Lapstone Framework

# Content

# Coding Conventions

## Pages

### Create a page

To create a page you have to do the following steps in any order:

* Create a HTML file /pages/page.<page name>.html.
  1. Copy the the template HTML in the file.
  2. Search for ##page and replace it with <page name>.
* Create a javascript file under /js/page/page.<page name>.js.
  1. Copy the template javascript in the file.
  2. Search for ##page and replace it with <page name>.
* Create a json file /js/page/page.<page name>.json.
  1. Copry the template json in the file.
  2. Search for ##page and replace it with <page name>.
* Register the page in the /js/page/pages.json file.

Now the page is created and can be used in your app.

### Templates

### Configuring a page

The page.<page name>.json file contains a set of obligate parameters. You can extend them if you need your special parameters.

#### Obligate fields

##### name

Your <page name>.

##### shortname

Reserved for the future. Please fill it with your <page name>

##### template

Define your page template. An empty string for no page template.

##### asyncLoading

true or false

If you want to use asynchronous page loading or not.

##### useKeepAlive

true or false

More details at the plugin.KeepAlive section.

##### loginObligate

true or false

More details at the plugin.Session section.

##### isGlobalPage

false

Deprecated mechanic. Will be updated in future versions.

##### contentRefresh

true or false

Automatically reload the page after [contentRefreshInterval](#_contentRefreshInterval) seconds.

##### contentRefreshInterval

int (>0)

Interval in seconds when the page should be reloaded. Necessary when [contentRefresh](#_contentRefresh) is true.

#### Sample configuration

{

"name": "competenceProfile",

"shortname": "competenceProfile",

"template": "DakoraGridPage",

"asyncLoading": true,

"useKeepAlive": true,

"loginObligate": true,

"isGlobalPage": false,

"contentRefresh": false,

"contentRefreshInterval": 0

}

### Minimal structur of a page / Create the DOM

Depending on your configuration in the json file lapstone calls the [creator()](#_creator()_1) or the [async.creator()](#_async.creator()_2) function in your page.<page name>.js.

#### config

The config object contains the object which is defined in the page.<page name>.json file.

Have a look [how to configure](#_Configuring_a_page) a page.

#### include

Files which are defined in the include array are loaded every time before the [creator()](#_creator()) or the [async.creator()](#_async.creator()_1) function is called.

#### include\_once

#### elements

If you using page templates the the async.elements object is containing the jQuery objects.

#### constructor()

On startup lapstone calls the constructor() function of every page. The plugins are already loaded at this time.

You have to return a jQuery.Deferred().promise() object.

#### creator()

Create your HTML page in the function body. Please use the [DOM manipulation conventions](#_DOM_manipulation). If you are using a [page template](#_Templates) the elements object contains the jQuery objects.

Now DOM manipulation is over and lapstone will run a jquery enchantment and shows the page.

#### async.promise

Todo – should contain the jQuery promise.

#### async.result

After the jQuery.Deferred() object returned by [async.creator()](#_async.creator()) is either rejected or resolved the async.result object contains the result.

#### async.elements

If you using page templates the the async.elements object is containing the jQuery objects.

#### async.creator()

Create your HTML page in the function body. Please use the [DOM manipulation conventions](#_DOM_manipulation). If you are using a [page template](#_Templates) the async.elements object contains the jQuery objects.

You have to return a jQuery.Deferred().promise() object. E.g.: a asynchronous webservice call: app.rc.getJson()

#### async.done()

After resolving the deferred object the async.done() function of your page is called.

The async.result object of your page contains the <parameter> of the jQuerry.Deferred.resolve(<parameter>) method.

Create your HTML page in the function body. Please use the [DOM manipulation conventions](#_DOM_manipulation). If you are using a [page template](#_Templates) the async.elements object contains the jQuery objects.

Now DOM manipulation is over and lapstone will run a jquery enchantment and shows the page.

#### async.fail()

After rejecting the deferred object the async.done() function of your page is called.

The async.result object of your page contains the <parameter> of the jQuerry.Deferred.reject(<parameter>) method.

Create your HTML page in the function body. Please use the [DOM manipulation conventions](#_DOM_manipulation). If you are using a [page template](#_Templates) the async.elements object contains the jQuery objects.

Now DOM manipulation is over and lapstone will run a jquery enchantment and shows the page.

#### async.always()

After resolving or rejecting the deferred object the async.done() function of your page is called.

