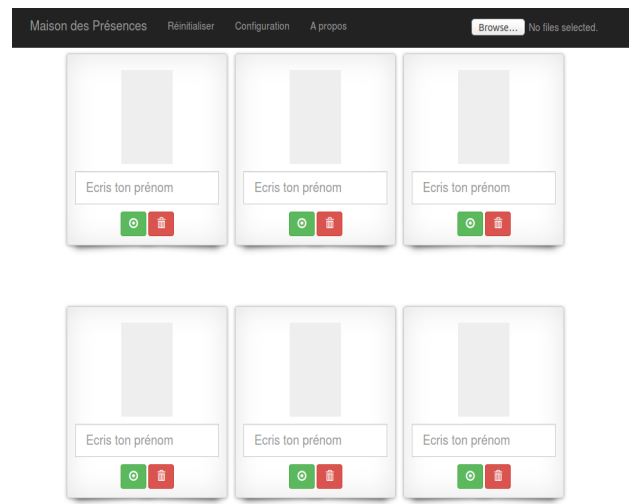


Maison des Présences (MDP)

Introduction

Maison des Présences is a light-weight Web app built with HTML5 and Javascript: jQuery and Bootstrap frameworks. Elementary school students may learn correct spelling of their own name as well as try to remember the names of other kids in the class. Audio and visual hints are given, whether the names were correctly spelled.

The app is built in a privacy preserving way so that the student photos stay local within the browser and do not traverse the Internet. The difficulty level of the exercise may be increased with different settings by for example taking into account accent and case sensitivity in input text.



Installation package



MDP.ZIP

Index.html



CSS

Gallery.css
Toggle-switch.css
Bootstrap.css
Shadows.css



FONTS

Glyphicons-halflings-regular.eot
Glyphicons-halflings-regular.svg
Glyphicons-halflings-regular.ttf
Glyphicons-halflings-regular.woff



JS

App.js
db.js
Simplestorage.js
Shuffle.js
Jquery.easy-confirm-dialog.min.js
Latinise.min_.js
Holder.js
Bootstrap.min.js
Load-image.min.js



SND

ko.ogg
ok.ogg

MDP package should be installed on a Web server under “Web-root” directory. (In case of Apache httpd this is htdocs-directory). Please name a sub-directory to your liking (e.g. ./MDP/) where to extract MDP.ZIP contents. The proper functioning of the application requires also Internet connectivity from the client device / Web browser (due to the usage of certain online JS-libraries). Main files include:

- **index.html** – main Single Page Application file, graphical layout of HTML5 components.
- **app.js** – jQuery javascript application, with handler functions for user actions.
- **db.js** – IndexedDB javascript persistent storage interface for app.js (images and metadata).
- **Gallery.css, Shadows.css**, etc. – Stylesheets for visual effects of portrait photo placeholders.
- **OK.ogg, KO.ogg** – Sound files for correct and incorrect name input values.

Application configuration

MDP is intended to be used in two versions:

<http://my.server.com/MDP/?mode=p> (for Teacher with full functionality)

and

<http://my.server.com/MDP/?mode=e> (for Student with limited functionality)

There is no password to protect the access to these different views of the same application, so the URLs are better to be kept secret from the students at least. The lastly used view under configuration is preserved for next MDP application session if no mode parameter value is given.

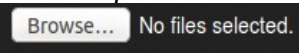
MDP uses HTML5 *localStorage API* and *IndexedDB API* for persistently storing pictures and associated metadata locally within the Web browser, while the target environment is Mozilla Firefox 33.0. The use of *IndexedDB* requires that in the Firefox browser settings:

“Preferences->Privacy->Remember history”

... should be enabled. If it is not, an Alert pop-up message will be shown.

Application preparatory tasks for the Teacher

After opening the MDP teacher view, a 5 x 5 -grid of empty portrait photo placeholders is presented. The teacher should have class portrait photos on USB stick or disk drive ready for importing them into MDP. Please note that modern digital cameras produce very accurate and large image files, that should be scaled down for smooth Web browsing. Firefox has 10Mb for *localStorage* and 50Mb for *IndexedDB* data size quotas, which in case of *IndexedDB* can be extended with user permission¹. The images can be imported to MDP in two ways:

1. Press the “*File upload*” - button and select one or several appropriate image files (*.jpg,*.gif, *.png). 
2. Through “*Drag 'n' Drop*” - functionality: Open file browser next to MDP browser window. Select one or several appropriate image files with mouse left-button and Ctrl / Shift -key pressed down. Then move the file selection while mouse left-button pressed down on top of the MDP browser window and release the mouse left-button.




