小程序运行时开发文档

1952890顾文涛 数据科学与大数据技术

**一 、开发需求分析**

开发任务的目标是完成一个小程序进行时，一个可以在浏览器（chrome）端运行小程序的小程序框架。

按照项目要求，小程序框架需要实现双线程，即利用webworker实现大多数小程序所走的逻辑视图层分离。

除了双线程外，小程序运行时还需要做到一个基于vdom的渲染框架。按照要求，即利用babel，首先通过 jsx 来描述页面，然后编译成 render function，执行后产生 vdom。

最后，是对DSL渲染的要求，主要 包含条件、循环以及引用。

**二、实现成果简介**

通过四周的时间，完成的成果如下图所示：

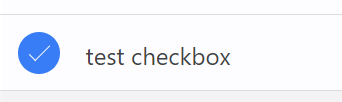


随着时间推移，右边泡泡中的数字会变大，代表有更新的消息：





中间有checkbox按钮用来测试jsx是否能够正常编译渲染：



**三、运行步骤**

小程序以及小程序运行时主要有两个主要步骤：

A npm install

目的：安装写于package-lock中的各种依赖。这里面直接使用到的依赖主要包含superstatic、babel、jsx、worker-load等。

B npm start

目的：启动项目。在package.json文件中写了start脚本的具体操作，如下图所示：



Npm start 主要做了两件事，一个是build操作，另一个是superstatic的server热启动操作。把安装和启动流程走完后得到的结果如下（node11.15.0）



