开课吧《WEB前端高级工程师》- 课程手册

模块一 ECMAScript 晋阶

第1章 - ES6 基础入门

- let 和 const
  - let 和 var 的差异
    - let 允许声明一个在作用域限制在块级中的变量、语句或者表达式
    - 块级作用域
    - var 声明的变量只能是全局或者整个函数块的
    - let 不能重复声明
    - let 不会被预解析
  - const 常量
    - 常量不能重新赋值
    - 不能重复声明
    - 块级作用域
    - const 不会被预解析
- 解构赋值
  - 对象的解构赋值
  - 数组的解构赋值
  - 字符串的解构赋值
- 展开运算符
  - 对象展开
  - 数组展开
- Set 对象
  - Set 对象的数据结构
  - Set 相关属性与方法
    - size 属性
    - clear()、delete()、get()、has()、add()
- Map 对象
  - Map 对象的数据结构
  - Map 相关属性与方法
  - size 属性
  - clear()、delete()、get()、has()、set()
- 函数新增扩展
  - 箭头函数
    - 箭头函数的各种写法
    - 箭头函数的 this 问题
    - 箭头函数的不定参问题
  - rest 参数设置
  - 参数默认值设置
- 新增数组扩展
  - Array.from()、Array.of()

    - find()、findIndex()、includes()
    - flat()、flatMap()
- 新增字符串扩展
    - includes(), startsWith(), endsWith()
    - repeat()
    - 模版字符串
- 新增对象扩展
    - 属性简洁表示法
    - 属性名表达式
- babel 使用
    - Babel 是一个 JavaScript 编译器
    - Babel 基本使用方法

第2章 - 基于 Class 的面向对象开发

- 面向对象编程的特点
    - 对象的组成
    - 对象的属性：状态特征的描述
    - 对象的方法：功能的实现
- 对象的创建
    - new Object()
    - 对象字面量：{}
    - 属性和方法的添加
    - 对象创建过程的封装
- 构造函数
    - 运算符new的执行过程和原理分析
    - this的使用
    - 构造函数 - 创建并初始化对象的函数
    - 构造函数书写规范
- prototype原型
    - 什么是prototype
    - prototype和 `__proto__`
    - 原型与原型链
    - 通过prototype实现公有属性和方法的复用和继承
    - prototype使用注意事项
- 包装对象
    - 什么是包装对象
    - 包装对象的作用
    - String、Number、Boolean
    - 字符串、数字、布尔与字符串对象、数字对象、布尔对象的区别及使用注意事项
- 对象常用操作
    - toString()
    - toString()的重写(overWrite)和实现过程
    - hasOwnProperty()方法实现自有属性判断
- for...in/for...of的使用及特点
- constructor属性的使用
- instanceof运算符
- 继承

    - 继承的特点
    - 继承的实现
        - 构造函数继承
        - 原型继承
        - 拷贝继承
            - 浅拷贝
            - 深拷贝
- ES6 中的面向对象
- class 关键字
- extends 关键字 - 继承
- super 关键字
- 静态属性、static关键字
- 工厂模式及观察者模式
- 单例模式使用
- 装饰者模式使用
- 案例
    - 王者荣耀英雄选择案例
- 组件介绍
    - 什么是组件?
    - 组件的特点
        - 方法、配置、事件
- 组件方法的作用和实现
- 组件配置的作用和实现
    - 配置的作用
    - 配置的实现
    - 实例配置和默认配置 - extend()
- 组件事件的作用和实现
    - 事件的作用
    - 基于属性的事件的实现
    - 基于属性的事件的弊端
    - addEventListener事件机制的实现
    - 事件监听器addEventListener
    - 事件触发器trigger
    - 事件容器
- 事件继承
- 预定义EventTarget类
- webComponent自定义组件
- 案例
    - 自定义弹窗组件封装
- 编写自己的工具库
- 闭包、自执行
- 对象成员与类成员
- 基于类的工具封装

    - 判断浏览器
    - 判断类型
- 基于对象的工具封装
    - 元素结果集包装
    - 常用 DOM 操作封装
    - 链式调用实现
- 扩展插件实现
    - 对象扩展
    - 类扩展
- 案例
    - jquery核心功能实现

第3章 - ES6 高阶

- 同步及异步概念
    - 同步和异步是一种消息通知机制
- promise的用法
    - 两种参数
    - 三种状态
    - then方法
    - then的返还值
- Async 函数 和 await
- Promise下的常用方法
    - resolve、reject、all、race
- try及catch方法
- 案例
    - 基于 Promise 封装的链式动画实现
- Object.defineProperty()
    - enumerable
    - configurable
    - set、get
- Object.defineProperties()
- assgin() 合并对象
- Object.is()
- Proxy
- 模块化编程
    - CMD 规范 和 AMD
- export 和 import
- 发布订阅模式
- mvvm框架中编译数据到视图
- 案例
    - 简版mvvm框架实现
- [扩展] Generator 和 Iterator

