染原生 DOM 组件
- 渲染自定义组件
- 实现 setState
- 对比属性
- 对比子元素
- 获得补丁数组
- 打补丁更新
- DOM-Diff 策略
- React 实战

React15

- 手写实现虚拟 DOM
- 手写实现类组件、函数组件和原生组件的渲染
- 手写实现 setState
- 手写实现 DOM-DIFF
- 手写实现 DOM 更新
- 手写实现合成事件和事务机制

React16

- 手写 React17 中的虚拟 DOM
- 手写 Fiber 架构算法和数据结构
- 手写 Hooks 的原理和实现

React17

- 手写 Fiber 架构算法和数据结构
- 手写 DOM-DIFF 算法
- 手写 React 合成事件系统
- 手写 React 中的 setState 异步渲染
- 手写 React 中的 Hooks
- 手写 Fiber 任务调度和更新策略

React18 新特性

- Vite2+React18 工程化开发
- ConcurrentMode(并发模式)和批量更新模式
- Suspense、SuspenseList 和 ErrorBoundary 案例实战
- 更新优先级和 setState 批处理优化
- startTransition 和 useDeferredValue 案例实战
- useSyncExternalStore 案例实战
- useInsertionEffect 案例实战
- React18 项目升级指南和开发建议

React18+Router6 的流式 SSR 服务端渲染

- 实现客户端和服务器的 SSR
- 实现客户端和服务器端路由
- 实现 redux 状态管理集成
- 实现客户端和服务器异步接口数据获取
- 实现 Koa 和 Express 服务器集成
- 实现注册登录和权限菜单
- 集成 CSS 和 SEO

React18

- 实现虚拟 DOM
- 实现 Fiber 结构
- 实现初次渲染
- 实现函数组件
- 实现合成事件
- 实现 useReducer
- 实现 useState
- 实现 DOM-DIFF
- 实现 useEffect
- 实现 useLayoutEffect
- 实现任务调度
- 初次渲染
- 更新渲染
- 并发渲染
- 批量更新
- 高优更新打断低优更新
- 实现 useRef
- 饥饿问题

- 实现 React Context

实现 ReactHooks

- 实现 useState
- 实现 useCallback 和 useMemo
- 实现 useReducer 和 useContext
- 实现 useEffect 和 useLayoutEffect
- 实现 forwardRef 和 useImperativeHandle

自定义 ReactHooks

- 实现 useRequest 实现分页请求
- 实现 useDrag 实现拖拽
- 实现 useForm 实现表单自动托管
- useAnimation 实现自定义动画

ReactRouter V6

- 路由配置和二级路由
- 路由懒加载
- 路由重定向
- 路由之权限管理和受保护的路由
- 手写一个完整的 `React-router6` 路由库(HashRouter、BrowserRouter、Route、Routes、Link、NavLink)
- 实现路由 hooks(useOutletContext、useLocation、useMatch、useNavigate,useOutlet)
- React 路由原理
- 实现核心路由模块
- 实现 history
- 实现路径参数
- 实现 Link 组件和 NavLink 组件
- 实现嵌套路由和 Outlet
- 实现跳转和重定向
- 实现受保护路由
- 实现配置式路由和懒加载

Redux

- Redux 核心用法 Action/Reducer/Store
- 手写实现 Redux、react-redux、connected-react-router
- 手写 Redux、react-redux、redux-logger、redux-promise、redux-thunk、redux-saga、redux-actions、reselect、redux-persist 等经典 redux 中间件类库
- 手写 hooks 版 react-redux(useStore、useReduxContext、useDispatch、useSelector)

Redux-toolkit

- Redux Toolkit 实战
- 实现 configureStore
- 实现 createAction
- 实现 createReducer
- 实现 createSlice
- 实现 createAsyncThunk
- 实现 Redux Toolkit Query

Recoil 和 xstate

- React Context
- Recoil 使用和实现
- 有限状态机
- 实现 xstate
- 实现 toggle 和 Machine

zustand

- zustand 的概念和优势
- zustand 项目实战
- 实现创建仓库、读取状态、更新状态
- 实现状态分享，计算属性和事件处理
- 实现日志和持久化中间件

useRequest

- 实现自动请求/手动请求
- 实现轮询、防抖和节流
- 实现屏幕聚焦重新请求
- 实现错误重试和 loading delay
- 实现 SWR(stale-while-revalidate)和缓存