The images should appear in the MDP application and equal amount of empty placeholders for photos are replaced. MDP aims to scale the pictures to target size of **200x250** pixels by default, while conserving aspect ratio. Different image manipulation options can be adopted to **app.js** from Appendix A. The name fields show initial values from the imported picture files. It is assumed that each picture file is named after the student in the picture: e.g. Pierre.png. Consequently the corresponding name input field should show “Pierre” in this case. The teacher may change the correct naming and remove

¹ <http://www.html5rocks.com/en/tutorials/offline/quota-research>

Installation and User guide

10/22/14

unwanted photos from the class composition as well:

1.  “Update” button may be used to store correct representation of student name and to store it in the IndexedDB.
2.  “Delete” button may be used to remove a picture photo and associated name from the IndexedDB.
3.  “OK” button is shown only in the student view. After typing the name of the person in the picture student may press either this or “Enter” from keyboard to validate the entry for feedback.

The other configuration options can be found under the main toolbar:



Image 1: Configuration pop-up window

After these preliminary steps the MDP application is ready to be used. The student version of MDP should show only **Class name** as title heading and **Student photos** with **Name input fields** next to **OK**-button. The main toolbar is hidden from the student. The IndexedDB data (pictures and names) are preserved for future application sessions. Even the clearing of the Firefox cache memories do not affect the MDP application data. If there is a need to reset the system to its initial state, the teacher may press the “Reset” menu item from the main toolbar which brings in the following pop-up.

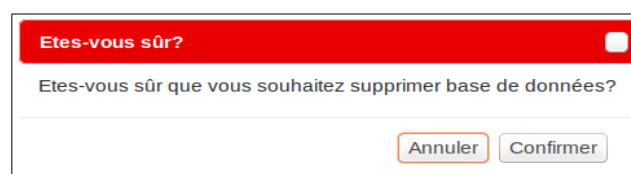


Image 2: Resetting the IndexedDB to its initial state

APPENDIX A Image Manipulation options of the 3rd party library used in MDP
(<https://github.com/blueimp/JavaScript-Load-Image#options>)

Options

The optional third argument to **loadImage()** is a map of options:

- **maxWidth**: Defines the maximum width of the img/canvas element.
- **maxHeight**: Defines the maximum height of the img/canvas element.
- **minWidth**: Defines the minimum width of the img/canvas element.
- **minHeight**: Defines the minimum height of the img/canvas element.
- **sourceWidth**: The width of the sub-rectangle of the source image to draw into the destination canvas.
Defaults to the source image width and requires *canvas: true*.
- **sourceHeight**: The height of the sub-rectangle of the source image to draw into the destination canvas.
Defaults to the source image height and requires *canvas: true*.
- **top**: The top margin of the sub-rectangle of the source image.
Defaults to 0 and requires *canvas: true*.
- **right**: The right margin of the sub-rectangle of the source image.
Defaults to 0 and requires *canvas: true*.
- **bottom**: The bottom margin of the sub-rectangle of the source image.
Defaults to 0 and requires *canvas: true*.
- **left**: The left margin of the sub-rectangle of the source image.
Defaults to 0 and requires *canvas: true*.
- **contain**: Scales the image up/down to contain it in the max dimensions if set to *true*.
This emulates the CSS feature [background-image: contain](#).
- **cover**: Scales the image up/down to cover the max dimensions with the image dimensions if set to *true*.
This emulates the CSS feature [background-image: cover](#).
- **aspectRatio**: Crops the image to the given aspect ratio (e.g. 16/9).
This feature assumes *crop: true*.
- **crop**: Crops the image to the maxWidth/maxHeight constraints if set to *true*.
This feature assumes *canvas: true*.
- **orientation**: Allows to transform the canvas coordinates according to the EXIF orientation specification.
This feature assumes *canvas: true*.
- **canvas**: Returns the image as [canvas](#) element if set to *true*.
- **crossOrigin**: Sets the crossOrigin property on the img element for loading [CORS enabled images](#).

Installation and User guide

10/22/14

- **noRevoke:** By default, the [created object URL](#) is revoked after the image has been loaded, except when this option is set to *true*.

They can be used the following way:

```
loadImage(  
    fileOrBlobOrUrl,  
    function (img) {  
        document.body.appendChild(img);  
    },  
    {  
        maxWidth: 600,  
        maxHeight: 300,  
        minWidth: 100,  
        minHeight: 50,  
        canvas: true  
    }  
);
```

All settings are optional. By default, the image is returned as HTML **img** element without any image size restrictions.