一套莫尔斯电报听写、翻译系统

单位来新人，惯例要先学三个月的莫尔斯报，由于训练软件是是十年前用MFC开发的，无法修改功能，且不兼容win7，经常有错误闪退，所以我弄一个简易的练习系统。从设计到完成一共用了七天左右时间，具备随机生成、考核、听写、翻译和中文译解功能，并可以控制报文类型，数量，速度，可以设置背景音乐和声音强度。



图一：主界面



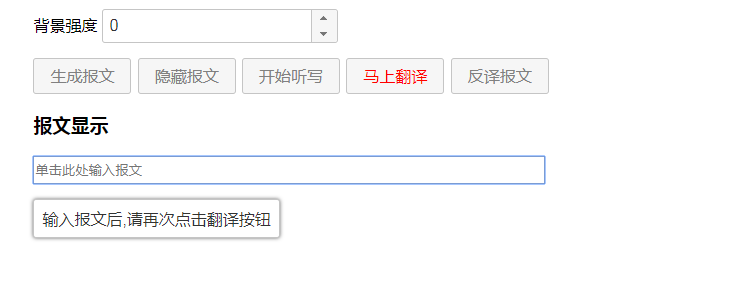
图二：支持五种报文



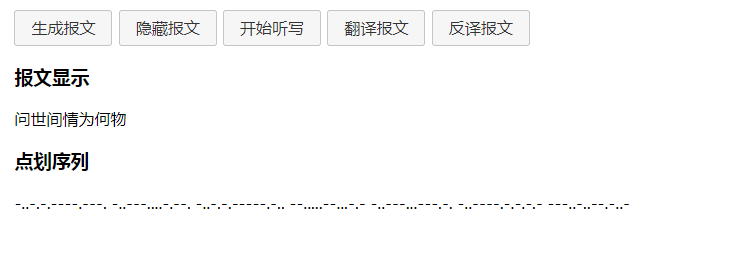
图三：随机生成20组字码



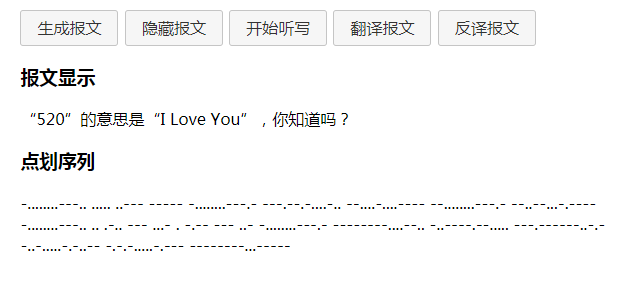
图四：点击听写，会按照设定速度发报，并可以设置背景杂音和杂音强度



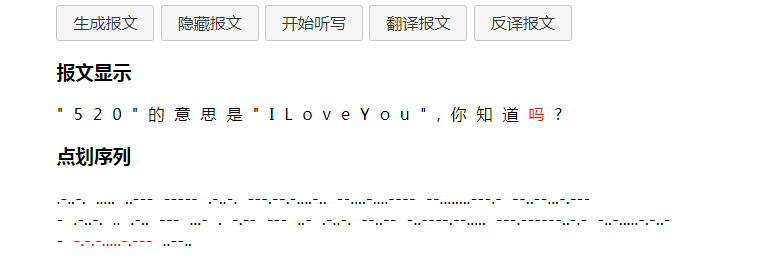
图五：翻译功能，目前支持中文，大写英文，小写英文以及各种符号



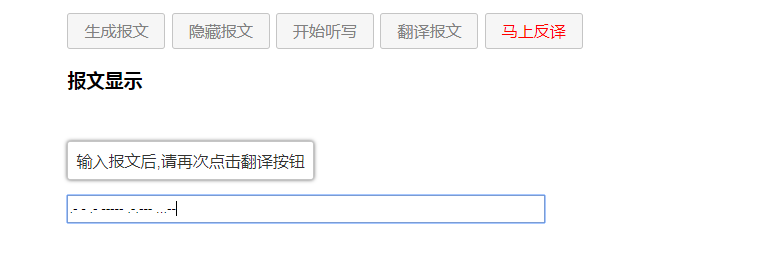
图六：点击红色按钮，进行翻译



图七：翻译功能支持不同语言，大小写和各种符号同时输入



图八：翻译出的结果支持听写，发报时相应的元素会标红，就不会看串行了~



图九：反译功能能够把莫斯码反译成可理解的报文

API

均位于morse.js当中

Getmorse：（line 9）用于生成各类型莫尔斯码，目前支持长码、短码、字码、混合码、勤务用语五种，可根据需要自行扩充

InitBtn：（line 70）用于为页面中所有按钮赋予功能

goTranslate：（line 221）翻译模块，首先判断是否有中文，无中文,使用混合码字典，有中文，则调用xmorse模块

artiTranslate：（line 259）反译模块，判断是否只有 . 和 – 的莫尔斯码 ，是则反译成报文，不是则翻译成莫尔斯码