AntDesignPro

- AntDesignPro 搭建
- 用户注册和用户登录
- 新增修改用户和用户列表
- 权限中间件

Dva 基础版

- 实现 dva
- 实现路径跳转

Dva 完整版

- dva 使用
- 实现 dva 和 reducers
- 实现 effects
- 实现 onEffect
- 实现 extraReducers
- 实现 onAction
- 实现 onReducer
- 实现 onError

UMI3 实践

- 项目创建、启动和构建
- 全局布局
- 嵌套路由和动态路由
- 权限路由

UMI 源码

- 约定式路由实战
- 源码调试 UMI
- 实现 UMI 的运行时系统
- 实现 UMI 的插件系统
- 实现编译时插件
- 实时目录生成功能
- 实现服务启动
- 实现运行时插件

UMI4

- Umi4 新功能和新特性
- Umi4 设计和架构
- Umi4 技术栈选型
- Umi4 开发最佳实践
- Umi4 Max 开发最佳实践

Formily2 实战

- 实现注册登录和分步表单
- 实现表单校验和表单布局
- 实现异步数据源和表单受控
- 实现联动逻辑和联动计算器
- 实现自定义组件和前后端数据差异兼容方案
- 实现管理业务逻辑和按需打包

Umi4+Formily2 开发动态表单系统

- Umi4 新功能特性解析
- AntDesignProV6 前瞻
- Formily2 案例实战
- 动态网站架构与接口设计
- 从零开发动态表单系统

从零实现 mobx 和@formily/reactive

- mobx+mobx-react 案例实战
- 实现 mobx 中的 observable、action、computed
- 实现 mobx-react 中的 Provider、inject、observe、Observer、useObserver
- 实现 mobx-react 中的 useAsObservableSource 和 useLocalStore
- @formily/reactive 对 mobx 的增强和优化

从零实现 Formily2.0 核心

- Formily 分层架构设计分析
- 实现@formily/reactive 核心
- 实现@formily/core 核心
- 实现@formily/react 核心
- 实现@formily/antd 核心
- 实现 JSON Schema 渲染流程

Immutable

- 什么是 Immutable
- Immutable 类库和优势
- Immutable 实现 React 性能优化
- Redux+immutable
- Redux-immutable 中间件
- 实现 React-router-redux

JSX 转换器

- AST 抽象语法树
- babel 工作流
- 实现旧版 JSX 转换
- 实现新版 JSX 转换
- 实现属性的转换

Mobx4

- Mobx Vs Redux
- Mobx 核心思想
- Mobx 的 Decorator
- Proxy
- 实现 mobx 的 observable
- 使用对可观察对象做出响应
- 实现 action
- mobx 实战
- mobx 优化

Mobx6

- observable、makeObservable、makeAutoObservable
- Autorun、computed、action、flow、bound、Reaction、When、runInAction
- mobx+mobx-react 综合案例
- 实现 observable、reaction、autorun、action
- 实现 observer、useObserver、Observer、useLocalObservable

ReactQuery

- ReactQuery 状态和配置
- useQuery 同步服务器和客户端状态
- useMutation 更改状态
- QueryObserver 状态订阅
- useQueries 并发同步状态
- useInfiniteQuery 分页查询
- useFetching 全局加载状态

手写 ReactSSR

- SSR 支持路由
- 集成 redux 和子路由
- 代理接口和服务器加载数据
- 登录和权限

- 支持样式
- 支持流式 SSR
- next.js 实践
- 调用接口
- 用户注册
- 实现懒加载组件和模块
- 集成 redux
- loading
- 服务器布署

Next.js

- 什么是同构
- Next.js 项目初始化
- 跑通路由和样式
- 二级路由
- 调用接口
- 懒加载
- 用户登录
- 集成 redux
- loading 效果
- 受保护路由
- 自定义 Document
- 新版初始化函数
- 项目布署

React 动画

- React 高性能动画实战
- React 动画的原理解析
- 实现 Transition
- 实现 CSSTransition
- 实现 SwitchTransition
- 实现 TransitionGroup

