珠峰前端架构课第 16 版(2022 年 9 月 18 日更新)

课程简介

本课程主要面向 1 年以上工作经验的前端开发人员，讲师通过在自己在国内一流互联网公司的多年开发和面试经验，整理出一线大型互联网公司中高级工程师必备的核心技能，以前端架构课程学习、课后作业 、讨论答疑和多人协作项目实战等方式，帮助学员在较短时间内达到阿里 P6+级以上水平。

课程优势

- 13年口碑 珠峰教育成立于 2009 年，13 年来，珠峰只做前端、做精前端，早在 2015 年就开设了前端架构课
- 教育理念 珠峰成立 13 年来，踏踏实实为学生服务。始终如一做好教育，这是我们永恒不变的初心
- 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，不断升级迭代大纲
- 顶尖师资 我们的讲师均来自 BAT 等一线互联网大厂，十七年全栈开发经验的前端大佬领衔授课，8 年以上的授课经验，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
- 科学规划课程体系 从数万学员中提炼核心实用技术，再经过数年的打磨，形成一套科学完整易学习的前端进阶课程体系
- 一线企业导师 有来自阿里、百度腾讯一线大厂的数位前端技家专业指导课程迭代与项目实践
- 360°学练评测 知行合一，注重刻意练习和及时反馈，通过直播课、项目实战、课后总结、每日一道面试题等多种手段帮助学员把知识内化为自己的能力，并可以举一反三，持续进步
- 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

第 16 版部分更新

实战训练营

- **可视化 CMS 编辑器实战训练营**
- **sentry 前端监控实战训练营**
- **微前端大型实战训练营**
- **nestjs 微服务训练营**
- **直播平台实战训练营**
- vite2+vue3+typescript 中台实战
- 手写 taro2 实战项目
- UI 自动化测试实战项目
- leetcode 刷题实战训练营
- vite 从实战到手写实现训练营
- 低代码平台开发实战训练营
- 从零实现 taro 训练营
- TypeScript 高级进阶训练营
- 抽奖系统实战训练营
- Typescript+ReactHooks 实现全栈项目

- 持续集成 CI/CD 实战
- K8S 实战训练营
- 微信电商小程序实战
- Antv/G6 实现思维导图实战
- 可视化监控实战训练营
- lowcode 可视化拖拽组件训练营
- 从零实现脚手架 vue-cli 实战
- nestjs 基础到实战项目实战
- Typescript+React17 组件库实战
- Vue3+ts 组件库实战
- 前端全链路监控实战训练营
- uni-app 电商实战训练营
- Egg.js 从实战到源码训练营
- 浏览器插件实战训练营
- Vscode 插件开发实战训练营
- npm 私服搭建和 npm 知识体系训练营
- 电商网站抢购
- electron 视频系统实战训练营
- nest-serverless-graphql 训练营

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**

nestjs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**
- **protocal 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

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

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FormattedMessag和UFormattedNumber

Umi4从实战到源码

- Umi4核心实战
- Umi4 Max核心实战
- 实现UMI4核心架构
- 实现UMI4插件系统
- 实现UMI4核心插件

从零实现 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 前端工程化最佳实践

- 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

从零实现 Vue3+TS 版 element-plus 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

从零实现 Vue Loader V15

- 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

从零实现 React Hooks

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

从零实现 Formily2.0 核心组件

- 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

Vue3+TS 从零实现专业级虚拟树组件

- 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

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

实现 ESLint 插件

- 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

Formily2.0 从实现到原理-实战篇

- 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

前端 CSS-in-JS 方案从实战到原理

- 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

实现 React 官方动画库

- 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

实现蚂蚁标准请求 Hook 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

ReactQuery 实战篇

- 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

从零实现超强 React 虚拟列表组件

- 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

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 完全新特性实战和升级指南

- 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

核心课程

Vue.js 全家桶

Vue@2 源码篇

- 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...）

Vue@3 源码篇

- 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

Vue 路由源码篇

- 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

Vuex 源码篇

- 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...