第4章 - [扩展]正则表达式

- 字符串操作
  - 查找字符串中的数字；
  - 正则如何如实现
- 正则的创建
  - 字面量创建
  - 通过构造函数
- 正则的匹配方法
  - 字符串方法
    - match
    - search
    - replace
    - split
  - 正则对象下的方法
    - test
    - exec
- 元字符
  - 正则表达式中有特殊含义的非□字母字符； - 字符类别（Character Classes）
  - `.`
  - 匹配行结束符（\n \r \u2028 或 \u2029）以外的任意单个字符
  - 在 `字符集合（Character Sets）` 中，. 将失去其特殊含义，表示的是原始值
  - `\`
  - 转义符，它有两层含义
    - 表示下一个具有特殊含义的字符为字面值
    - 表示下一个字符具有特殊含义（转义后的结果是元字符内约定的）
  - `\d` 匹配任意一个阿拉伯数字的字符
  - `\D` 匹配任意一个非阿拉伯数字的字符
  - `\w` 匹配任意一个（字母、数字、下划线）的字符
  - `\W` 匹配任意一个非（字母、数字、下划线）的字符
  - `\s` 匹配一个空白符，包括空格、制表符、换页符、换行符和其他 Unicode 空格
  - `\S` 匹配一个非空白符
- 字符集合
  - `[xyz]` 一个字符集合，也叫字符组。匹配集合中的任意一个字符。你可以使用连字符'-'指定一个范围
  - '`[xyz]`' 是一个反义或补充字符集，也叫反义字符组。也就是说，它匹配任意不在括号内的字符。你也可以通过使用连字符 '-' 指定一个范围内的字符
- 边界
  - `^`
  - 匹配输入开始。如果多行（multiline）标志被设为 true，该字符也会匹配一个断行（line break）符后的开始处
  - `$`
  - 匹配输入结尾。如果多行（multiline）标志被设为 true，该字符也会匹配一个断行（line break）符的前的结尾处
  - `\b`

    - 匹配一个零宽单词边界（zero-width word boundary）
    - `\B`
    - 匹配一个非零宽单词边界（zero-width word boundary）
- 分组
    - (子项)
    - 可以使用 () 对表达式进行分组，类似数学中分组，也称为子项
    - 索引分组
    - 命名分组
        - (?...)
        - groups属性
    - 捕获匹配
        - 具有捕获（capturing）特性，即会把匹配结果保存到（子项结果）中
        - (x)
    - 非捕获匹配
        - 不具有捕获（capturing）特性，即不会把匹配结果保存到（子项结果）中
        - (?:x)
    - 零宽断言/预查（Assertions）
        - 用于指定查找在某些内容(但并不包括这些内容)之前或之后的内容
        - 正向零宽断言/预查
        - 肯定
            - (?=pattern)
        - 否定
            - (?!pattern)
        - 负向零宽断言/预查（注意：ES2018新增）
        - 肯定
            - (?<=pattern)
        - 否定
            - (?<!patten)
    - 捕获与零宽断言的区别
        - 捕获：匹配的内容出现在结果中但不出现在子项结果中
        - 零宽断言：完全不会出现在结果
- 反向引用
    - `\n`
        - 这里的 n 表示的是一个变量，值为一个数字，指向正则表达式中第 n 个括号（从左开始数）中匹配的子字符串
- 数量词汇
    - `x{n}`
        - n 是一个正整数。前面的模式 x 连续出现 n 次时匹配
    - `x{n,m}`
        - n 和 m 为正整数。前面的模式 x 连续出现至少 n 次，至多 m 次时匹配
    - `x{n,}`
        - n 是一个正整数。前面的模式 x 连续出现至少 n 次时匹配
    - `x-`
        - 匹配前面的模式 x 0 或多次

    - `x+`
        - 匹配前面的模式 x 1 或多次。等价于 {1,}
    - `x?`
        - 匹配前面的模式 x 0 或 1 次
    - `x|y`
        - 匹配 x 或 y
- 匹配模式
    - `g`
        - global，全局模式：找到所有匹配，而不是在第一个匹配后停止
    - `i`
        - ignore，忽略大小写模式：匹配不区分大小写
    - `m`
        - multiple，多行模式：将开始和结束字符（^和$）视为在多行上工作，而不只是匹配整个输入字符串的最开始和最末尾处
    - `s`
        - dotAll / singleline模式：. 可以匹配换行符
    - `u`
        - unicode，unicode模式：匹配unicode字符集

        ```
        console.log(/^.$/.test("\uD842\uDFB7"));
        console.log(/^.$/u.test("\uD842\uDFB7"));
        ```

    - `y`
        - sticky，粘性模式：匹配正则中lastIndex属性指定位置的字符，并且如果没有匹配也不尝试从任何后续的索引中进行匹配
- 正则工具

模块二 前后端数据交互

第5章 - [扩展]客户端存储

- cookie 简介
    - 响应头信息
        - set-cookie
    - 请求头信息
        - cookies
    - cookie 属性
        - key：名称
        - value：值
        - expires / max-age：保存时间
        - http-only：安全保护
    - 客户端浏览器 cookie 接口
        - document.cookie
    - 实例：
        - 用户注册登陆
- storage - 本地存储

    - localStorage
        - setItem方法
        - getItem方法
        - removeItem方法
        - clear方法
    - sessionStorage
        - setItem方法
        - getItem方法
        - removeItem方法
        - clear方法
    - storage 事件
    - localStorage 与 sessionStorage 差异
    - 实例：
        - 共享购物车
- Application Cache
    - 简介
    - manifest 属性
    - 缓存清单
        - text/cache-manifest 头信息
        - CACHE字段
        - NETWORK字段
        - FALLBACK字段
    - 缓存状态
        - UNCACHED(未缓存)
        - IDLE(空闲)
        - CHECKING(检查)
        - DOWNLOADING(下载中)
        - UPDATEREADY(更新就绪)
        - OBSOLETE(废弃)
    - 事件
        - cached
        - checking
        - downloading
        - noupdate
        - obsolete
        - updateready
- cookie 与 storage 的差异