React 虚拟列表

- React 虚拟列表的核心原理
- React 虚拟列表综合实战
- 从零实现固定高度虚拟列表
- 从零实现不定高度虚拟列表
- 滚动状态和滚动到指定条目

React 性能优化

- 实现编译阶段的优化
- 实现路由切换优化
- 实现更新阶段优化
- immutable.js
- 时间分片和虚拟列表渲染大数据量
- React 性能分析工具
- React 性能分析器
- 使用 React.Fragment
- React.Lazy 延迟加载组件
- 错误边界(Error Boundaries)
- PureComponent
- 长列表优化
- React Devtool

React 拖拽

- HTML5 拖放 API 全解析
- react-dnd 可视化拖放排序实战
- React Hooks、高阶组件和关注点分离等设计模式
- 从零实现 react-dnd-html5-backend
- 从零实现 react-dnd 核心 API

React 测试

- 测试框架
- Jest 实战
- 测试覆盖率
- Matchers
- 测试 DOM
- 异步请求
- 钩子函数
- 测试 mock
- TDD(Test-Driven-Development)
- BDD(Behavior Driven Development)
- UI 测试

搭建 AntDesign4 组件库

- webpack 配置
- storybook 文档和组件编写
- 单元测试+E2E 快照测试+代码覆盖率
- eslint+prettier+editorconfig
- git hook
- 编译发布
- 持续集成

React 组件库

- 从零实现 Button 组件和单元测试
- 使用 VuePress 实现专业文档
- 从零实现表单组件
- 从零实现布局组件
- 从零实现输入组件
- 从零实现拖拽文件上传组件
- 从零实现时间选择组件
- 从零实现模态窗口组件
- 从零实现轮播图组件
- 从零实现分页和表格渲染组件
- 从零实现树型组件
- 从零实现无限滚动组件

React 源码中算法实战大汇总

- 位运算和二进制原理和应用
- 最小堆原理和应用
- 链表原理和应用
- 树的深度和广度优先遍历原理和应用
- 栈的原理和应用

从零实现 React 任务调度和 lane 模型

- 实现基本任务调度
- 实现时间切片
- 实现多个任务调度
- 实现任务优先级
- 实现延迟任务和任务取消

从零实现国际化解决方案

- react-intl 核心概念和工作流
- React18 和 react-intl 实战
- Vue3 和 react-intl 实战
- 实现 IntlProvider
- 实现 FormattedMessage 和 UFormattedNumber

Typescript+React 工程化实践

- nunjucks 模版引擎、yaml 配置与法 | mock.js 模拟数据
- dva
  - 创建应用
  - 集成 AntDesign
  - 定义路由和 UI 组件
  - 链接仓库
  - 使用 effects 和 reducers
  - 从零实现 dva 所有核心 API
- umi3
  -Umi3 和 antd-design-pro 中后台项目实践
  - Umi3 的项目整体目录结构
  - Umi3 集成 antd-design-pro
  - Umi3 中配置式路由、动态路由和权限方案
  - Umi3 中 dva、redux 和 redux-persist 数据流解决方案
  - Umi3 中国际化实战
  - 手写 Umi3 自定义插件
  - 手写 Umi3 核心实现

跨端开发

ReactNative

- UIExplorer 项目
- css 盒子模型和样式
- css 元素浮动
- flexbox 布局
- ReactNative 长度单位
- RN 事件
- React 动画原理
- 实现一个 Navigator
- App 架构之目录结构、路由和组件
- App 架构之网络和 Container

- App 架构之命名空间
- ReactNative 第三方插件
- 珠峰课堂项目实战

Flutter

- dart 语法
- flutter 环境配置
- 常用组件
- 布局
- 导航和动画
- flutter 版珠峰课堂项目实战

electron

- 配置 electron 环境
- 主进程和渲染进程
- 进程间通信
- 文件对话框操作
- 消息通知和快捷键

uni-app

- 调试
- 使用 hbuilder
- flex 布局
- 多端发布
- 路由和动画
- 微信分享
- uni-app 版珠峰课堂项目实战

微信小程序

- 小程序与普通网页开发的区别
- 注册小程序-公众号注册
- 下载微信开发者工具
- 如何在 vscode 中开发微信小程序
- 小程序尺寸单位 rpx
- 小程序导入样式方法
- 小程序的选择器
- Image 高度自适应问题
- 给页面加背景色
- opent-type 获取用户信息

