TOP 实践报告

13331078 胡江川

主要工作：

1. 学习 livescript 语法结构
2. 配置 grunt 环境
3. 学习 TOP 思想，整合思路，重写 S1~S5

前次问题：

1. 对于上一次的作业，我的思路是从简至难，先完成简单的点击事件，再到复杂的请求响应和逻辑组织，最后针对可能出现的 bug 进行细微修正。
2. 按钮没有使用对象处理，需要平凡地进行取对象操作和传参。在鼠标移除区域后，采用了糟糕的 reset 方式，对每个按钮都重新绑定了 click 事件。由于没有采用对象的概念，reset 起来过程冗杂。
3. Robot 采用了递归思想，通过请求回调时调用来点击下一个按钮。为了保证 S1 功能的完整性，robot 在点击前会判断当前按钮是否可按，否则递归自身。
4. S5 的 handler 冗长，复用程度低，语法结构不清晰。

TOP 尝试：

1. **由大体到细微**。从最简单的 S1 做起，首先添加所有的功能，功能的执行先不考虑

```
$ ->
  add-clicking-to-fetch-numbers-to-all-buttons!
  add-clicking-to-calculate-result-to-the-bubble!
  add-reset-when-mouse-leave!
```

   考虑到多个 button 使用继承对象会更便于管理，于是编写了 Button 类，bubble 由于需要多种操作所以同样编写 Bubble 类。然后开始逻辑的处理。首先 button 有两种功能：fetch-and-show/reset。其次 4 种状态：enabled/disabled/wait/done。外加判断 bubble 和开关其他按钮的函数。Bubble 有 3 种功能：开关/计算/reset。S1 就完成了。检查无 bug 通过。
2. **想到什么写什么，但同时保持思维清晰**。像 S1，一般写着写着会突然发现需要一些辅助函数来完善，然后就添加进去。S2~S5 中需要添加机器人，那么先添加机器人 robot。然后就会想到机器人应该是在@+点击时启动，于是添加@+点击事件。接着想到 robot 会在回调时点击下一个按钮，给机器人添加 get-next-button 功能，完善 init 函数。最后根据 S2~S5 不同机器人的要求改写 button 回调方式。
3. **删繁就简，组织语言便于理解**。不同于原生 js，livescript 提供了很好的语法结构，使得语言变得通俗易懂，那么在写的时候也更加注重了语言的组织。
4. **Bug 逐渐修正**。Bug 难免会出现，需要不断地修改代码。我更倾向于分模块地来检查，比如 button 的点击请求分为一个模块，bubble 分为一个模块，robot 分为一个模块。在写得过程中，写完一个模块即可在浏览器中检查。最后完成时在总体检查。TOP 过程我理解为一种自顶而下的广搜过程，所以要考虑到方方面面。
5. **Focus on coding,借用第三方插件简化工作**。这或许也是最令我头疼的地方了，目前还无法完全做到 grunt 一次执行，所有工作都完成。还有许多要学习的东西。