playBg：（line 279）播放选定的背景音乐

closeBg：（line 289）终止选定的背景音乐

async play：（line 296）用于处理系统产生的 点、划、字、词等事件，决定如何播放和等待

playAudio：（line 322）区分点、划进行播放调度

sleep：（line 342）系统休眠，用于生成听写时的时间间隙

createTranslate：（line 349）将翻译出的报文转译成系统可播放、识别的格式。思路是字符串转数组，数组元素包裹span插入dom，以实现跟随听写字符标红，摩斯码打碎成点划并加入byte、word等控制信息

createMessage：（line 419）随机生成若干组报文，实现思路同上

initWidget：（line 480）初始化widget，注意不要在widget内再封装widget，没有内部通信方法

playDot：（line532） 为正弦波添加矩形滤波器后调制，在440hz上0.1秒内声音近似为点

playLine：（line 567）同上，只是将包络时长设置为dot的三倍

总结：

一、关于gulp，由于之前一直在学webpack，所以经常看到webpack和gulp相比较的文章，正好这次有这个需求，就尝试了一下gulp，使用体验就是很简单，用了二小时看了几篇文章就很熟练了，当然缺点也是简单，一些事情做不到，还有一些事情要用额外插件来做，配置起来很麻烦。

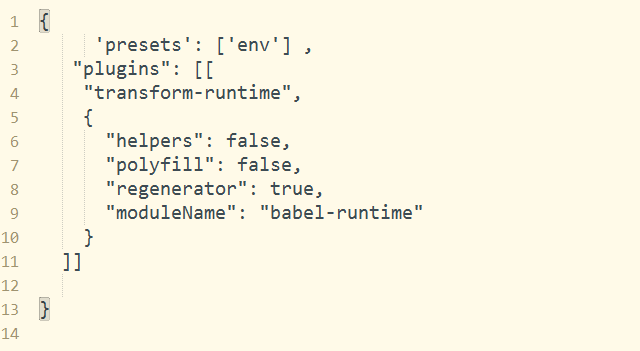
比如这个程序里绝大部分使用es5语法，有一些[…]、promise、require之类的es6语法，还有await、async这种es7语法,而浏览器只认识es5,这就导致必须用babel转码，而babel在gulp需要手工配置，这是一个天坑。