- 注册小程序-直接注册
- 小程序审核流程
- 添加开发人员
- 快速创建小程序
- 介绍开发者工具
- 小程序的目录解构及四种文件类型
- 手动创建一个项目
- 小程序文件的作用域
- view 与 text 组件介绍
- 授权得到用户信息
- 数据绑定
- 判断用户是否授权
- 条件渲染及 block 组件
- 事件及事件绑定
- data-xxx 大小写问题
- 页面跳转
- 设置 tabBar
- 配置导航样式
- swiper 组件
- 列表渲染
- 页面生命周期
- 转发分享
- request 请求后台接口
- http-promise
- web-view 组件
- 获取用户收货地址
- 获取地理位置
- 自定义组件
- 回答同学的一些问题
- 小程序支付及其他支付方式的讨论
- 自定义 lesson 组件
- 自定义 star 组件
- 编写全部课程页面
- 搜索页面样式
- 数据缓存
- 根据搜索内容显示数据
- 无搜索数据的处理
- 下拉刷新

- 加载更多
- 模糊查询
- 设置上拉触发事件距离
- 跳转详情页并动态设置导航文字
- 课程详情页面样式
- button 分享及拨打电话
- animation
- wxs
- 编写评论页代码
- 使用 scroll-view 组件时的注意事项

数据可视化

Canvas

- Canvas 基础
- Canvas 弧和圆形
- Canvas 文字
- Canvas 图片
- Canvas 运动

FlappyBird

- canvas 基础知识
- 画布和画图
- background 实现
- 实现大地
- 绘制管道
- 绘制小鸟
- 碰撞检测
- 场景管理

Three.js 跳一跳

- 基础信息属性配置
- WebGL 介绍以及 Three.js 的基础应用
- 几何体创建以及相机镜头位置改变
- 更新相机坐标实现视觉动画
- 绑定事件实现 jumper 跳跃功能
- 回顾思路梳理逻辑
- 最终完成实现成功和失败的处理和重置操作

Linux

- Linux 与 Windows 的不同
- Linux 安装和虚拟机的使用
- 桥接、NAT、Host-Only 等网络连接
- 快照、克隆、挂载点和分区
- Linux 常用命令 VI 编辑器、用户与权限管理、服务管理、软件管理、网络管理、系统命令
- Shell 实战 监控服务和主机网络状态

MySQL

- MYSQL 安装与使用
- MYSQL 系统架构
- 数据处理之增删改查
- 数据类型和约束分页
- 索引和慢查询性能分析
- 数据库安全之防止 SQL 注入
- 数据库设计 ER 图设计
- 数据库事务和锁 |
- 数据库设计之三大范式
- 分组和聚合函数
- 基于角色的权限访问控制（Role-Based Access Control）

Redis

- 5 种数据结构及使用场景
- API 的理解和使用
- Redis 客户端
- 发布订阅
- 事务
- 备份和恢复

Mongodb

- Mongodb 启动与连接
- 数据库操作
- 集合操作
- 插入、更新、删除、查询操作符
- 条件操作符、与和或、分页查询
- 执行脚本
- 权限
- 索引

- 用户和权限
- 导入导出数据
- 备份与恢复
- 锁定和解锁数据库
- 安全措施
- 数据库高级命令
- 固定集合
- Gridfs
- 建立索引
- 二维索引
- 主从复制
- 副本集
- 分片
- Mongoose
    - Schema
    - Model
    - Entity
    - 查询、删除、更新、查询文档

Nginx

- nginx 的安装和使用
- 模块和基本配置
- 正向反向代理等应用场景
- CDN
- 浏览器缓存
- 跨域
- 防盗链
- rewrite
- 负载均衡集群

Jenkins 持续集成

- jenkins job
- shell 集成
- 集成 nginx 和 git
- 持续集成和部署
- travis gitlab ci

Docker

- 虚拟机
- Linux 容器
- Docker 核心概念
- Docker 架构
- Docker 镜像
- Docker 容器
- Dockerfile
- Docker 数据盘
- 网络配置
- docker-compose