第6章 - NodeJs

- Node.js介绍
- 环境搭建
- 模块化
    - CommonJS 规范
    - 模块加载
        - require 方法 - 导入
        - module 对象

        - exports 对象 - 导出
    - 模块分类
        - 文件模块
        - 文件夹模块
        - 核心模块
        - 第三模块（node_modules）
        - 模块加载机制
            - 相对模块（文件模块、文件夹模块）
            - 绝对模块（第三方模块、核心模块）
- NPM 包管理工具
    - 常用命令
        - init：初始化
        - search：查找
        - install：安装
        - update：更新
        - remove：删除
    - 创建 package 模块
    - package.json 文件
        - name：包名
        - version：版本
        - main：入口程序
        - scripts：执行脚本
        - dependencies：运行依赖
        - devDependencies：开发依赖
    - 注册与发布
        - 注册账号：https://www.npmjs.com/
        - 发布包
            - publish 命令
            - unpublish 命令
- Koa 介绍
- 项目创建
- Koa 安装
- Koa 使用
    - 实例化
        - Application 对象
        - Context 对象
    - 请求
        - Request 对象
    - 响应
        - Response 对象
    - HTTP 协议
        - 头信息
            - 简介
            - Content-Type
        - 状态码

        - 200
        - 404
        - 301、302 - location 头
      - 中间件
        - use 方法
        - next 方法
        - 异步 async
- 实例：
  - 文章信息展示
  - node.js - url 模块
  - node.js - fs 模块
  - node.js - queryString 模块
- 第三方中间件
  - 路由中间件 - Koa-Router
    - use方法
    - get、post
    - redirect方法：重定向
  - 正文解析中间件 - Koa-body
    - 解析类型
      - form
      - json
      - text
  - 静态文件加载koa-static
- mysql数据库
  - 数据持久化
  - mysql数据库安装
  - mysql操作
    - 命令操作
    - 图形化操作
    - 代码操作
  - sql语句
    - 增、删、改、查
  - nodejs中mysql2模块操作mysql
- 前端轮循获取数据
- SSE (server send event) 服务器推送数据;
- websocket协议
- nodejs中使用socket.io模块实现服务端推送；

第7章 - 前后端交互

- XMLHttpRequest 对象
  - open方法
    - 请求类型
    - url
    - 同步与异步

    - send方法
        - 发送请求
        - get请求与post请求
            - querystring
                - 编码与缓存
            - 请求正文
            - setRequestHeader方法
                - application/x-www-form-urlencoded
    - 事件
        - onload
        - onreadystatechange
    - 属性
        - status：状态码
        - responseText：响应文本
        - responseXML：XML类型
    - ajax 封装
        - 请求封装
        - 请求数据封装
        - 响应数据解析数据封装
- 实例：
    - ajax 注册与登陆
- Ajax 上传实现
- Ajax子集upload使用
- FormData对象
    - append方法
    - content-type 头信息：multipart/form-data
- upload 事件
    - onloadstart
    - onprogress
    - onabort
    - onerror
    - onload
    - ontimeout
    - onloadend
- 实例：
    - 无刷新上传，进度监控、速度计算
- 跨域请求
- 同源策略
- 跨域的问题和常用解决方式
- JSONP
    - JSONP 的概念
    - JSONP 的原理
    - JSONP 的实际应用
- CORS
    - 跨域资源共享标准 - cross-origin sharing standard

    - 简单请求
        - 请求方法求头信息
        - Content-Type 字段值
    - CORS 预检请求
        - 请求方法
        - 头信息
        - Content-Type 字段值
    - HTTP 请求首部字段
        - Origin
        - Access-Control-Request-Method
        - Access-Control-Request-Headers
    - HTTP 响应首部字段
        - Access-Control-Allow-Origin
        - Access-Control-Expose-Headers
        - Access-Control-Max-Age
        - Access-Control-Allow-Credentials
        - Access-Control-Allow-Methods
        - Access-Control-Allow-Headers
- 后端代理
    - 基于node.js的proxy
    - 请求转发
- Fetch 基本用法 Request 对象 Headers 对象 Response对象
    - text方法
    - json方法
- Fetch 与 XMLHttpRequest 的差异
- axios
    - 拦截器
    - 适配器
    - 扩展模式
    - 案例
        - 简版axios功能实现

模块三 工程化开发

第8章 Webpack

- 模块化回顾
    - 模块化核心
    - ESM
        - 模块作用域
        - import 与 export
    - CommonJS
        - 模块作用域
        - module、module.exports

        - require
    - AMD
        - define
        - return 导出
        - 前置导入语法
    - UMD
        - 模块同构
- webpack 安装
- webpack 基础打包功能与结构分析
- 配置
    - entry
    - output
    - loaders
    - webpack 中的 module
        - file-loader：静态文件资源处理模块
        - url-loader：基于 file-loader 的 url 处理模块
        - 样式处理：
            - css-loader：css 样式处理模块
            - style-loader：style 样式注入模块
            - sass-loader：css 预处理器模块
        - plugins
            - HtmlWebpackPlugin：html 文件构建
            - clean-webpack-plugin：打包文件清理
            - mini-css-extract-plugin：css 文件提取
    - sourceMap
    - WebpackDevServer
    - Hot Module Replacement (HMR:热模块替换)
        - 模块热替换原理与逻辑实现