VueSSR 原理篇

- 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

Vue@3 实战篇

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

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

组件化开发

- 什么是组件和组件的应用
- 组件的属性和校验
- 组件之间的通信
- EventBus 应用
- 组件 slot 用法

从零封装专业级 Vue3.0 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

路由篇(权限控制)

- vue router 的基础应用
- 编程式导航
- 路由的嵌套
- 路由重定向
- 路由守卫
- 路由懒加载
- 路由元信息
- 实现动态权限
- 菜单、按钮及权限认证、登录权限

Vue SSR 服务器端渲染

- 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

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

Vue2 & Vue3 核心应用进阶篇@2

- 组件 N 种通信方式
- 自定义指令，v-has 按钮级权限、v-lazy 实现图片懒加载
- Vue.extend 之 Vue 组件在线运行器
- Vue 中 JSX 语法高阶使用
- Vue 中递归组件之菜单权限
- SSR 之 Nuxt 实战
- Vue 页面预渲染、Vue 骨架屏@3
- Vue3 新特性 teleport、suspense 应用
- Vue3 中 CompositionAPI 详解
- 对比 Vue3 和 Vue2 中的区别和改进

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

Vue 面试进阶

- 通过源码深度剖析 40 道 Vue 面试题，再也不必担心面试题不会答了
- 从基础面试 -> 原理剖析 -> 项目优化 一应俱全

Node.JS

Node.js 核心模块

- 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

Node 框架

- 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

分片断点文件上传

- 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

webpack5 工程化

- grunt、gulp、webpack、rollup 和 parcel 的实战和对比
- webpack5 实战 entry、output、loaders、plugins、文件指纹
- webpack5 优化(dll、resolve、模块热替换、压缩、代码分割、可视化性能分析工具)
- Webpack5 源码分析,懒加载原理、热更新原理
- 编写自定义 loader(style-loader、css-loader、less-loader、babel-loader 等)
- 编写自定义 Plugin(html-webpack-plugin)
- 编写 Babel tree shaking 插件
- Webpack 的事件机制 tapable 和 AST 抽象语法树
- 手写自己的 Webpack 包括 make、seal、emit、 动态 import、splitChunks、支持 loader 和 plugin
- 从零实现 webpack 中的骨架屏插件
- 从零实现 webpack 优化神器之 DllPlugin
- 从零实现 Webpack 中的热更新(HMR)
- 从零实现 Webpack5 模块联邦原理并实现微前端
- webpack 实战、性能优化和源码流程等面试题训练营

React 全家桶

实战篇

- create-react-app 使用的深入配置
- JSX 本质
- 原生组件和自定义组件
- 自定义组件的属性和状态
- 事件处理
- 新旧生命周期
- 上下文和高阶组件
- 组件通信方式
- 全部的 Hooks

源码篇

手写 React15 源码

- 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

手写 React16 源码

- 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

手写 React17 源码

- 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

手写 React18 源码

优化篇

- JSX 原理和虚拟 DOM 原理
- setState 异步原理实现
- React 中的事务实现
- 使用 immutablejs 和 PureComponent
- ErrorBoundary、Suspense 和 Fragment
- React 中的高阶组件和 render props
- 实现 React 骨架屏 webpack 插件
- React 中的图片和组件懒加载
- React 中的长列表优化，只渲染可视区域 DOM
- Jest+Enzyme 实现 React 单元测试

路由篇

- 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)

Redux 篇

- 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)

