Just Another Automated Home

A home automation app & GUI by Mary Shaw

I built this home automation project for an interview at Adobe Systems. The goal was to create an extensible home automation app with visual effects, using a static page, jQuery, and some basic asynchronous server interaction.

I looked at some examples online, but in the end I chose to build my own from scratch. My app is set up to support any SVG, with its accompanying JSON describing the home, its rooms, & its automated features.

The Official Requirements

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.

The Logical Pieces

1. Home: The home automation starts with a home. A home can have automation features that work at the whole house level. My example includes temperature at the whole-home level.
2. Rooms in Home: The automated home contains rooms, each with automation features just for that room. The room's SVG effects are controlled by an SVG element ID. My example includes a living room, a few bedrooms, a kitchen, an entrance hall, and a sun deck out back.
3. Home/Room Features: The system is set up with a flexible set of automation features, which can belong to either the whole house or a single room. My example includes lights, curtains, and temperature.
4. Feature Data Types: The home automation features boil down to different data types, which define how the control panel is set up and what gets saved to the server. My example includes Boolean and Integer types. The lights & curtains are Booleans and the temperature is an Integer.

Setting up the Web Page

The web page markup isn't hugely involved. It has just a few necessary pieces.

1. JQuery dependency: JQuery version 1.11.2 is loaded from a CDN.

2. Data Types JS file: Defines the way features with specific data types act, with regards to setting up the control panel, saving via AJAX, and animating the SVG component.
3. Home Setup JS file: The app's main workhorse.
4. CSS Styles, which could be swapped out to change the page's look & feel.
5. A control panel div, where the control panel form elements are dropped when the home is initialized.
6. An SVG file, a floor plan map of the home, with element IDs used to animate the home.
7. A call to the "initHomeData()" function which loads the home JSON file & performs the setup.
8. A reference to the HOME JSON file, which is loaded via AJAX and defines almost everything: the home, the feature types it uses, the home-level features, its rooms, and the room-level features.

Feature Type Data Types

There are currently two supported data types: integer and boolean. To add another one, create an element in the JADATA.dataTypes object. The array key will be referenced by features that use this data type, so it is important. And you'll need to define the functions controlHTML(), controlValue(), and effect().

```
JADATA.dataTypes[key] = {
   "controlHTML": function (featureType, feature, domID) {
      // set up some HTML here for the control panel,
      // and return that HTML;
      return newHTML;
   },
   "controlValue": function (featureType, feature, domID) {
      // return the control's actual value,
      // which may or may not be what is in the form field
      return theValue;
   },
   "effect": function (featureType, feature, domID, controlValue, svgElement)
{
      // make a change to the SVG element
      // based on the feature's info & the control state
   }
};
```

Feature Types

There are currently 3 supported feature types: lights, curtains, and temperature. To add another one, add something to the "features" attribute of your home object.

All feature objects should have a name and a data type. The other attributes depend on your data type.

Boolean Type Features

A Boolean needs a "true" value and a "false" value. And, the true & false opacity changes for controlling the SVG effect.

```
"curtain": {
   "name": "Curtains",
   "data-type": "boolean",
   "true-value": "Closed",
   "false-value": "Open",
   "true-opacity-change": 0.2,
   "false-opacity-change": -0.2
}
```

Integer Type Features

An integer needs a “units” value (the temperature units is degrees Fahrenheit). It also needs a min/max, to help tell the HTML control how to behave.

```
"temp": {
   "name": "Temperature",
   "data-type": "integer",
   "units": "&#8457;",
   "min": 60,
   "max": 90
}
```

Rooms

A room object within in the home object needs to have the following attributes: name, svg-id, and features (also an object). The key is important.

```
"entrance-hall": {
   "name": "Entrance Hall",
   "svg-id": "g-entrance-hall",
   "features": { … }
}
```

Room & Home Features

The home & room objects have a list of features controlled at the whole-home level. To add a new feature, add something to that JSON object. The attributes need to contain a name, a feature type, and a current value (for initializing the page). The description is optional. The key is important.

```
"temp": {
   "name": "Temperature",
   "description": "Control the temperature of the home (Fahrenheit)",
   "feature-type": "temp",
   "current-value": 68
}
```

The SVG home map graphic

You can use just about any SVG home graphic; the important thing to remember is that each room needs its own ID for animation, and each room ID needs to be labeled in the HOME JSON file.