第9章 TypeScript

- TypScript 介绍
- TypeScript 环境配置
- TypeScript 编译命令
- TypeScript 编译配置
    - tsconfig.json
- 类型系统
    - 类型标注
    - 类型检测
    - 基础类型
- 接口的使用
- 类型声明文件
    - @types/**
- 模块系统

    - 模块导出
    - 模块导入
    - TypeScript 的模块系统的规则与编辑
        - ES6 Modules
        - CommonJS
        - AMD
        - UMD
    - 内部模块
        - 命名空间：namespace
- TypeScript 在 React 中的使用
- 装饰器
    - 装饰者模式
    - —experimentalDecorators 配置
    - 装饰器语法
        - 装饰器使用限制
    - 装饰器分类
        - 类装饰器
        - 属性装饰器
        - 方法装饰器
        - 参数装饰器
        - 访问装饰器
    - 装饰器实现
        - 装饰器工厂
        - 装饰器求值
    - 装饰器组合
    - 元数据
        - metadata
        - reflect-metadata 库
        - —emitDecoratorMetadata 配置
- TypeScript 在 Vue 中的使用
- 基于 TypeScript 进行 Koa 的二次web框架封装

第10章 git 版本控制工具

- git 简介与安装
    - 版本控制系统
    - window 及 mac安装
- 入门命令
    - 创建仓库 init
    - 查看文件状态 status
    - 追踪文件 add
    - 提交 commit
    - 查看提交状态 log
    - 删除文件 rm
    - 移动文件 mv
    - 查看修改 diff
- 文件的三种状态及三个工作区域

    - 三种状态：已修改、已暂存、已提交
    - 工作区域工作区、暂存区、仓库
- git工作流程
- 中文乱码的处理
- git 的分支管理
    - master
    - 可变指针
    - 创建分支 branch
    - 切换分支 checkout
    - 创建并切换分支 -b
    - 合并分支 merge
    - 合并分支 rebase
        - merge 和 rebase 的区别
    - 快速前移
        - 快速前移产生的问题
        - 禁止快速前移 --no-ff
    - 分支冲突的产生及解决
    - 删除分支 -d / -D
    - 标签操作 tag
- git 版本控制
    - 取消合并 --abort
    - 撤销提交 --amend
    - 取消暂存 reset
    - 撤销文件修改 checkout
- git存储
    - 存储暂存区域内容 stash
    - 查看存储内容 stash list
    - 取出存储内容 stash apply
    - 移除存储 stash drop
- 简化操作
    - 命令别名 alias
    - .gitignore 配置文件
- git 远程仓库同步
    - 提交到远程仓库 push
        - 不同分支的推送
        - 分支的删除
        - 标签的推送
        - 标签的删除
    - 克隆远程仓库至本地 clone
        - 克隆分支
    - 拉取更新 pull
    - SSH 密钥 的生成及使用

模块四 Vue 全家桶

第11章 Vue.js 全家桶

- new Vue()
  - el 选项
  - data 选项
- 模板渲染
  - {{}} 语法
- 指令
  - 输出
    - v-text、v-html.....
  - 属性绑定
    - v-bind
    - 样式：class 与 style 绑定
  - 条件渲染
    - v-if、v-else、v-else-if、v-show
  - 列表渲染
    - v-for
  - 事件绑定
    - v-on
      - 参数
        - event 对象
      - 修饰符
  - 表单
    - v-model：双向数据绑定
- 自定义指令
  - 全局 Vue.directive && 局部 directives 选项
  - 钩子函数
  - 钩子函数参数
- 过滤器
  - 全局 Vue.filter && 局部 filters 选项
  - 管道符：|
- 计算属性
  - computed
  - getter
  - setter
- 侦听器
  - watch
- 组件基础
  - 全局 Vue.component && 局部 components 选项
    - 组件名称约定
    - 选项
      - template：组件模板
        - 顶层节点
      - data：组件私有数据 - （React -> state），必须是函数
    - 组件数据通信基础

        - props
        - event
            - $emit
    - 插槽
        - 内置 <slot> 组件
        - slot属性
        - 具名插槽
            - name 属性
        - 插槽作用域
- props 验证
    - 类型检查
    - 非 props 特性
    - 继承特性
    - 禁用继承特性
- v-model
    - model 选项
        - value
        - event
- .sync 修饰符

- vue-cli
    - 安装：npm install -g @vue/cli
    - 常用命令
        - 通过命令行创建应用：vue create <应用名称>
        - 通过 UI 界面方式创建应用：vue ui
        - 启动本地应用：npm run serve
        - 打包：npm run build
    - 单文件组件
        - template
        - script
        - style
        - lang 属性
- 组件深入
    - 组件生命周期
        - beforeCreate
        - created
        - beforeMount
        - mouted
        - beforeUpdate
        - updated
        - beforeDestroy
        - destroyed
    - 动态组件
        - 内置 <component> 组件
        - 内置 <keep-alive> 组件
        - 生命周期