手写 React 服务器端渲染 SSR

- 客户端渲染 VS 服务端渲染
- React 中的服务端渲染
- 同构的原理和意义
- SSR 中使用路由

- SSR 中使用 Redux
- SEO 优化
- 预渲染

Next.js 实战

- Next.js 路由和二级路由
- 实现样式集成
- 集成 koa
- 实现懒加载
- 集成 redux
- 调用接口
- 实现 loading 效果
- 注册登录和权限
- 新旧版初始化函数
- 服务器布署

mobx 篇

- mobx 实战 observable、computed、autorun、when、reaction
- 手写一个 mobx 类库

React 状态管理

- React 原生 Context API 实现状态管理
- React 官方出品状态库 Recoil 实战
- 手写 React 官方出品 Recoil 核心 API
- 有限状态机模式的工作流程和原理
- 基于有限状态机的状态库 XState 实战
- 手写实现 XState 核心 API

从零封装自己的 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 React-RouterV6.0 正式版

- 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

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

搭建 AntDesign4.0 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

实现 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

React18

- React 中的并发模式(concurrent mode)
- React 中的批量更新
- Suspense 和 SuspenseList
- startTransition 和 useDeferredValue

实现 React DnD

- HTML 拖放 API
- React DnD 核心概念
- React DnD 拖拽排序实战
- 实现实现 Provider 和 Monitor、Registry
- 实现 useDragSourceConnector、DragSource、DragType

React Router V6

- 路由基本原理
- 实现 history
- 实现路径参数
- 实现 Link 导航
- 实现嵌套路由和 Outlet
- 实现 NavLink
- 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

专题课

从零实现 React Hooks**

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

从零实现 Formily2.0 核心组件

- 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

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

Formily2.0 从实现到原理-实战篇

- 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

前端 CSS-in-JS 方案从实战到原理

- 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

实现 React 官方动画库

- 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

实现蚂蚁标准请求 Hook 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

ReactQuery 实战篇

- 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

从零实现超强 React 虚拟列表组件

- 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

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 完全新特性实战和升级指南

- 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

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FormattedMessag和UFormattedNumber

Umi4从实战到源码

- Umi4核心实战
- Umi4 Max核心实战
- 实现UMI4核心架构
- 实现UMI4插件系统
- 实现UMI4核心插件

从零实现 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 前端工程化最佳实践

- 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

从零实现 Vue3+TS 版 element-plus 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

从零实现 Vue Loader V15

- 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

Vue3+TS 从零实现专业级虚拟树组件

- 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

实现 ESLint 插件

- **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**

V8

- 计算机模型
- V8 的编译流水线和执行过程
- V8 的内存管理和垃圾回收
- V8 内存泄露分析和性能优化

低代码平台开发

- 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

gulp 实战到源码

- 编译样式、脚本和 HTML
- 压缩图片、拷贝静态文件、删除目录和合并压缩
- 自动加载插件、开发服务器和监听文件变化
- 实现 gulp
- 实现 undertaker 和 vinyl-fs
- 实现 gulp 常用插件

实现插件版 create-vite

- 实现配置命令
- 获取选择项和回答
- 支持下载模板
- 添加路由插件
- 解析并应用插件
- 脚手架发布

实现 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

函数式编程

- 函数式编程概念和核心
- 纯函数和柯里化
- 组合和 PointFree
- 函子高阶应用

前端网络面试题大汇总

- 七层网络协议
- tcp 和 udp
- 滑动窗口和懒启动
- https 原理
- http2 和 http3

微前端从实战到源码

- systemjs 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
- 模块联邦使用和工作原理

浏览器渲染原理和性能优化

- 浏览器渲染流程
- 浏览器资源加载和执行流程
- Performance API 应用
- 手写浏览器解析响应过程
- 前端性能优化大汇总

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

实现 immer 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

实现 css-loader

- css-loader 实战
- 实现 css-loader
- 支持@import 功能
- 实现 style-loader
- PostCSS 抽象语法树

实现 UMI3.0

- 约定式路由实战
- 源码调试 UMI3.0
- 实现 UMI3 的运行时系统
- 实现 UMI3 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

实现 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 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

