Jigsaw Map: a simple and customizable web library for interactive maps

Zhibang Wang 517030910298

I. Abstract

为了在网页上展示学术关系图等学术地图, 浏览器需要绘制许多的节点和线段的 dom 元素. 大量的 dom 元素会导致浏览器的卡顿, 影响用户正常的浏览体验. 为了改善这一现象, 之前的一些工作试图引用将网页地图中的分层思想引用进来, 意在减少浏览器无效的加载, 按需加载需要的视觉要素. 在我的项目中, 我搭建了一个类库 Jigsaw Map, 抽象出了一些分层地图的逻辑, 使使用者能在不同的场景自定义地使用分层加载.

Key words: web, map, ts.

II. Introduction & Related work

学术地图可以很直观的观看识别学术的聚集, 迁移信息. 通过 web 的形式展示学术地图无疑是最方便的. 交大的 acemap 团队就在其官网 https://www.acemap.info/上展示多张不同种类的学术地图.svg 是一种矢量图技术, 他能保证画面不会随着放大而失真,acemap 目前在网页上展示的学术地图使用的正是这种 web 技术. 然而在页面上绘制大量的 svg 元素会导致页面的明显卡顿.acemap 上的学术地图大概有超过 5000 个节点和文字以及超过 15000 条线段. 在我们的简单测试下, 直接缩放这张地图会导致浏览器非常卡顿甚至短暂卡死, 在删除掉所有的边之后, 缩放依然有明显的停顿感, 而在只保留所有节点的情况下缩放会比较顺畅. 然而, 用户不可能不去关注文字和线段, 这是这张学术地图的关键组成部分.

分层加载技术是一种广泛应用在网页地图上的技术, 由于地图本身过于庞大, 一次全部加载是不可能的, 也是没有必要的, 只加载用户关注的地图部分并在适当的时机更新地图可以大幅提高用户体验并且减少不必要的资源消耗. 目前主流的地图都使用了这种技术, 但实现方式各有不同. 搜狗地图和谷歌地图基于切片图片, 并在页面上添加了一下其他的交互元素. 百度地图和高德地图基于 canvas, 它的内部实现更加复杂, 为更复杂的地图系统提供了保障. 而不管是哪种实现方式, 其大致的核心思想是相同的.

leaflet 是一个开源的交互式 web 地图的 library. 其实现了分层地图的基本框架, 并对图片,svg 等地图形式都提供了相应的接口, 提供了一系列交互和动画支持, 使用户可以轻松的使用它来创建自己的地图组件. 然而,leaflet 也有一定的不足. 其高度的封装度导致了自定义变得困难, 不灵活. 例如,leaflet 每一层级的图片必须包含下一层级的 4(2x2) 张图片 (当然这个数字可以满足大量的需求), 其对图片的尺寸也有一定的规定. 我们也很难对加载这一行为本身进行一些 DIY.

虽然 leaflet 能完成一定的常规需求, 但对于一些特殊需求就无法很好的满足. 因此, 在这次大作业中, 我把重心放在了分层加载的设计上, 希望制作出一个更加简单, 高可定制性的可交互的分层地图库, 在能更好的展示目前的学术地图的同时, 也能具备一定的复用性, 应用到其他的一些场景中去.

这次的大作业使用 typescript 来进行开发, 他是 JavaScript 的超集, 提供了类型检查的功能, 使工具库的开发能够避免一些意想不到的类型错误, 保证程序的健壮性.

III. Implement

A. 数据设计

分层加载地图的数据结构其实就是一个矩阵. 可以轻松地通过二维数组实现, 而与普通矩阵不同的是, 他在 x 和 y 方向都有一个头部指针, 他的是确定显示地图区域的坐标, 以及更新地图的一个标志. 需要注意的是, 这个矩阵的总大小需要超过我们的可视区域. 当我们拖动地图的时候, 程序判断是否拖动到一定的阈值, 如果达到阈值, 我们就将指针指向的切片移动到列表的最末端, 同时更新这个切片显示的内容, 并移动头部指针指向下一个切片. 需要注意的是, 移动的切片其实在矩阵中的索引并没有发生变化, 只是改变了其在浏览器中的位置坐标. 移动的过程可以如图 1所示展示了在 x 轴移动时其中一行切片的变化情况, 我所完成的这个地图库的高可自定义性就体现在了这个 *move and update* 的过程中, 这个过程中会提供给切片新的 xy 编号以及地图的深度和一些其他信息, 根据这些信息, 我们就可以控制地图更新的内容了.

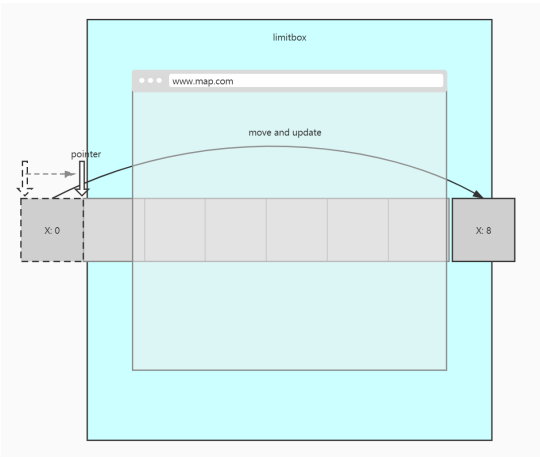

Fig. 1. structure

当然如果快速移动地图的坐标, 这种阈值判断的方法每次只能移动一个切片, 势必是会很低效的(比如要移动矩阵一个方向长度的好几倍). 每个切片要被更新许多次, 然而实际上我们可以通过计算一次移动到位. 我在实现这个功能的时候也考虑了这一点. 程序会判断地图移动坐标的多少, 如果超过一定量, 会用更高效的批量更新方法更新移动地图.