        - activated
        - deactivated
- 动画
    - 过渡条件
        - v-if、v-show、组件根节点、动态组件
    - 过渡管理操作
        - css
        - js
    - css 过渡
        - 过渡类名
        - 内置 <transition> 组件
            - name 属性
    - js 过渡
        - 过渡钩子
- 插件
    - Vue.use 方法
    - install 方法
    - Vue.mixin 的用法
- 路由介绍
- 路由安装
    - npm install vue-router
- 实例化
    - new VueRouter
    - 选项
        - base：基础路径
        - mode：模式
            - history
            - hash
        - routes：路由对象
            - path：路由路径
            - component：路由绑定的组件
            - name：路由名称
- <router-view> 组件
- <router-link> 组件
    - to 属性
    - replace 属性
    - append 属性
    - tag 属性
    - event 属性
    - active-class 属性
    - exact 属性
    - exact-active-class 属性
- 动态路由
    - router 选项
    - $router：全局router对象

        - 跳转路由：push、replace
    - $route：当前路由对象
        - 获取路由数据
            - params 属性，动态路由
            - query 属性，获取 queryString
- 导航守卫
    - 路由导航解析流程
    - 全局守卫
        - beforeEach
            - to 属性
            - from 属性
            - next 函数
        - beforeResolve
        - afterEach
    - 路由独享守卫
        - beforeEnter
    - 组件内守卫
        - beforeRouteEnter
        - beforeRouteUpdate
        - beforeRouteLeave
- 路由元信息
    - meta 属性
- 路由懒加载
    - import 方法
    - 分组
        - webpackChunkName
- 组件通信
    - props/$emit（父子通信）
    - $refs/ref（父子通信）
    - $children/$parent（父子通信）
    - $attrs/$listeners（父子通信、跨级通信）
    - provide/inject（父子通信、跨级通信）
    - eventBus（父子通信、跨级通信、兄弟通信）
    - vuex（父子通信、跨级通信、兄弟通信、路由视图级别通信）
    - localStorage/sessionStorage等基于浏览器客户端的存储（父子通信、跨级通信、兄弟通信、路由视图级别通信）
- Vuex 介绍
- Vuex 安装
    - npm install vuex
- 实例化
    - new Vuex.store
    - state 属性
        - 元数据存储仓库
    - getter 属性
        - 派生数据存储仓库，类似组件计算属性

- mutation 属性
  - 存储动作，用来修改仓库中的数据
    - state，payload
  - 同步操作
  - 无返回结果
- action 属性
  - 类似 mutation
  - 异步操作
  - 返回 Promise
- 辅助函数
  - mapState
  - mapGetters
  - mapMutations
  - mapActions
- Module
- namespace
  - createNamespacedHelpers

- vue3
  - 构建
    - vue-next-webpack-preview
    - vue-cli-plugin-vue-next
    - Vue Composition API
  - new 与 createApp
  - Fragment
  - OptionsAPI 的弊端
  - CompositionAPI
    - setup 函数
      - props 与 context
    - 响应式数据定义
      - reactive 函数
      - ref 函数
      - computed 函数
      - readonly 函数
      - watchEffect 与 watch 函数
    - 生命周期
      - setup
      - onBeforeMount
      - onMounted
      - onBeforeUpdate
      - onUpdated
      - onBeforeUnmount
      - onUnmounted
      - onErrorCaptured
      - onRenderTracked
      - onRenderTriggered
    - 依赖注入（DI）

        - provide 函数
        - inject 函数
      - Template Ref
      - 工具函数
        - unref、 toRef、 toRefs、isRef、isProxy、isReactive、isReadonly
    - Teleport 与 Suspense
    - 杂项
      - v-model 的新变化
      - 新的自定义指令规范
      - 新的过滤器
      - 其它随时更新的特性

第12章 Vue 实战 - 美食杰电商

- Vue 实战美食杰电商

第13章 Vue + TS 实战 - trello

- Vue + TS - 《trello》

模块五 React 全家桶

第14章 - React 全家桶

- React 简介
  - 命令式编程 和 声明式编程
  - 组件化开发
  - 专注视图层
- JSX
  - 插值表达式
  - 各类型数据在插值中的使用
  - 列表渲染
  - 条件渲染
  - JSX 使用注意事项
- create-react-app
  - 脚手架安装
  - 脚手架使用
  - 项目结构说明
- Component 组件
  - 类组件定义
  - 组件调用
- 组件间通信
  - props 和 state
  - 单项数据流
  - 父组件向子组件通信
  - 子组件向父组件通信
  - context 跨组件通信
- Component 的生命周期
  - 生命周期过程

        - 组件挂载阶段
        - 组件更新阶段
        - 组件卸载阶段
    - 生命周期演变史
    - PureComponent
- React 中的 DOM 操作
    - Refs
        - string Ref
        - createRef()
    - dangerouslySetInnerHTML
- keys
    - key 取值时的注意事项
- 事件系统
    - React 中的合成事件
    - 在 React 中使用原生事件
- 表单
    - 受控组件
    - 非受控组件
- 函数式组件
    - 无状态组件
- React Hooks
    - 常用 hook
    - React 其他 hook
    - hooks 使用原则
    - 自定义 hook
- React-router 使用
    - 路由和前端路由
    - 单页面应用 - SPA
    - 安装 react-router-dom
    - HashRouter 和 BrowserRouter
    - Route 组件
        - 路由匹配规则
            - 默认匹配规则
            - exact 精确匹配
        - render 和 component
    - Switch 组件
    - Link 组件
    - NavLink 组件
        - activeClass
        - activeStyle
        - isActive
    - Redirect 组件
    - 路由参数
        - history