The async.result object of your page contains the <parameter> of the jQuerry.Deferred.resolve/reject(<parameter>) method.

Create your HTML page in the function body. Please use the [DOM manipulation conventions](#_DOM_manipulation). If you are using a [page template](#_Templates) the async.elements object contains the jQuery objects.

Now DOM manipulation is over and lapstone will run a jquery enchantment and shows the page.

#### async.abort()

Todo – cancle the deferred object

#### setEvents()

Register events in the setEvents() function of the page. Nowhere else!

To avoid a huge amount of events within you app lapstone has a mechanism for unbinding and rebinding events.

Declare your events in the following way:

$(“#<page name>”).on(“<event name>”, “selector”, function(event){});

or use the page id form your configuration:

$(this.config.pageId).on(“...

Lapstone will unbind the events after leaving the page and rebinds it when you come back again.

#### functions

Write your private page functions in the functions object.

#### events

The events object contains the events triggered by jQuery mobile. You can use them, but they will be updated when jQuery mobile v.2.0 is released.

## Plugins

### Create a plugin

To create a plugin you have to do the following steps in any order:

* Create a javascript file /js/plugin/plugin.<plugin name>.js
  1. Copy the the template HTML in the file.
  2. Search for ##plugin and replace it with <plugin name>.
* Create a json file /js/plugin/plugin.<plugin name>.json.

1. Copry the template json in the file.
2. Search for ##plugin and replace it with <plugin name>.

* Register the page in the /js/page/pages.json file.

Now the plugin is created and can be used in your app.

### Configuring a plugin

#### Obligate fields

##### name

string

Your <plugin name>

##### shortname

string

A short name to acces the plugins [public functions](#_functions_1).

e.g.: app.<shortname>.<public function>();

#### Minimum sample configuration

{

"name": "Exa",

"shortname": "exa"

}

### Functions and fields

#### config

The config object contains the object which is defined in the plugin.<plugin name>.json file.

Have a look [how to configure](#_Configuring_a_plugin) a plugin.

#### constructor()

The constructor() function is called once when the plugin is loaded on startup.

You have to return a jQuery.Deferred().promise() object.

#### pluginsLoaded()

The pluginsLoaded() function is called when all plugins constructor() functions have been run. At this point you have access to the the plugins [public functions](#_functions).

You have to return a jQuery.Deferred().promise() object.

#### pagesLoaded()

You have to return a jQuery.Deferred().promise() object.

#### definePluginEvents()

#### afterHtmlInjectedBeforePageComputing()

#### pageSpecificEvents()

#### functions

## Lapstone startup

Just for information. Do not care if you have no idea.

1. Lapstone initialisation
2. Load the configuration for lapstone. /js/lapstone.json
3. Load plugins.
   1. Load plugins configuration
   2. Verify the plugins configuration.
   3. Load the plugins and call the constructor.
   4. Verify the plugins.
   5. Calling the plugins loaded event.
   6. Define the plugin events.
4. Load pages.
5. Update framework.
6. Enchant pages.
7. Wait for jQuery mobile mobileinit event.
8. Wait for apache cordovas deviceready event.
9. Trigger lapstone event.

## DOM manipulation

Use jQuery and nothing else.

1. Create the HTML element without attributes. $(“<div>”)
2. Add classes to the element. .addClass()
3. Add attributes to the element by using an attribute object. .attr({})
4. Append other HTML elements. .append()
5. Do not use the .css() function. For styling use LESS. [Documented here](#_Styling).  
   It’s allowed to use .css() just for .css(“display”:”none”).

# Styling

## Skinning

## Styling

In debug mode Lapstone uses LESS to style the apps. Get further information at lesscss.org.

The style files are locadted under the /css/ folder. Depending on using the skin plugin the files are in the /css/ folder or in the /css/skin/<skin name>/ folder.

Lapstone comes out of the box with three skin files:

* colors.less

Define all of your colors here.

* fonts.less  
  Define all your fonts here.  
  The folder for fonts is ./fonts/
* global.css.less  
  @import all other less files in this file.

In release mode every <style name>.css.less file is mapped an compressed to a <style name>.css file. The LoadExternalScripts or the Skin plugins will automatically loads CSS version of your style.

### Style file conventions

1. Create a page.<page name>.less file for each page in your app.
2. Create a layout.<layout name>.less file for each style overlap in your pages.
3. If you need more style files in your app add <style name>.css.less files. Add it to the plugin\_LoadExternalScripts.config or plugin\_Skin.json file.