B. 架构设计

我认为, 一个分层地图应具备缩放, 拖拽的功能, 然而这些功能的实现其实与地图本身不应具有相关的关系. 一个地图的状态其实只与 x,y 坐标, 放大倍数这三个量有关, 其余的均为附加属性. 而我们的拖拽以及缩放, 其实只是在原有的 x,y 或者放大倍数上再增加一个变化量. 在我的设计中, 使用拦截器的设计思想, 外部可以直接修改 x,y,scale, 拦截器拦截修改属性, 并改变地图的状态. 实际上, 地图内部也维护了一组相关坐标量, 他与外部调用的量有一定的转换关系, 整体架构如图 2所示

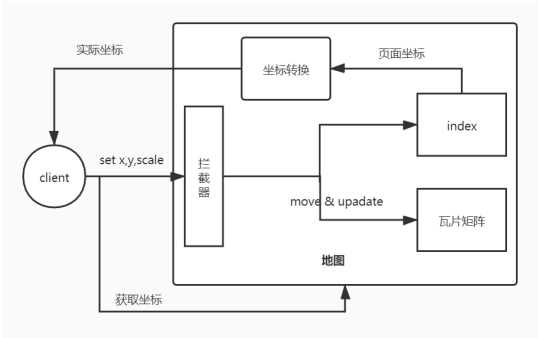

Fig. 2. frame

由于我们保存了每个切片的 xy 序号, 同时两个头部指针指向了左上角的第一个切片, 我们能通关换算轻易地算出页面上每一个位置所对应的实际地图坐标 (比如经纬度). 因此, 在缩放时, 我们可以轻松地获取鼠标位置的经纬坐标, 推算出新的图层对应的左上角切片的 xy 序号以及位置偏移量, 从而更新图层达到缩放切换图层的目的. 实际上, 后续我们为地图添加放大镜以及鹰眼的功能, 也很大程度地依赖于这一套指针带来的坐标转换系统.

我们在地图初始化的时候, 自动地为它添加事件处理系统, 事件处理系统与上文提到的拦截器和坐标转换系统对接, 从而达到了交互地图的功能.

C. Extra Feather

除了基础的缩放和拖动功能外, 我也为地图添加了其他的两个功能来提高可交互性: 右键放大镜, 鹰眼小地图.

放大镜的实现其实是与缩放更新图层非常类似的, 通过坐标转换系统得到鼠标当前位置的经纬坐标, 通过经纬坐标得到小地图左上角的切片的 x-y 标号以及整体的地图偏移量, 从而生成放大镜.

鹰眼功能的实现相对比较复杂. 鹰眼的内部维护了一套新的坐标转换系统. 在原有的地图上添加监视器, 与鹰眼进行绑定. 当原地图移动时, 触发监听函数, 传递的参数是通用的经纬坐标, 从而达到鹰眼地图的指示器会与地图同时移动, 而当我们移动鹰眼的指示器的时候, 他会换算一组浏览器坐标系中的坐标, 调用地图的拦截器模拟拖拽地图.

在添加完成这两个功能之后, 我们的整体架构如图 3所示:

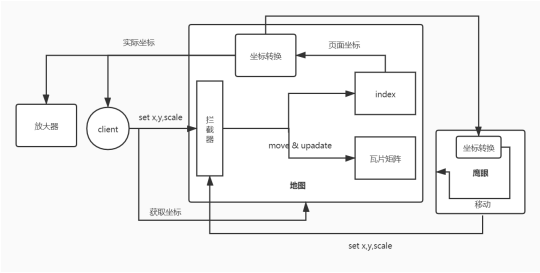

Fig. 3. frame complete

D. Backend

在完成了上述功能之后, 我们的交互地图库基本上已经完成了. 当然, 我们需要把他利用在学术地图上. 由于我的大部分项目时间都投入了之前的设计中, 因此没有太多的经历去处理后端. 所以我们的分层分片策略相对来说比较简单. 首先从 acemap 上下载了一些学术地图的 svg 文件. 用 python 写了一个简单 svg 解析, 判断 path 两端之间的距离, 将超过一定距离的 path 删除掉, 这样, 我们就得到了拥有不同边数的多个地图层. 从而使我们放大观察的时候不会受到太多无关边的干扰. 将这些处理后的 svg 转换成不同大小的图像, 并切割成统一大小的图片, 我们的后端图片就算完成了, 将其直接放置在静态文件夹就可以使用了.

IV. Display

我的成果展示网页目前部署在 gitpage 上:https://karshbang.github.io/jigsaw-map/, 由于使用的是 gitpage, 因此静态资源加载得会比较慢, 如果有专门的服务器加载速度应该会比较好. 除了学术地图, 为了展示这个类库的高自定义的性质, 在网站上还展示了前端模拟切图 (将一张大图片在前端模拟切成许多小图片模拟分层加载) 以及一个骨架地图 (显示切片的 xy 序号以及目前的层数)

V. Future work

目前我的 library 已经可以完成一个分层加载器的所有基础功能了, 但是还有许多可以做的改进由于人力时间有限暂时无法完成. 目前虽然支持自定义事件, 但是事件是完全自定义的, 而像图片这样的 dom 元素如果我们想要做到点击某区域触发事件用原生的方法是很难做到的, 因此我想在未来提供类似于 zRender 中的事件处理机制, 优化事件体验.

其次, 由于时间有限, 目前只对 pc 端做了适配, 移动端的体验不佳, 也是未来的改进方向.

目前的后端切片算法也比较简单, 图片上还是会有许多的冗余信息, 可能需要提供更好的切片算法优化图像显示的内容, 并且能提供一些非静态资源的 api 来支持更好的前端展示效果.