susi tech doc

Some random information

susi builds up from two javascript user interfaces, the other is the admin side which is not meant to be public, and the public user interface which will allow only to view the presentations built by admins.

Susi contains no real backend engine, almost all code is javascript. Php is only used to resize images and save them to the servers hd.

Database in use is elasticsearch, which is document based database system with very powerful search functionalities. All data is sent to and retrieved from the database via get and post requests directly from the client computer. this is by no means anything even remotely close to a secure system.

> *note: even though elasticsearch is document oriented database, it still has schema. When trying to save differently formed data into the database with the same _type-attribute, it will produce some weird errors. So datatypes of objects must be the same, for example you cannot change the name-field of the "marker"-type object into boolean, it must be string.*
>
> *Schema is created into database when first object is saved. Different types of objects can be saved by changing the _type attribute of the object. this will cause the elasticsearch to save that object into different index.*

Folders

/
contains index.html to start up the client presentations and some javscript files that are used by the client and admin interface.

/admin
contains admin interface and related files

/leaflet
self-explanatory?

/lib
contains javascript libraries used by susi, like jquery, hammer and so on.

/markericons
icon images for markers

/user
user interface files

Client

Client application works as the end user interface for susi, which can view the presentations made with the admin tool.

Files related to the client application are found in the /user directory.

presentation.js

Main functionality of user interface, handles user actions, theme loading, adding layers to the map and so on..

layerloader.js

responsible for loading the layers from database and creating leaflet layers from them. layerloader-function has different methods for loading different types of layers, type of the layer being loaded is defined in the data stored in to the database.

file contains also some helper functions, such as “loader”.

leaflet-extension.js

this file contains some extentending prototypes for leaflet layers for more functionality. compatibility with newer versions of leaflet is questionable.

map.js

This is the presentation map object which is responsible for controlling the underlying leaflet map. this is used by the presentation.js.

markericon.js

contains functions for creating the html marker icon with images. returns L.DivIcon

menu.js

contains menu object and button object used in menu. this is the main menu that is visible in presentation.

style.js
contains admin defined style information, it is loaded when presentation starts and is used by menu and presentation objects.

style.js has become obsolete and difficult, and is not necessarily technically required, but removing it will require some hunting, where it has been used and replacing the code. it is just a storage unit for style info and it is capable of giving out specific style data by request.

touchable.js
early version of transformable object… in other words, an html div object that could be manipulated with touch events. does not work correctly and should be sacked.

history.js
stores presentation states so back-forward functionality could be implemented.

breadcrumbs.js
this stores information about layers and can be used to show user what layers are currently on… something like breadcrumbs on web pages. breadcrumbs will monitor the map and update itself when layers are added or removed.

Admin

The susi admin interface, which allows the user to create and edit presentations and save them into the database. files related to admin interface are located in the /admin directory.

index.html
is the main html page. buttons and other clickable objects work by using the “actions.js”, functionality is defined in html element parameters “data-action” and “data-target”.

actions.js
this contains functionality of the admin interface, all clickable buttons and objects use this to function. Editor toggling, layer loading, project saving etc are all found here.

[some-name]editor.js
these files contain “editor”-object for leaflet layers, these will interact with the interface and allows the user to edit leaflet layer objects, for example put markers on a map, or draw lines.

browser.js
is the repository browser window. extends editorwindow. this offers functionality to “browse” the repository stored in the database.

editorwindow.js
is basic window for admin interface, used by all objects that open up in different window in admin view.

imageinput.js
small class, or object to create image input elements into the admin interface. offers functionality to drag & drop or select images in to the input field.

imageupload.js
image uploading window for uploading images to the server.

inputset.js
class for creating multiple inputfields into web page. this also allows data to be loaded and retrieved from the fields. Also supports selectable lists.

fields can be defined as follows:

```
myInputs = new inputSet(
                        [
                                {
                                        name:'button_rotate',
                                        type:'boolean',
                                        caption:'rotate buttons',
                                        default:'true',
                                        title:'randomly rotate menu buttons'
                                },
                                {
                                        .. another field...
                                }
                        ]
                        );
```

example of this functionality is in styleeditor.js which contains almost all field types supported by inputset.

inputset html element can be retrieved by the .getElement() function, which will return jquery html element, which in turn can be placed into dom. For data, getData() and load(data). Loading format is the same as data coming out.

langselector.js

small language selector bar. used by some input windows to select different language to be stored.

picker.js

is the "picking" window, or the image selector. allows user to select items from the window and returns selected items after selection is complete, supports singe or multiple selections.

styleeditor.js

this is the presentation style editor window, which allows the user to edit style properties for the presentation.

php image scaler

image.php is the image handler for susi, it can crop, resize and save images into the server which are sent as POST-requests from the ui.

Images are send to image.php as POST, the data is JSON-object.

```
post data is formed as follows:

data = {
	// array of image files
	files : [
	{
		data:image data as base64 string,
		type: image type (jpg/png/gif),
		size: file size in bytes,
		name: file name,
		origheader: original file header (image/jpeg, image/png)
	}

	],
        // resize: processed when upload complete
        resize:[[128,128],[64,64]] // array of sizes to save on server

    }
```

get an image by GET request (via URL)

Images can be requested from the image.php, either in full original resolution or scaled / cropped to certain resolution specified by the request. Imagehandler will check if the file exists in requested resolution, if not, the original image will be read and scaled to the requested resoltion and saved to the server.

crop image to certain height & width

```
image.php?img=[filename]&x=[resolution_x]&y=[resolution_y]
```

scale to height

```
image.php?img=[filename]&y=[resolution_y]
```

Original image

```
image.php?img=[filename]&full=true
```

Images cannot be deleted through php by requests.