代码结构文档

本文档将会展示topic_river.js中的函数调用的内部基本逻辑结构。文档最后的结构图或许能帮助你更加直观的认识代码结构。

首先，topic_river.js中共有4个接口，分别为create, add_word_cloud, addFuncOnClickEvent, addToken.

create接口能够帮助用户建立起静态的河流图，它通过调用create_topic_river和create_detail_graph，分别创建总体的河流图和细节图。
在create_topic_river中，依次顺序调用create_cut_off, create_archive_path, create_river_path, create_stream_tail, create_stream, create_comment_area, create_stream_text，来完成绘制和初始化。
而在create_cut_off, create_archive_path, create_river_path, create_stream_tail, create_stream中通过调用核心函数来实现绘制过程。

addToken接口能够帮助用户加载流数据，并且以小球动画的形式展示在河流图上，并且在一定阶段的累积后，实现河流图的整体流动。当流数据累计到达上限后，addToken将会掉用move_river来实现河流的整体流动，否则通过cal_stream_path后进行create_path或者change_path来实现stream graph的更新。
在move_river中也会通过调用核心函数来实现绘制过程。

add_word_cloud用于生成字云。

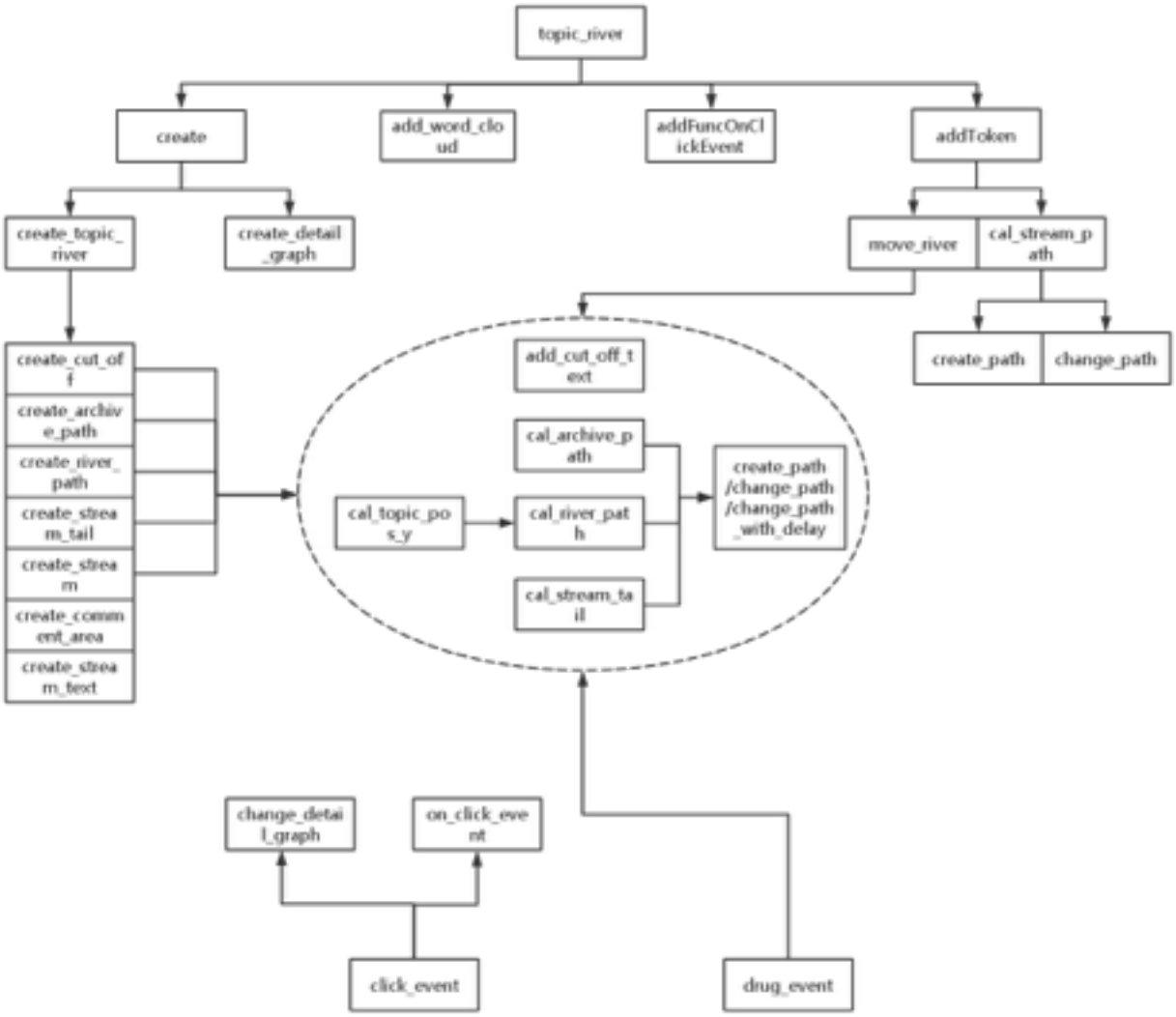
addFuncOnClickEvent用于添加点击事件的调用函数。

另外topic_river.js除了生成河流图以及添加数据流动效果，还提供了一定的交互，包括鼠标经过，点击以及拖动，在这里我们主要展示鼠标点击和拖动的实现。

当click_event发生的时候，detail_graph（折线图）将会显示出来，通过调用change_detail_graph实现。同时将会调用on_click_event，这个函数就是通过addFuncOnClickEvent添加的函数。

当用户拖动分割线的时候，用户能够调整河流图各个区域的大小，这个功能也是通过调用核心函数实现。

核心函数包括分割线的标签的更新，archive区域的绘制，河流图区域的位置，stream区域尾巴的绘制。而archive区域的绘制，河流图区域的位置，stream区域尾巴的绘制都可以重用create_path。


topic_river
create
add_word_cloud
addFuncOnClickEvent
addToken
create_topic_river
create_detail_graph
move_river
cal_stream_path
create_path
change_path
create_cut_off
create_archive_path
create_river_path
create_stream_tail
create_stream
create_comment_area
create_stream_text
add_cut_off_text
cal_archive_path
cal_topic_pos_y
cal_river_path
cal_stream_tail
create_path/change_path/change_path_with_delay
change_detail_graph
on_click_event
click_event
drug_event