服务器布署

- 服务器布署与运维
- Nginx+Docker 持续集成

Kubernetes

- Kubernetes 核心
- Kubernetes 安装布署
- 配置 Master
- 直接布署 nginx
- 通过 yaml 布署 mysql
- Pod、deployment 和 Service
- k8s 部署
- 灰度发布
- 滚动发布
- 服务可用性探针
- 储存机密信息
- 服务发现
- 统一管理服务环境变量
- 污点与容忍
- 布署 MYSQL
- 布署前端项目
- 布署后端项目
- 集成 Jenkins
- 推送触发构建
- 配置 WebHooks

解决方案

Vue3 低代码

- 拖拽编辑器搭建
- 拖拽的实现
- 实现拖拽的辅助线的功能
- 实现重做和撤销功能及快捷键
- 实现 json 的导入导出
- 实现菜单功能
- 实现编辑菜单功能
- 实现操控栏渲染
- 实现操作栏配置属性
- 实现数据的双向绑定
- 实现范围选择器物料
- 下拉菜单物料实现
- 实现自定义组件大小功能
- 调整组件大小的功能

RxJS

- RxJS 中的 Observable、Observer、Subscription、Operators、Subject 等核心概念讲解
- RxJS 中的弹珠图等可视化工具
- RxJS+React Hooks 异步接口请求和防抖节流实战
- 实现 Observable、Observer、Subscription
- 实现防抖节流等 Operators 和 Subject
- mergeMap,switchMap、concatMap、mergerAll 和 concatAll 等
- fromFetch、share、debounceTime、takeUntil 和 withLatestFrom 等
- RxJS 封装 http 请求功能
- RxJS 控制异步任务最大并发数
- RxJS 实现 AutoComplete 组件
- RxJS 实现拖拽功能封装
- observable-hooks 实战和原理

Serverless

- Serverless 云函数
- serverless framework
- Serverless Components
- 云函数 SCF 组件
- API 网关组件
- 布署静态网站

- 布署 express 项目
- 布署 express+layer 项目
- 部署 Vue+Express 全栈应用

TS+React 珠峰课堂

- webpack 环境搭建
- 底部页签导航
- React 动画
- Redux 改变课程分类
- 实现头部轮播图
- 课程列表列表
- 长列表优化(只渲染可视区域)
- 下拉刷新(节流)
- 上拉加载(防抖)
- 记录滚动条位置
- 课程详情
- 用户注册和登录
- 受保护的个人中心
- 购物车动画

TS 后台开发

- node 中的 TS 类型
- Express 中的 TS 类型
- mongoose 中的 TS 类型
- 各种 Express 中间件的 TS 类型

TS 开发 React

- Typescript 配置 React
- React 中的 TS 类型
- Redux 中的 TS 类型
- React 路由中的 TS 类型
- connected-react-router 中的 TS 类型

前端监控 SDK

- 前端监控目标
- 前端埋点方案
- 监控 JS、Promise、资源加载错误
- 接口异常采集
- 白屏和加载时间

- 性能指标和卡顿
- PV 和用户停留时间
- 可视化报表查询

前端监控实战

- 埋点上报
- 数据处理
- 计划任务
- 前台展示

动态 CMS 项目

- Umi4 新功能特性解析
- AntDesignProV6 前瞻
- Formily2 案例实战
- 动态网站架构与接口设计
- 从零开发动态表单系统

弹幕

- Websocket+Canvas 弹幕

微前端实战

- 微前端工程化
- 同时支持 angular、vue、react 的微前端框架实战
- single-spa 和 qiankun 实战

微前端原理

- System.js 的实现原理
- single-spa 实战和手写实现
- qiankun 的基本使用和手写实现
- microApp 和无界的基本使用和手写实现
- 模块联邦使用和工作原理

爬虫

- 用 superagent+cheerio 爬取网页内容
- 数据持久化到 mysql 数据库
- 用户个性化邮箱订阅标签
- 数据更新按兴趣分发推送邮件
- 用 ElasticSearch 实现全文检索