        - match
        - location
    - withRouter
    - 动态路由
    - router hooks
        - useHistory
        - useLocation
        - useParams
        - useRouteMatch
- redux
    - 为什么使用 redux
    - redux 核心概念
        - state
        - reducer
        - store
        - action
    - createStore() 方法
        - combineReducers
        - applyMiddleware
    - store 相关方法
        - getState()
        - dispatch(action)
        - subscribe(listener)
        - replaceReducer(nextReducer)
    - redux-thunk
        - 对象参数
        - 函数参数
        - 利用 thunk 中间件对 react 进行异步操作
- react-redux
    - react-redux 安装
    - `<Provider store>`
    - connect()
    - react-redux hooks
        - useDispatch 获取 dispatch
        - useStore 获取 store
        - useSelector 获取 state

第15章 - React+Antd实战

- CNode 项目

第 16 章 - 移动端事件专题

- touch 事件
- touch 事件与 mouse 事件的区别

    - mouse 事件延迟问题
    - touch 事件点透
- 移动端阻止默认事件的问题
    - start 阶段阻止默认事件
    - move 阶段阻止默认事件
    - end 阶段阻止默认事件
- touchEvent
    - touches
    - targetTouches
    - changedTouches
- 滑屏交互实现
    - move 过程中取消默认事件带来的问题
- orientationchange
    - orientation 横竖屏检测
    - Media 监听横竖屏的问题
- devicemotion
    - window.DeviceMotionEvent
    - acceleration
    - accelerationIncludingGravity
    - IOS 和 安卓下的取值问题
    - IOS http 协议问题
    - IOS 12 下的授权问题
    - IOS 13 下的授权问题
- 摇一摇功能封装
- 防抖函数实现
- 节流函数实现
- [扩展] 移动端多指操作和Tap事件
    - 安卓下多指操作兼容
    - 自定义 Tap 事件
- [扩展] deviceorientation
    - 类 VR 应用实现
    - 淘宝造物节解析

第17章 - [扩展]better-scroll

- better-scroll 基础使用方法
    - new BScroll(wrap)
    - better-scroll 原理说明
    - 使用 better-scroll 时注意问题
    - 利用 better-scroll 实现各种滑屏
    - better-scroll 基础使用方法
        - startX、startY
        - scrollX、scrollY、freeScroll
        - eventPassthrough
        - bounce、bounceTime
        - momentum
        - refresh() 方法

- 案例：自定义滚动条
  - scrollbar 高级组件
- 案例：手机淘宝幻灯片实现
  - snap 高级组件
  - probeType 和 scroll 事件
  - scrollEnd 事件
  - getCurrentPage()
- 案例：仿 iso 经典选择器控件
  - wheel 高级组件
  - getSelectedIndex()
- 案例：上拉加载和下拉刷新功能实现、
  - pullDownRefresh 和 pullUpLoad 高级组件
  - finishPullDown()、openPullDown(config)、closePullDown()
  - finishPullUp()、openPullUp(config)、closePullUp()
  - pullingDown、pullingUp 事件
- 案例：自定义索引列表

第18章 - React Hooks 移动端实战

- React Hooks 移动端实战

模块六 数据可视化

第19章 - Canvas

- Canvas概述
  - Canvas 作用
    - 绘图
    - 图像编辑
    - 游戏
    - 动画
    - 视频处理
  - Canvas 是什么
    - 广义来说，Canvas 是 h5 新增的Canvas 2d 绘图功能
    - 具体来说，在html 中，Canvas 既是标签，也是画布，用javascript 可以在Canvas 中绘制图形
  - 基本绘图流程
    - 建立画布
    - 获取上下文对象
    - 设置颜色
    - 设置图形
    - 绘制图形
  - 案例：矩形
- 绘制图形
  - Canvas 的坐标系和栅格
  - 矩形
    - fillRect(x, y, width, height)

        - strokeRect(x, y, width, height)
        - clearRect(x, y, width, height)
    - 路径
        - 直线 lineTo(x,y)
        - 圆弧 arc(x, y, radius, startAngle, endAngle, anticlockwise)
        - 切线圆弧 arcTo(x1, y1, x2, y2, radius)
        - 三次贝塞尔 bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
        - 二次贝塞尔 quadraticCurveTo(cp1x, cp1y, x, y)
        - 矩形 rect(x, y, width, height)
    - 案例：机器人
    - 案例：水滴
- 样式设置
    - 图形着色的区域
        - 描边区域
        - 填充区域
        - if(){} else if() {}
        - if(){} else if() {} else {}
    - 图形着色的方式
        - 纯色
        - 渐变色
        - 图案
    - 描边的样式
        - 着色样式 strokeStyle
        - 线宽 lineWidth
        - 线条末端样式 lineCap
        - 线的拐角样式 lineJoin
        - 拐角最大厚度 miterLimit， 只适用于lineJoin='miter' 的情况
        - 虚线样式 setLineDash(segments)
        - 返回虚线样式 getLineDash()
        - 虚线偏移量 lineDashOffset = value
    - 投影
        - 位置
            - shadowOffsetX
            - shadowOffsetY
        - 模糊度 shadowBlur
        - 颜色 shadowColor
    - 案例：霓虹灯
- 文本设置
    - 文本样式
        - 字体 字号 ... font
        - 水平对齐 textAlign
        - 垂直对齐 textBaseline
    - 绘制方法
        - 填充文字 fillText(text, x, y [, maxWidth])
        - 描边文字 strokeText(text, x, y [, maxWidth])
    - 案例：布艺文字
- 图像操作