实现 JSX 转换模块

- 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

实现实用的 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

实现@vue/cli 脚手架 4.0

- @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

实现 create-react-app

- 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

实现 rollup 和 tree-shaking

- 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

实现 px2rem-loader

- 移动端适配实践方案
- CSS 抽象语法树
- px2rem-loader 实战
- 如何编写自定义 loader
- 实现实现 px2rem-loader

实现 React 的 keepalive 组件

- 基于 fiber 实现 React 缓存组件
- 支持保持滚动状态
- 支持缓存生命周期管理
- 使用 LRU 管理缓存
  -

前端人的健康之道

- 颈腰椎病痛的原理及对策
- 肩周炎和鼠标手的原理和对策
- 干眼症的分级症状和对策
- 跑步之道
    - 科学跑姿
    - 跑前热身
    - 跑后拉伸

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

前端编译原理

- 编译器工作流程
- 有限状态机
- 语法分析
- 递归下降算法
- 上下文无关文法
- 语法分析器
- 遍历语法树
- 语法树转换器
- 代码生成器
- 实现 babel 和 babel 插件
- 从零实现 JSX 到 JS 的 babel 转换插件

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

前端二进制应用实战

- 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

手写 Nest.js 版的 IOC 容器

- 控制反转和依赖注入
- 实现服务的注册
- 实现值的获取
- 实现注入和获取

Typescript

- 开发环境
- TypeScript 安装和编译
- 数据类型
- 函数
- 类
- 接口
- 泛型
- 结构类型系统
- 类型变换
- 模块 VS 命名空间
- 类型声明
- 声明文件
- Typescript+React 集成开发
- Typescript+Vue 集成开发
- Typescript 工程化

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

从零实现 vue-cli

- 脚手架项目创建
- 解析命令行参数
- create 命令实现
- config 命令实现
- 项目发布和布署

前端监控和埋点系统

- 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
- 前端监控项目实战

serverless 云开发

- 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

微前端实战与原理

- 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
- 从零编写一个微前端框架
    - 应用状态管理
    - 应用的加载管理
    - 路由管理

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

GraphQL+Nest.js 微服务开发

- TypeScript+Nest.js 项目构建
- Nest.js 模块定义
- GraphQL 的服务器搭建与查询操作
- Nest 实现后端微服务

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
- electron 版珠峰课堂项目实战

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

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

阿里企业级开发框架 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

网络安全

- Web 漏洞的分析与防御
- XSS(跨站脚本攻击)
- CSRF(跨站请求伪造)

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

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

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

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

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

CI/CD 持续集成

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

Mongodb

- Mongodb 安装和使用
- Mongodb 的系统架构
- Mongodb 高级查询
- Mongodb 索引
- Mongodb 安全与权限
- mongoose 模块之 Schema
- mongoose 模块之 Model

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

express+mongoose 多人博客

- 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

Typescript+React+Redux 网校课堂

- 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

Egg.js+AntDesignPro+MySQL 开发企业级管理系统

- 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

React+Mongodb+websocket 开发多人聊天室

- 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

cheerio+mysql+nodemailer 开发个性化新闻爬虫

- 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

使用 canvas 开发开发 flappy bird

- 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

使用 three.js 开发微信小游戏跳一跳

- 基础信息属性配置
- 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
- 场景管理之场景

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

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)

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

项目实战训练营

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**

nestjs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**
- **protocal 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**
- **后台开发**

- **个人中心和支付**
- **数据可视化**
- **布署上线**

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**

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

从零实现脚手架 cli 实战

- 脚手架项目创建
- 解析命令行参数
- 实现 create 命令
- 实现 config 命令
- 项目发布和部署到 npm

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

就业与面试指导

- 阿里面试官就业面试指导
- 头条面试流程分享
- 头条四轮面试真题演练

软技能

- 大厂职级系统
- 大厂能力模型
- 大厂晋升流程
- 时间管理和精力管理