Node 博客项目

- 初始化项目和依赖的模块
- 跑通路由
- 使用 bootstrap 渲染模板
- 实现用户注册的功能
- 实现用户的登录功能
- 实现会话功能并控制菜单显示
- 增加登录状态判断中间件
- 成功和失败时的消息提示
- 实现上传头像并在导航的右上角显示个人信息
- 新增发表文章
- 首页显示文章列表
- 删除文章
- 更新文章
- 实现搜索功能
- 实现分页的功能

聊天室

- 什么是实时通信
- 什么是 Websocket
- websocket 数据帧格式解析
- 从零实现 websocket 协议
- websocket 和 http 的对比
- 使用 socket.io 实现聊天室
- 匿名聊天
- 有用户名的聊天和用户列表
- 用户私聊
- 划分不同的聊天房间
- 消息持久化

骨架屏

- 创建 webpack 插件
- 启动 express 服务
- 启动 puppeteer
- 截取骨架内容
- 元素转换
- css 抽象语法树

BFF

- BBF 的应用场景和架构设计
- Node 搭建高性能 BFF 服务
- 使用 RPC 协议提升性能
- 使用 Redis 实现高速缓存
- 使用异步消息队列 MQ 实现海量任务处理

OAuth

- OAuth 设计思路
- OAuth 工作流程
- 客户端的授权模式
- 接入 QQ
- 开发自己的 Oauth 系统

CMS 系统

- 用户注册和登录
- JWT 权限认证
- 用户头像上传
- 用户手机号注册与登录
- 页面和按钮菜单权限
- 用户列表管理
- 为角色分配权限
- 为角色分配用户
- egg.js+mysql 实现后端接口

CSS-IN-JS

- CSS-in-JS 解决方案的优缺点
- CSS-in-JS 实现方案之 styled-component 和 emotion
- CSS-in-JS 中的 CSS 方法使用和属性优先级
- 样式化组件实战
- 关键帧动画和主题使用
- 实现 CSS-in-JS 核心组件

GraphQL

- GraphQL 概念
- 使用 GraphQL 查询和变更数据
- 后端搭建 GraphQL 服务器
- ReactHooks 和 GraphQL 项目实战

PWA

- manifest.json 配置
- service worker 生命周期
- fetch
- 请求拦截
- cache api 以及缓存策略
- Notification
- API
- workbox 应用
- Vue 中应用 PWA

RBAC

- 基于角色的权限访问控制
- 数据库设计与 SQL
- Passport 配置与实现
- 前台配置与管理

就业指导

- 就业辅导
- 前端就业之道
- 大厂晋升指导

实战训练营

Vue3 低代码训练营

- 拖拽编辑器搭建
- 拖拽的实现
- 实现拖拽的辅助线的功能
- 实现重做和撤销功能及快捷键
- 实现 json 的导入导出
- 实现菜单功能
- 实现编辑菜单功能
- 实现操控栏渲染
- 实现操作栏配置属性
- 实现数据的双向绑定
- 实现范围选择器物料
- 下拉菜单物料实现
- 实现自定义组件大小功能
- 调整组件大小的功能

Sentry 前端监控实战训练营

- 下载 sentry docker 仓库并部署
- 创建前端项目并接入 sentry sd
- 上传 sourcemap
- 面包屑 Breadcrumbs、Traces、Transactions 和 Spans
- 页面加载和性能指标
- 导航操作检测
- Alert 报警
- 其他：手动上报错误、设置上报等级、错误边界组件、集成 redux、rrweb 重播

微前端大型落地实战训练营

- 主应用和多子应用
- 主子应用模版
- webpack 插件
- 打包微前端项目
- 浏览器插件
- 类 qiankun 沙箱
- 404 捕获
- 渲染器
- 调试工具
- Vue bindings 优化整合

从零到一手写 taro2 实战训练营

- 小程序模板
- 项目创建和核心文件
- 输出入口文件与配置文件
- 入口与 transform 函数
- 制作虚拟 DOM

Nest 微服务电商实战训练营

- Nest 配置、日志模块、自定义装饰器和 Redis 缓存
- websocket 和消息队列、注册中心和配置中心
- protocol buffers 和服务通信
- 用户、商品服、地址、库存、购物车、订单、支付、和消息服务
- PM2、docker-compose、redis、jenkins 布署上线