首先babel-core 和 gulp-babel版本不能用最新的，会报错误找不到@babelcore，至少要退化两个以上的版本



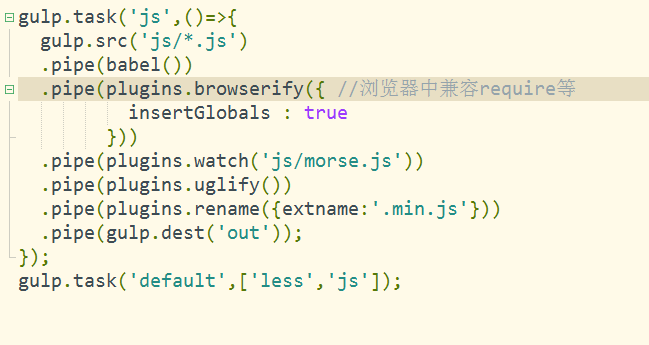
图十：我使用的babel-core 和 gulp-babel版本

其次要支持es7，必须添加babel-plugin-transform-runtime，同时在.babelrc文件中添加对应的plugins



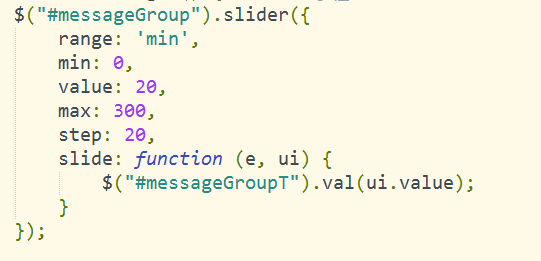
图十一：支持es7 asyns的配置

你以为这就完了？transform-runtime会把regenerator转换成require，只是把es7转成了es6，可以在node上运行，但是想要用script直接引用还要添加gulp-browserify转成es5，注意顺序，要在babel()之后,在uglity()之前

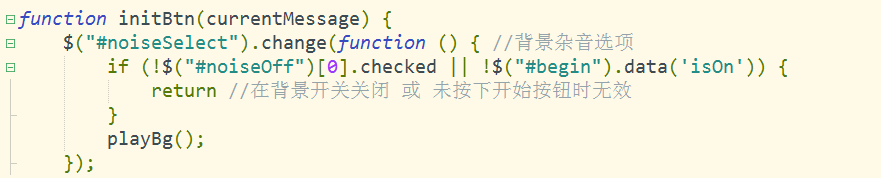


图十二: gulp-browserify的配置

二 关于jQueryUi框架，页面使用了jQueryUi，也没多想，就是想试试没用过的东西，尝尝鲜，在菜鸟教程看了几篇文章大概半天上手，在思路上它是对dom元素进行widget初始化的方式来表现ui，但是这些widget屏蔽了外部通信事件，不容易找到解决方案，比如这个软件中，一开始我把背景杂音和背景强度封装在一个controlGroup当中，结果select的onchange事件居然失效了！震惊，后来不得不用css实现了controlGroup的样式。这个地方如果不熟悉的话坑会很多，所以我感觉没有bootstrap+jquery效率高，jQueryUi对我来说应该是浅尝辄止，以后不会再用了。



