# 前端

## 文件断点续传

### 技术点

前端通过FileList对象获取到相应的文件，按照指定的分割方式（Blob.prototype.slice 返回源文件的某个切片）将大文件分段，然后(借助http可并发性)并发将文件传给后端，后端再按顺序一段段将文件进行拼接。

前端：

Promise.all() 并发切片

后端：

何时合并：前端携带切片最大数量的信息，当后端接收到最大数量信息时合并

## JS

### Symbol 2021年3月30日

作用:保持属性唯一性

;let obj = {}

;let a = Symbol("a")

;obj[a] = 123

### Event

#### event.target 和 event.currentTarget

curentTarget 表示当前绑定的元素，target表示触发事件的元素

#### 阻止冒泡或者捕获

event.stopPropagation() 可以阻止事件的冒泡或者捕获

event.stopImmediatePropagation() [Event](https://developer.mozilla.org/zh-CN/docs/Web/API/Event) 接口的 stopImmediatePropagation() 方法阻止监听同一事件的其他事件监听器被调用。

### XMLHttpRequest

#### Content-Type (Mime-Type) 编码类型

Content-Type(内容类型)，一般是指网页中存在的Content-Type,定义网络文件的类型和网页的编码，决定浏览器将以什么形式，什么编码读取这个文件，这就是经常看到一些PHP网页点击的结果却是下载一个文件或是一张图片

### PerformanceNavigation

接口呈现了如何导航到当前文档的信息,这个类型的对象可以被只读属性Performance.navigation调用

#### Performance.navigation.type

TYPE\_RELOAD(1)代表是通过刷新按钮到达该页面

### Document.referrer

返回一个url，当前页面就是从这个URI所代表的页面跳转或打开的

## Vue

### vue.config.js

#### pluginOptions (vant全局变量)

[https://cli.vuejs.org/zh/config/#pluginoptions](https://cli.vuejs.org/zh/config/" \l "pluginoptions)

这是一个不进行任何 schema 验证的对象，因此它可以用来传递任何第三方插件选项

vant全局变量

  pluginOptions: {

    "style-resources-loader": {

      preProcessor: "less",

      patterns: [

        path.resolve(\_\_dirname, "./src/assets/css/global.less")

      ]

    }

  }

### 第三方插件

#### [Vue I18n](https://kazupon.github.io/vue-i18n/zh/) (中英文切换) 2021年3月30日

// 通过选项创建 VueI18n 实例

const i18n = new VueI18n({

locale: 'ja', // 设置地区

messages, // 设置地区信息

})

该messages通过require或import引入（js 文件种通过module.export, export default），npm run build浏览器会在使用时加载js，此时浏览器会报错，不支持export 或者module

### 绑定原理

defineReactive getter时候把依赖注入，setter的时候通知watcher更新

### vuex原理

$store通过mixin混入进组件，在beforeCreate钩子时期加入$data里

### v-for Map

v-for = "([key,value],index)" in map

## Git

## SSH

SSH为建立在应用层基础上的安全协议。

SSH是目前较可靠，专为远程登录会话和其他网络服务提供安全性的协议。

### id\_rsa.pub

公钥。

一般在C:\Users\pc\.ssh。

## Axios

### CancelToken

<https://www.cnblogs.com/qlongbg/p/12966813.html>

CancelToken常用在封装的请求中，用来取消上一一面axios请求

在路由跳转时，若当前页面的数据量过大，而我们立即点击跳转下一页面，那么可能会出现，当前页面接口还在pending状态，页面已经跳到新页面，旧的请求依旧没有停止。

这将会十分损耗性能，这时我们应该先取消掉之前还没有获得相应的请求，再跳转页面。这就是CancelToken的作用

1在main.js里写一个全局httpRequestList的空数组，用来装我们的cancel函数

2在封装的get和post请求里面，将cancel函数推入httpRequestList数组

// 清空cancelToken中的cancel函数

function clearHttpRequestingList() {

  if (Vue.$httpRequestList.length > 0) {

    Vue.$httpRequestList.forEach((item) => {

      item()

    })

    Vue.$httpRequestList = []

  }

}

3在路由守卫中，写一个执行cancel方法的clearHttpRequestingList方法，在每次跳转之前执行clearHttpRequestingList()函数

## Moment

### 某月第一天(startOf)

moment(month).startOf('month').format("YYYY-MM-DD")

### 某月最后一天(endOf)

moment(month).endOf('month').format("YYYY-MM-DD")

### 某月减几个月(subtract)

moment(month).subtract(length, 'month').format("YYYY-MM")

### 两个日期相差的月数(diff)

moment(endDate).diff(moment(startDate),"month")

### 某日期在某日之前(isBefore)

moment(vm.endMonth).isBefore(vm.startMonth)

## Lodash

### 防抖\_.debounce(func, [wait=0], [options={}])

当调用动作触发一段时间后，才会执行该动作，若在这段时间间隔内又调用此动作则将重新计算时间间隔。

### 节流\_.throttle(func, [wait=0], [options={}])

预先设定一个执行周期，当调用动作的时刻大于等于执行周期则执行该动作，然后进入下一个新的时间周期。

### 忽略某个属性 \_.omit(obj,property1,property2...)

截取对象时忽略这些个属性

# 后端

## Excel生成器

import org.apache.poi.ss.usermodel.Workbook; （接口）

import org.apache.poi.xssf.usermodel.XSSFWorkbook;（实现类）

## 避免 Spring 的 AOP 的自调用问题。

(https://www.cnblogs.com/xd502djj/p/10940627.html)

在Spring的AOP代理下，只有目标方法由外部调用，目标方法才由（Spring 生成的代理对象）来管理，这会造成自调用问题。

在Spring的AOP代理下spring的代理对象才能发生回滚

若同一类中的其他没有@Transactional 注解的方法内部调用有@Transactional 注解的方法，有@Transactional 注解的方法的事务被忽略，不会发生回滚。见清单 5 举例代码展示。

清单 5.自调用问题举例

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | @Service  public class OrderService {      private void insert() {  insertOrder();  }  @Transactional      public void insertOrder() {          //insert log info          //insertOrder          //updateAccount         }  } |

insertOrder 尽管有@Transactional 注解，但它被内部方法 insert 调用，事务被忽略，出现异常事务不会发生回滚。

上面的两个问题@Transactional 注解只应用到 public 方法和自调用问题，是由于使用 Spring AOP 代理造成的。为解决这两个问题，使用 AspectJ 取代 Spring AOP 代理。