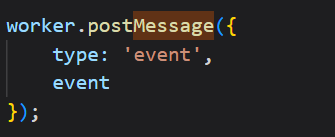
在localhost：8080里就可以得到小程序运行效果。

**四、代码开发分析**

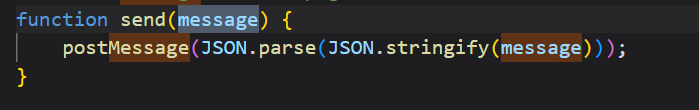
**4.1 webworker双线程通信**

按照文档的指导，应当把渲染层与逻辑层分离，因此在项目中，包含dom.js以及worker.js。两者之间的通信直接通过postmessage完成，如下图所示：

Dom.js

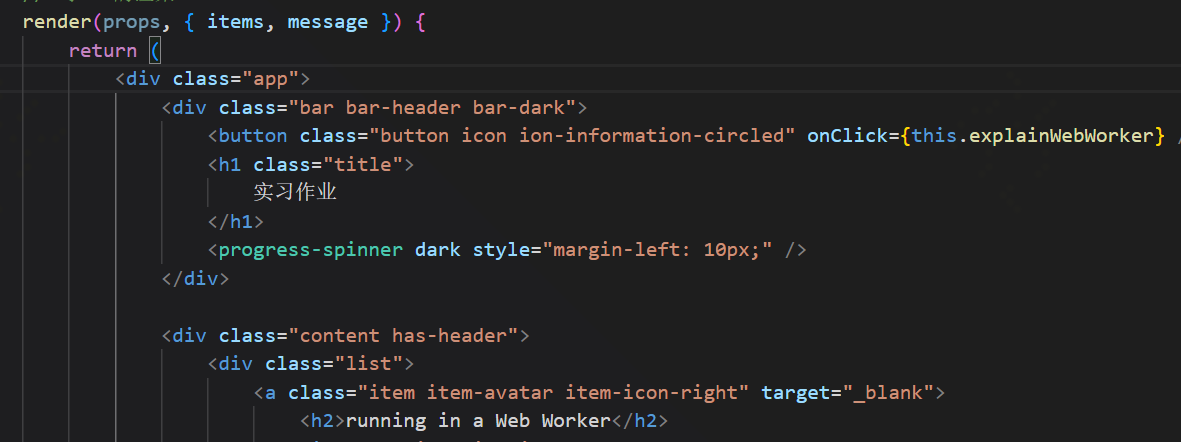


Worker.js

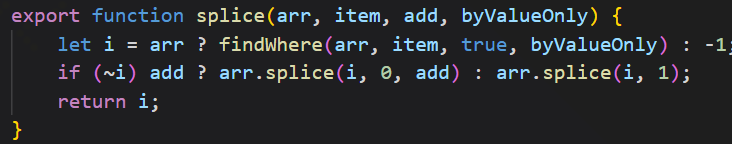


**4.2 jsx生成fuction与vdom**

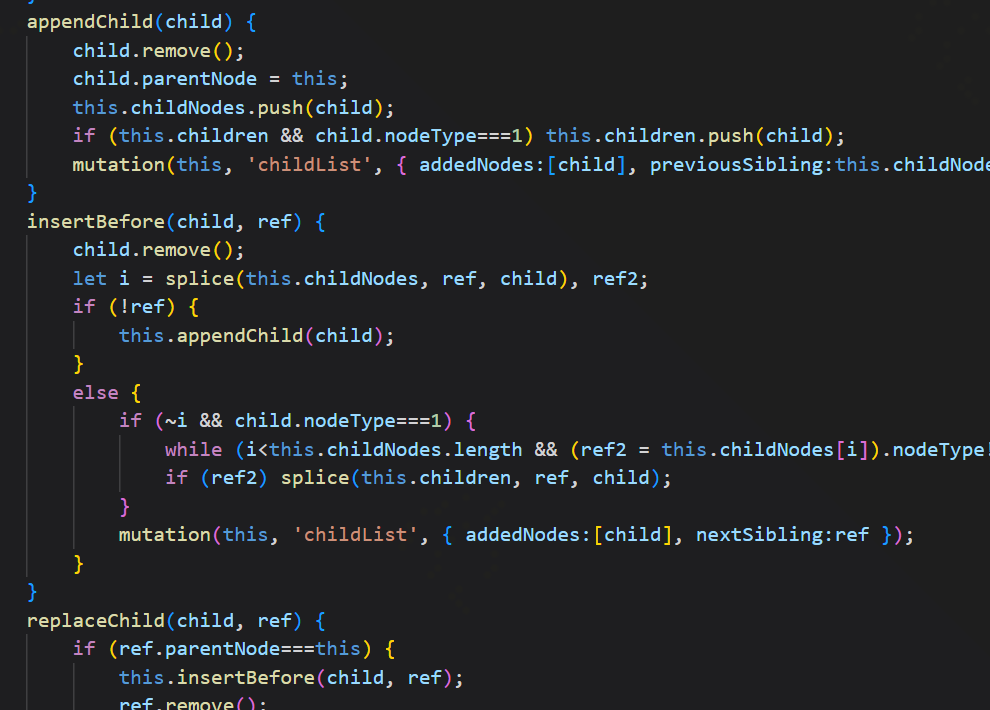
在app.js中，包含了jsx如下：





通过render可以把jsx转化为render function。这其中涉及到的核心递归，在lib/util.js之中：

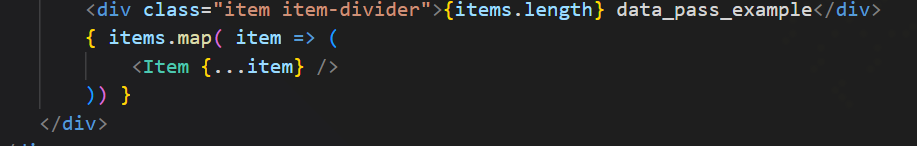
除了核心递归以外，还包括了其他函数：



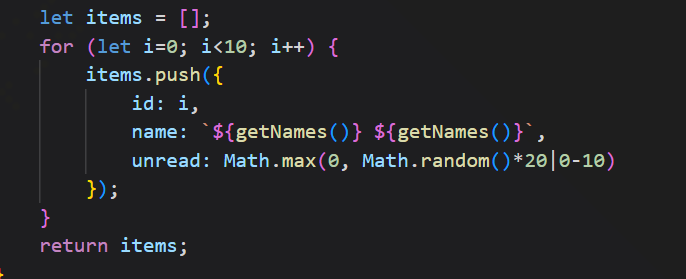
**4.3 DSL**

由于时间的限制，在DSL上并没有实现if与for的编译，但是做到了数据传递，在app.js中包含数据传递的方式如下：





其中render是对jsx的render，而items在示例app中如下：



**五、心得与总结**

通过为期四周的字节培训，我了解到了更多关于前端小程序的运行底层逻辑，特别是编译渲染的知识。这些知识都是学校学不到的，因此感激字节提供的培训机会，希望以后能够在前端开发的方向上学到更多，产出更多。