Short:

* The player sits on an SSH-accessible device
* The manager has a login to the device and can SCP files (defined below) to a directory
* This directory is served over a non-caching HTTP server, bound to ::1 only
* After sending files it simply resets/refreshes the client looking at the files (Chromium)
  + This avoids us putting down the device while updating, e.g. if we transfer a several hour HD video of an aquarium
* If anything goes wrong, who knows?
* If we want to delete old media?
* This is all ASAP, the manager will be more technically involved.
* We technically have **half a KLOC** here ourselves, if we count the messy OWM plugin thing

# Transferred Files Specification

Several (2) classes of files are defined. We define the path `~` here to be the application root directory, from which the local-only HTTP server serves static files.

## Content Files

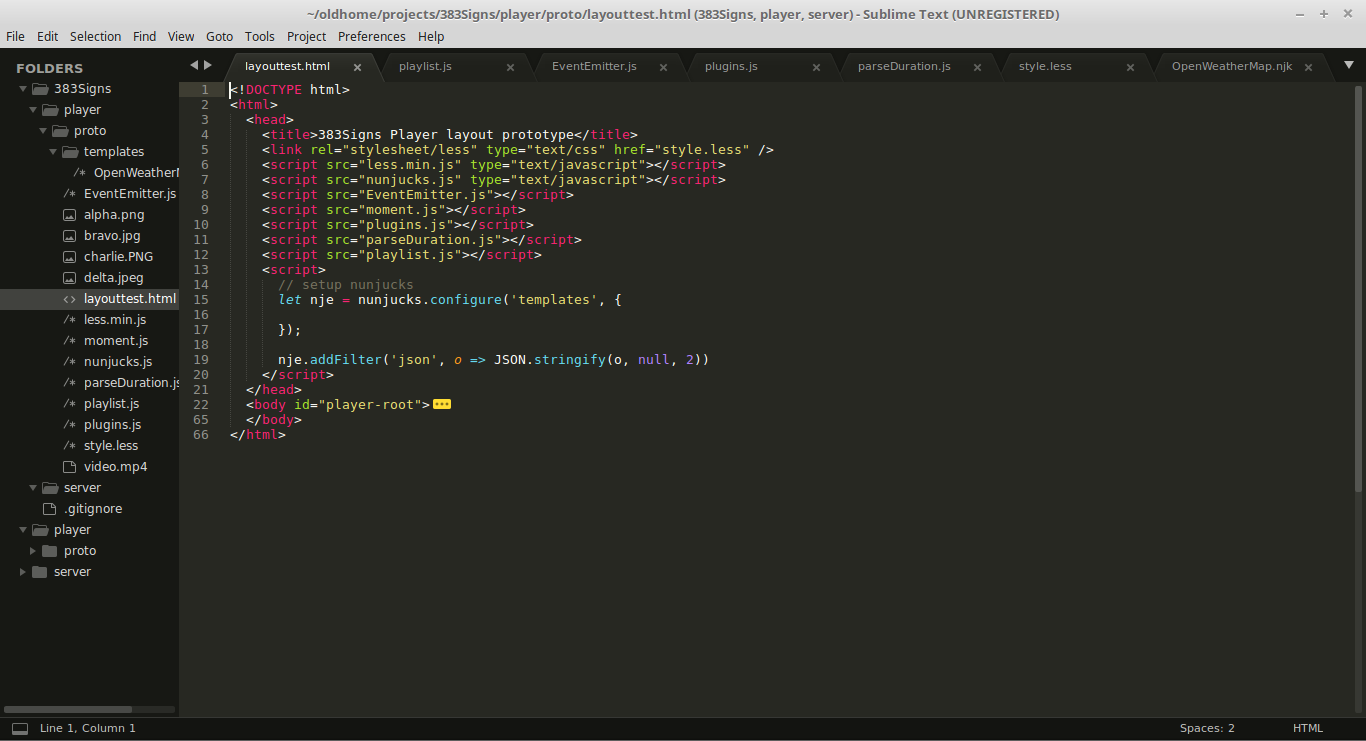
These files are those that are displayed. They include images and videos. Anything that is supported by Chromium is acceptable. The destination shall be ~/content/$filename. These will be pointed to by “media items” later...

## Template File

This file defines the template for playing. It is an HTML files with a standard container (containing wrapping elements, required script references). In the body/dynamic/configurable section is the data defining the content played, plugins, positioning, timing, and other aspects. The inner body should be plain HTML with no references to external files (excluding content files). Much positioning and display data are described as classes, all other data are in data-xxx tags. Possible patterns are described below.

Note, as much of the layout as possible should be defined with flexboxes. Though not exposed here, keep it in mind.

### Outer Template



This is an earlier functional prototype of the outer template.

NB: the ever helpful document construction engine has gone and capitalized everything at the start of a line, use lowercase letters!

### Template

A template is defined to be a single set of content/layout to be played, with `.template`. These exist directly under #player-root.

**Properties**

* class=”template”
* data-next: Defines when this template should be done and we move to the next
  + “All-playlist”: Wait for all playlists under this template to be finished (DEFAULT)
  + “Any-playlist”: Wait for any playlist under this template to be finished (NOIMP)
  + “Duration”: Wait a specified amount of time as defined in data-duration
* Data-duration: Defines how long to wait for data-next=”duration”. See duration appendix

### Container

A layout scheme. Must be `.container`. Must be either `.horizontal` xor `.vertical`. May contain other containers.

**Properties**

* None really?

### Content

An element containing content, either a single media element or a playlist. Must be `.content`. Should probably be `.auto`, for an equally weighted flex, though more should be implemented later. May be `.playlist`, see playlist below

**Properties**

* Classes
  + Auto: Flex to one width with no basis.
  + Playlist: Designates that we contain a playlist. See the playlist section

### Playlist

A content element that contains a playlist. Must be `.playlist`. Must contain one or more `div.playlist-item`. Playlist items may contain media elements.

**Properties**

* None

### Media Item

The elements containing visual media. Generally img or video, though anything will fit.

Elements with native properties still respect those. For example, a video should likely have the autoplay and loop attributes. Both img and video must have src attributes defined.

**Properties**

* Class
  + Distort: Distorts/stretches the item to fill its allotted space.
  + Fit: Makes the item as large as possible to fill its space without distorting
* Data-duration: If in a playlist item, defines how long to show this item. Default is defined elsewhere