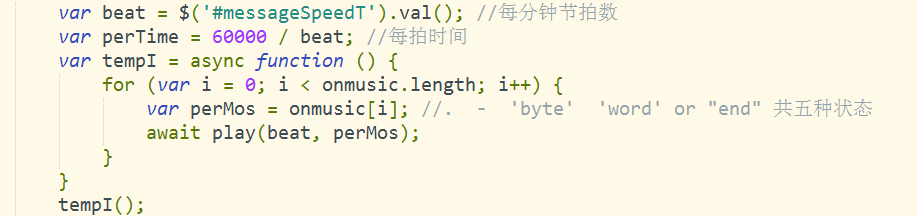
图十三：jQueryUi的初始化十分简单易懂



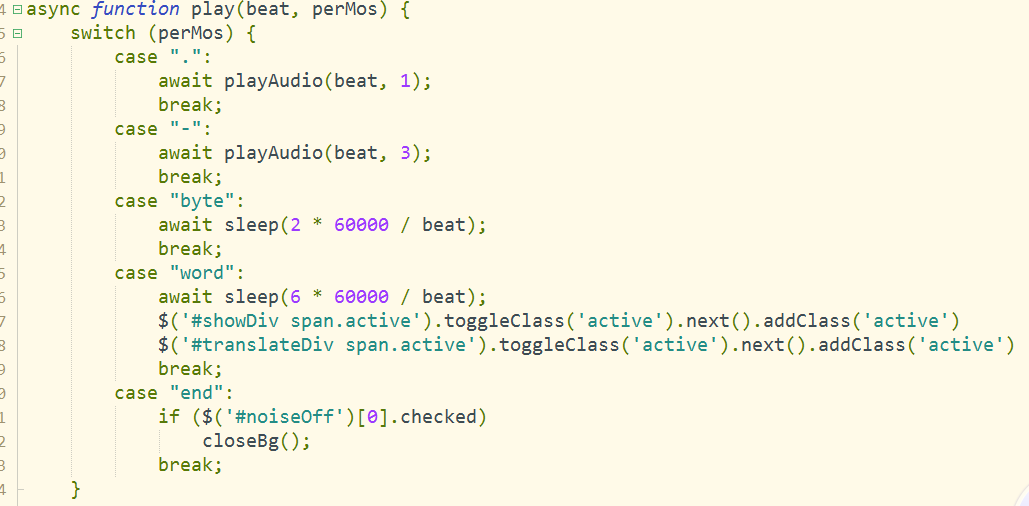
图十四：select的change事件

三 关于异步低速函数，以前在node里就是无限的回调函数嵌套,在es7中有了await和async这两个方法。

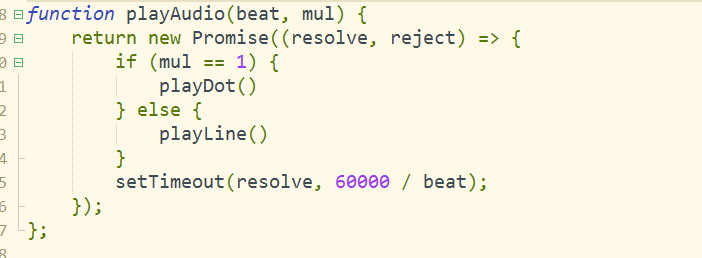
await和async就类似于回调嵌套，但声明他们的时候不会执行，如图十五和图十六中，tempI和play全是async函数，他们把任务不断细分，函数执行是在图十七和图十八中的playAudio和sleep函数，他们两个都是play中的分支任务，当他们返回new Promise时会执行自身。在内存中定义自己，而当setTimeOut运行之后返回的resolve会结束await状态，父函数play会继续执行，从而可以实现使用setTimeOut节拍来控制发报速度



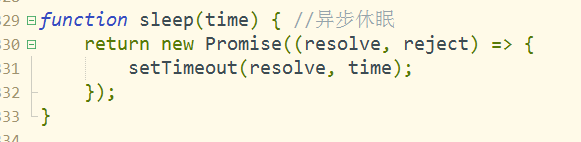
图十五：tempI



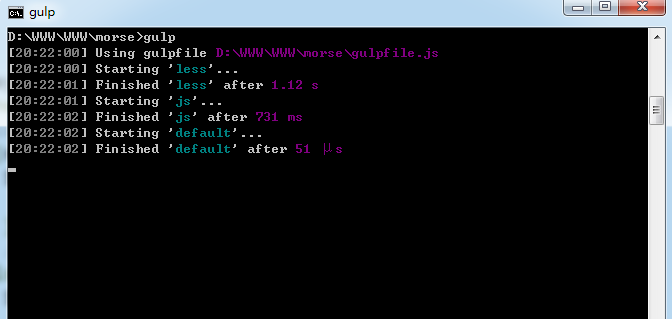
图十六：tempI 的子函数play



图十七：play的子函数playAudio



图十八：play的子函数sleep

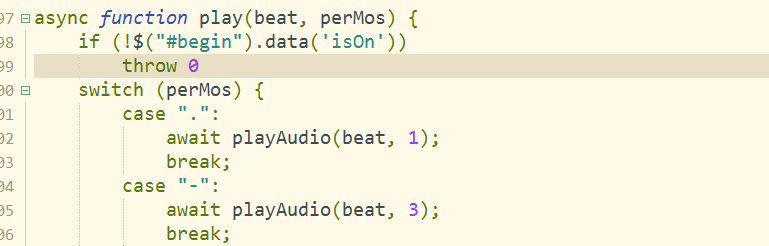


图十九：gulp正确运行

最后，关于使用开关取消promise链，控制播放与停止的思路：为开关添加isOn属性，点击时能控制isOn在true和false之间切换，然后在async play中监听此属性，若关闭，则抛出错误，如图二十一，然后在其父函数中catch这个错误，打印错误，如图二十。该方法的核心就是在子函数中监听全局变量，然后抛出错误给父函数进行处理。



图二十：使用try-catch运行异步函数



图二十一：子函数play中监听点击事件，抛出错误