Automated Home Solution Approach

# Overview

Automated Home (AH) is to create an extensible smart home application using static page, jQuery and GUI.

This application is designated to integrate all the smart devices in one house in a consolidate control panel. The customer could change the behaviours of smart devices registered in the house. In AH, each smart device is implemented as a plugin. Due to the event-driven architect, it is fairly easy to create a new plugin if there is any new type smart device. It is also very convenient if the customer would like to automatically manage several devices by publishing specific events to the page.

AH is a demo written by James in the weekend. The core JavaScript module is built from scratch; however the GUI module is borrowed from an online example <https://github.com/marybeshaw/Just-Another-Automated-Home>. The server storage is currently mocked as required but it can be easily integrated with another REST API.

# Requirement

A JavaScript application simulating house automation: pressing a button on a control panel would visually turn on a light, change the temperature or close the curtains. Some constraints:

* The application must use jQuery.
* The components must have HTTP based "server" interaction (use a static file for simplicity, data persistence is not required). For example, the heating component retrieves the current temperature from the server and also sends the desired one back to the server.
* The solution has to be extensible and documented, so that we can develop our own components that react to events.

The application will be executed on a plain HTTP server with no possibility to run code server side and is being viewed in 2 major browsers of your choice.

# Features

* Requires jQuery only
* Event-driven architecture
* 3 build-in components: Heating, Lights & Curtains
* A demo to emulate automatic device control based on the hours of the day.
* Easy to extend with new component or interact with the existing components
* JSON based configuration file

# Solution Approach

## Definition of components

All the smart devices are abstracted as a component that can be plugged into one house. The definition of the house and the components are persisted into a JSON format file. There are two major items in the configuration file.

### Components

One component represents one active smart device in the house. Components are assembled by their category. The category is linked with a plugin that needs to be loaded in the application.

Each component has got a unique identifier. The ‘type’ attribute is to distinguish the sub-category of devices. The ‘data’ attribute holds the running time data that a component will persist. Here is a live example of a light component.

1. "light-001":{
2. "type":"smart-led",
3. "data":{
4. "domId":"light-001",
5. "name":"Light",
6. "status":"on",
7. "from":18,
8. "to":21
9. }
10. }

### Floor plan

This section in the configuration holds the location of smart devices in the house. The component array includes all the unique componentIds. The svgId is used by visual effect. It is the same as the section id in the floor plan SVG file.

1. "living-room":{
2. "svgId":"g-living-room",
3. "components":["light-001","curtain-001"]
4. }

## Component Plugin

Each type of smart devices is implemented in a separate plugin file. Each plugin file needs to register itself in the Home JavaScript in order to initialize properly. In the implementation of Plugin, there are two main types of functions. Repaint functions is to interact with GUI. Meanwhile, Add/Change/OnReady function is related with Publish/Subscribe particular event. It is easy to create your own plugin by referring the implementation of Light component.

## Event driven approach

As AH is an extensible solution, the solution is using an event-driven architecture. A Pub/Sub framework is implemented based on JavaScript. Each component will be reacted with a few standard events like “refresh” and “change”. The customized event subscriber could also be added into the plugin file.

Refresh event is triggered at application initialization as well as the behaviour changing. Refresh event could not just be triggered directly but also be triggered by another event.

Change event is triggered when the user change the behaviour of the component. It could be triggered from GUI or another timer event.

Customized event example can be found in Light component. There is an out-of-box feature that emulates the automatic device control using customized event.

## Server communication

AH is currently mocking all dynamic requests to the server. The status of the whole house is representing in a JSON file. It could be replaced by a REST API which retrieves the data from a database or other storage.

## Interaction Log

The History component on the GUI is to record all the changes made by either the customer or triggered by internal events. It could be used to troubleshoot the event publish/subscribe in runtime.