可视化 CMS 编辑器实战训练营

- 响应式和组件自动布局
- 父子组件嵌套以及自动布局
- 支持拖拽生成丰富的布局样式
- 抽离可以复用生成的组件
- 服务端渲染
- 支持小程序

直播平台全栈开发实战训练营

- 布局和页面
- 礼物和弹幕
- 直播和评论页面
- 管理后台开发
- 个人中心和支付
- 数据可视化
- 布署上线

Vite2+Vue3+TS 中台管理系统

- 面包屑
- 标签导航
- 侧边栏(权限菜单)
- 自定义 icon（Svg Sprite 图标)
- 拖拽看板
- 路由检索
- 主题切换 (基于 element-plus)
- Screenfull 全屏
- 图片上传
- 登陆注册(jwt)
- 权限控制(系统管理：用户管理、角色管理、菜单管理)
- 权限验证(页面权限、指令权限)

Nest-serverless-graphql 实战

- 脚手架安装使用
- WebIDE 创建云函数
- Serverless Framework
- Serverless 布署 Egg 项目
- Serverless 布署静态项目
- Serverless 布署 Nest.js 项目

- Nest.js 项目对接 MySQL
- GraphQL 核心应用
- 全栈项目实战

持续集成 CI/CD

- 掌握 Docker 的安装与镜像容器基础操作
- 掌握如何使用 docker 安装 Gitlab/Jenkins
- 掌握使用 Jenkins + Gitlab 实现自动化构建
- 搭配镜像仓库实现制品版本管理
- 使用 Ansible 批量将制品部署到服务器

Kubernetes 核心和项目实战

- Kubernetes 安装与配置
- 集群的方式配置 Master
- 集群的方式配置节点
- 直接布署 nginx
- 通过 yaml 布署 mysql
- 布署 Pod、service 以及 ingress
- 实现灰度发布
- 实现滚动发布
- 实现服务可用性探针
- 机密信息的存储
- 服务发现
- 统一管理服务环境变量
- 污点与容忍
- 布署 MYSQL 数据库
- 布署后端服务
- 布署前端应用
- 集成 jenkins 集成
- 通过 WebHooks 推送触发持续构建

Egg.js 源码和实战训练营

- 路由和中间件
- 控制器和服务
- 模板渲染和插件系统
- 定时任务和错误处理
- 生命周期和框架扩展定制
- 源码分析并手写 Egg.js
- Egg.js 实战

Vue 专业级后台管理系统

- 建立 Vue CLI4 版本的环境搭建
- 路由配置的模块化管理
- Element-ui 的应用和组件模块化管理
- Axios 的二次封装
- Vuex 的模块化配置
- 获取后台数据应用
- 轮播图实战
- 用户注册和登录权限的应用
- 验证码的应用
- 路由和菜单权限的渲染和处理
- 对 webSocket 的封装和使用

typescript+ReactHooks 实现珠峰课堂

- 用 Webpack 搭建 React+ts 环境
- 路由配置和使用 antd4 搭建布局
- 移动端使用 rem 和 flex 进行布局
- 实现首页头部导航效果
- express+ts 实现后端项目
- 实现个人中心和用户注册登录的功能
- 实现首页加载，防抖节流，虚拟化列表的功能
- 实现购物车效果
- 实现服务器部署功能

微信小程序实战

- 用 express 实现后台
- 全局配置每个页面
- 实现防抖节流，列表优化功能
- 实现购物车功能
- 获取微信头像等相关权限

Antv/G6 实现思维导图实战

- 用 G6 绘制实现鼠标、键盘事件监听伪代码功能
- 实现图标展开和收缩功能
- 实现自定义边框，添加和编辑文案功能
- 控制开口方向布局，实现控制按钮样式
- 对节点事件的监听和实现修改文案和添加子节点功能

Nestjs 基础到实战项目实战

- 了解 nestjs 的脚手架及其常见命令
- 了解 nestjs 中依赖注入、模块、生命周期等知识
- 熟练对 ejs 模板、cookie 和 session 的使用
- 熟练掌握数据库中表、外键和事务等知识
- 熟练掌握实现增删改查功能
- 熟练掌握 TypeORM 在 nestjs 的应用
- 熟练掌握 nestjs 的守卫拦截和过滤器的使用
- 守卫做用户鉴权，进行权限处理功能
- RBAC 权限系统项目，完整的前后台一体化项目