- 建立图像源
  - HTMLImageElement
  - HTMLVideoElement
  - HTMLCanvasElement
- 绘制图像
  - 绘图 + 位移 drawImage(image, x, y)
  - 绘图 + 位移 + 缩放 drawImage(image, x, y, width, height)
  - 绘图 + 裁切 + 位移 + 缩放 drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)
- 案例：gif 动画

- 像素级操作
  - ImageData
    - data
    - width
    - heidth
  - 新建 ImageData() 对象
    - new ImageData()
      - new ImageData(width, heidth)
      - new ImageData(Uint8ClampedArray, width, heidth)
    - ctx.createImageData()
      - ctx.createImageData(width, heidth)
      - ctx.createImageData(ImageData)
  - 获取Canvas 的ImageData() 对象：ctx.getImageData(x, y, width, height)
  - 在Canvas 中显示ImageData：putImageData(ImageData, dx, dy, dirtyX, dirtyY, dirtyWidth, dirtyHeight)
  - 遍历像素的方法：
    - 逐像素遍历
    - 逐行列遍历
  - 案例：图像置灰
  - 案例：图像加马赛克

- 变换
  - 状态管理
    - save() 保存上下文对象的所有状态
    - restore() 恢复上一次保存的上下文的状态
  - 移动 translate(x, y)
  - 旋转 rotate(angle)
  - 缩放 scale(x, y)
  - 案例：时钟

- Canvas 动画
  - 制作动画的基本步骤
  - 驱动动画的方法
  - 速度加速度
  - 弹性运动
  - 补间动画
  - 用户交互

- 案例：粒子计时器
- 合成
  - 透明度合成 globalAlpha
  - 路径裁剪 clip()
  - 全局合成 globalCompositeOperation
  - 案例：刮刮乐
- Canvas 实战
  - 柱状图
  - 饼图

第20章 - ECharts

- echarts入门
  - echarts 概述
    - 浏览器的画图方式
    - 官网展示
  - 快速上手
    - 下载 ECharts
    - 引入 ECharts
    - 建立dom 容器
    - 基于dom 容器，初始化echarts实例
    - 建立图表配置项
    - 根据图表配置项，显示图表
  - 案例：折线图
- 常用组件
  - 标题 title
    - 主标题 text
    - 副标题 subtext
    - 位置 left right top bottom
    - 边框颜色 borderColor
    - 边框宽度 borderWidth
    - 可见性 show
  - 图例 legend
    - 数据 data
    - 图例朝向 orient
    - 位置 left right top bottom
    - 可见性 show
  - 工具箱 toolbox
    - 功能：feature
      - 保存为图片 saveAsImage
      - 配置项还原 restore
      - 数据视图工具 dataView
      - 数据区域缩放 dataZoom
      - 动态类型切换 magicType
  - 提示框 tooltip
    - 触发类型：trigger
      - item
      - axis

    - 系列列表 series
        - 标记点 markPoint
        - 标记线 MarkLine
    - 案例：柱状图
- 常用图表
    - 饼图 pie
    - 仪表盘 gauge
    - 散点 scatter
    - K 线 candlestick
    - 雷达 radar
    - 地图 map
- 高级应用
    - 多坐标轴
    - 异步数据
    - 数据集 dataset
    - 区域缩放 dataZoom
    - 视觉映射 visualMap
    - 事件
    - 富文本标签
- 扩展知识
    - webpack 中使用echarts
    - 渲染器
    - 三维 echarts
    - 微信中使用echarts
- 实战
    - 数据可视化大屏

第21章 - [扩展]拖放操作和FileReader

- 元素拖拽
    - draggable 设置元素为可拖放
    - dataTransfer.setData() 设置拖拽时传递信息
    - dataTransfer.getData() 设置拖拽时传递信息
    - 拖放相关事件
        - ondragstart 拖动开始时触发
        - ondrag 拖动过程中触发
        - ondragend 在拖动结束时触发
        - ondragenter 拖动的元素进入放置目标时触发。
        - ondragleave 当拖动的元素在放置目标中移出时触发。
        - ondragover 当元素或文本选择在放置目标中移动时触发。
        - ondrop 在有效放置目标上放置元素或文本选择时触发
- 系统文件的拖放操作
    - dataTransfer.items、dataTransfer.files
    - webkitGetAsEntry 和 isDirectory
    - createReader() 和 readEntries 读取文件夹 目录
- FileReader 读取文件信息
    - 事件处理
        - FileReader.onabort

        - FileReader.onerror
        - FileReader.onload
        - FileReader.onloadstart
        - FileReader.onloadend
        - FileReader.onprogress
    - 相应读取方法
        - FileReader.abort()
        - FileReader.readAsArrayBuffer()
        - FileReader.readAsDataURL()
        - FileReader.readAsText()
- 案例：系统文件拖拽显示

第22章 - [扩展]音频、视频操作

- 标签
    - audio 、video
    - source
- 支持格式说明
    - 音频：MP3、OggV
    - 视频：Ogg、MP4
- 媒体属性设置
    - controls: 显示或隐藏用户控制界面
    - autoplay: 媒体是否自动播放
    - loop: 媒体是否循环播放
    - currentTime: 开始到播放现在所用的时间
    - duration: 媒体总时间(只读)
    - volume: 0.0-1.0的音量相对值
    - muted: 是否静音
    - autobuffer: 开始的时候是否缓冲加载，autoplay的时候，忽略此属性
    - paused: 媒体是否暂停(只读)
    - ended: 媒体是否播放完毕(只读)
    - error: 媒体发生错误的时候，返回错误代码 (只读)
    - currentSrc: 以字符串的形式返回媒体地址(只读)
- Video 额外属性
    - poster: 视频播放前的预览图片
    - width、height: 设置视频的尺寸
    - videoWidth、 videoHeight: 视频的实际尺寸(只读)
- 媒体方法
    - play(): 媒体播放
    - pause(): 媒体暂停
    - load(): 重新加载媒体
- 全屏显示
    - elem.requestFullscreen()
    - elem.webkitRequestFullScreen()
    - elem.mozRequestFullScreen ()
- 相关事件
    - canplay 当浏览器可以播放音频/视频时
    - canplaythrough 当浏览器可在不因缓冲而停顿的情况下进行播放时
    - durationchange 当音频/视频的时长已更改时
    - error 当在音频/视频加载期间发生错误时

    - loadeddata 当浏览器已加载音频/视频的当前帧时
    - loadedmetadata 当浏览器已加载音频/视频的元数据时
    - loadstart 当浏览器开始查找音频/视频时
    - pause 当音频/视频已暂停时
    - play 当音频/视频已开始或不再暂停时
    - playing 当音频/视频在已因缓冲而暂停或停止后已就绪时
    - progress 当浏览器正在下载音频/视频时
    - ratechange 当音频/视频的播放速度已更改时
    - seeked 当用户已移动/跳跃到音频/视频中的新位置时
    - seeking 当用户开始移动/跳跃到音频/视频中的新位置时
    - stalled 当浏览器尝试获取媒体数据，但数据不可用时
    - suspend 当浏览器刻意不获取媒体数据时
    - timeupdate 当目前的播放位置已更改时
    - volumechange 当音量已更改时
    - waiting 当视频由于需要缓冲下一帧而停止
- 音频可视化
- 自定义播放器

第23章 - [扩展]地理信息获取和百度地图API

- navigator.geolocation
- 单次定位请求： getCurrentPosition(请求成功，请求失败，数据收集方式)
    - 请求成功函数
        - 经度: coords.longitude
        - 纬度: coords.latitude
        - 准确度: coords.accuracy
        - 海拔: coords.altitude
        - 海拔准确度: coords.altitudeAcuracy
        - 行进方向: coords.heading
        - 地面速度: coords.speed
        - 时间戳 : new Date(position.timestamp)
    - 请求失败函数
        - 失败编号： code
            - 0: 不包括其他错误编号中的错误
            - 1: 用户拒绝浏览器获取位置信息
            - 2: 尝试获取用户信息，但失败了
            - 3: 设置了timeout值，获取位置超时了
    - 数据收集:
        - enableHighAcuracy: 更精确的查找，默认false
        - timeout: 获取位置允许最长时间，默认infinity
        - maximumAge: 位置可以缓存的最大时间，默认0
    - 多次定位请求 : watchPosition(像setInterval)
        - 移动设备有用，位置改变才会触发
        - 配置参数：frequency 更新的频率
    - 关闭更新请求 : clearWatch(像clearInterval)
- 百度地图API 基本使用
    - 密钥申请
    - 百度坐标转换
    - 创建地图

    - 添加标注
    - 标记信息
    - 本地存储信息持久化
- 案例：行走日记

模块七 小程序开发

第24章 小程序开发

- 小程序简介
- 小程序环境
    - 目录结构
        - app.json
            - 全局的公共配置:是当前小程序的全局配置，包括了小程序的所有页面路径、界面表现、网络超时时间、底部 tab 等
        - app.wxss
            - 小程序公共样式表
        - app.js
            - 小程序逻辑
        - project.config.json
            - 记录开发者工具配置信息例如界面颜色、编译配置等等
        - sitemap.json
            - 配置小程序页面是否被微信索引
- 小程序页面
    - js：页面逻辑
    - wxml：页面结构
    - json：页面配置
    - wxss： 页面样式
- 事件系统
- 模板引用
    - 定义模板
    - 引入模板
- 网络请求
- 案例
    - 新闻列表实现
- 自定义组件
    - 创建自定义组件
    - 使用自定义组件
    - 组件传参
- 小程序中websocket请求
    - 创建一个 WebSocket 连接
    - 监听 WebSocket事件
    - 发送消息
- 案例
    - 《3秒谁准》多人小游戏
- 云开发

    - 云数据库
        - 增加
        - 修改
        - 删除
        - 查询
    - 云函数的使用
- 小程序用户授权
- mpvue构建项目及使用
- 定位功能
- mpvue中的云开发

第25章 小程序实战 - 美食点评系统

- 小程序实战 - 美食点评系统

模块八 面试题起底

第26章 面试题起底

- Promise 源码解析
    - Promise 类
    - Promise 状态
        - PENDING
        - FULFILLED/RESOLVED
    - Promise.then 实现
- 宏任务与微任务
    - then 的返回值
    - promise.catch 方法
    - promise.finally 方法
    - Promise.resolve 方法
    - Promise.reject 方法
    - Promise.all 方法
    - Promise.race 方法
    - Promise.allSettled 方法
- React 核心原理解析
    - Virtual DOM 解析
    - Diff 算法解析
- 面试分析
    - 简历编写
    - 面试前准备