Typescript+React 组件库实战

- 搭建环境，制作颜色和排版
- 完成发包流程
- 实现 Button、Icon、Avatar、Radio 等组件
- 实现轮播图、进度条、拖拽树组件等功能
- 实现 message、upload、分页和表格的组件
- 使用 hooks 和 ts 技术栈完成本次项目

Vue3+ts 组件库实战

- 打包部署 Gitee 的 page 服务
- 手动实现 vue3 工程化和路由配置
- 手动实现 Button、Input、Radio、Icon 等组件
- 手动实现表格、表单、数据视图、导航等功能组件
- 基于 TS 开发，按需加载，自定义主题
- 组件性能优化，支持树组件和表格组件虚拟滚动
- 可以专注开发，无需关心页面传参以及前进后退逻辑控制

前端全链路监控实战

- 前端监控概述、sentry、前端监控系统架构、fee 仓库
- 配置打点服务
- SDK 的使用
- 配置 kafka 和 filebeat
- 部署 mysql 和 redis，修改 fee，并部署 server 部分
- 部署 client 部分，完成整体流程

计算机基础

编译原理

- 编译器工作流
- 有限状态机
- 词法分析
- 上下文无关文法
- 分词
- 准备进行词法推导
- 实现结果的计算
- 实现减法和小括号
- 生成 AST 语法树
- AST 语法树的遍历
- AST 语法树的转换
- 代码生成
- 优先级结合性
- 解决左递归和结合性的矛盾
- 实现 babel 和 babel 插件
- 从零实现 JSX 到 JS 的 babel 转换插件

计算机网络

- OSI 七层模型
- TCP/IP 参考模型
- 物理层
- 数据链路层
  - 以太网
  - 总线型拓扑
  - 冲突检测
  - MAC 地址
  - 以太网帧
  - ARP 协议
- 互联网层(网络层)
  - IP 协议
  - 选址
  - 子网掩码和子网划分
- 传输层
  - TCP 数据包
  - TCP 序列号

    - 滑动窗口的拥塞检测
    - 三次握手和四次挥手
    - UDP 协议
- 应用层
    - DNS 协议
    - HTTP 协议
    - HTTPS 协议

网络协议实现

- 基于 Node 的 TCP 服务器从零实现 HTTPS 协议
- 基于 Node 从零实现 websocket 协议
- 基于 Node 从零实现 Ajax 对象 XMLHttpRequest

计算机网络面试题

- 网络分层的真实含义是什么？每一层都做了什么事？
- 键入网址再按下回车后究竟发生了什么？
- 为什么建立连接是三次握手，关闭连接是四次挥手呢？
- DNS 解析过程及原理？
- CDN 的原理是什么？
- TCP 协议如何保证传输的可靠性？
- TCP 协议用什么方式去解决拥塞的？
- 什么是滑动窗口协议与慢启动？
- TCP 和 UDP 区别？
- HTTPS 和 TLS 工作原理？
- HTTP1.1 和 HTTP2、3 的区别是？
- HTTP 和网络层有哪些优化手段？

实现 HTTP

- 实现 XmlHttpRequest
- 实现 POST

HTTPS

- http 通信解决的问题
- SSL 和 TLS 的区别
- SSL 协议
- HTTPS 协议改进过程
- 安全传递密钥
- 安全证书
- 实现 HTTPS 服务器

Git

- Git Flow 工作流
- Git 高级应用

二进制原理

- 计算机中的数值表示
- 进制转换
- 计算机中的数据
- 小数
- IEEE754 标准
- 1+ != 3
- JS 大数相加

二进制应用

- 计算机原码、反码和补码的原理
- 前端浮点数精度问题和超大数求和原理
- 前端中的 FormData、Blob、File、ArrayBuffer、TypedArray、DataView、DataURL、ObjectURL、Text
- 前端图片裁剪和上传预览
- 纯前端实现音频的合并剪辑处理

前端性能优化

- webpack 优化方案
- 浏览器缓存原理和最佳设置策略
- CDN 缓存优化
- EventLoop 异步更新
- 避免回流和重绘
- 节流与防抖
- 通过 Performance 监控性能