Une correction pour le TP

- Voyons un exemple de correction du TP :
  - Premier fichier « **contacts.html** » qui permet de créer un contact et de l'enregistrer : soit dans un fichier texte soit **dans la base localStorage du Navigateur**, soit sur le serveur via un un appel Ajax. Voici le fichier JavaScript associé stockant dans le **localStorage** : « **contacts.js »**

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Security-Policy"
                          content="default-src 'self' data: gap: https://ssl.gstatic.com 'unsafe-eval';
                          style-src 'self' 'unsafe-inline'; media-src *; img-src 'self' data: content:;">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="UTF-8">
    <meta name="format-detection" content="telephone=no">
    <meta name="msapplication-tap-highlight" content="no">
    <meta name="viewport" content="user-scalable=no, initial-scale=1,
                                       maximum-scale=1, minimum-scale=1,
                                  width=device-width">
    <link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
    <!--link rel="stylesheet" type="text/css" href="css/index.css"-->
    <title>Hello World</title>
</head>

<body>
    <nav class="navbar navbar-default">
      <div class="container-fluid">
        <div class="navbar-header">
          <a class="navbar-brand" href="#">Contacts Greta</a>
        </div>
        <ul class="nav navbar-nav">
          <li class="active"><a href="index.html">Accueil</a></li>
          <li><a href="contacts.html">Contacts</a></li>
          <li><a href="countries.html">Liste des Pays</a></li>
          <li><a href="#"></a></li>
        </ul>
      </div>
  </nav>
(...)
```

Une correction pour le TP

- Voyons un exemple de correction du TP :
  - Premier fichier « **contacts.html** » (suite) :

```
<div class="tab-content">
  <form class="form-horizontal" role="form" id="contactform" method="POST" accept-charset="utf-8" action="/modifierMonProfil">
    <div class="form-group">
      <label for="titre" class="col-sm-2 control-label">Titre/Name/Firstname : </label>
      <div class="col-sm-2">
        <select class="form-control" id="titre" name="titre" required="required" >
            <option value="M" selected="">Monsieur</option>
            <option value="Mme">Madame</option>
            <option value="Mlle">Mademoiselle</option>
        </select>
      </div>
      <div class="col-sm-4">
        <input class="form-control" id="nom" placeholder="Name" value="" name="nom"  type="text">
      </div>
      <div class="col-sm-4">
        <input class="form-control" id="prenom" placeholder="Firstname" value="" name="prenom"  type="text">
      </div>
    </div>
    <div class="form-group">
      <label for="adresse1" class="col-sm-2 control-label">Adress 1 : </label>
       <div class="col-sm-10">
         <input class="form-control" id="adresse1" placeholder="Primary adress" value="" name="adresse1"  type="text">
       </div>
    </div>
    <div class="form-group">
      <label for="adresse2" class="col-sm-2 control-label">Adress 2 : </label>
      <div class="col-sm-10">
        <input class="form-control" id="adresse2" placeholder="Additionnal adress" value="" name="adresse2"  type="text">
      </div>
    </div>
    <div class="form-group">
      <label for="code_postal" class="col-sm-2 control-label">Zip Code : </label>
      <div class="col-sm-2">
       <input class="form-control" id="code_postal" placeholder="Zip Code" value="" name="code_postal"  type="text">
      </div>
      <label for="ville" class="col-sm-2 control-label">City : </label>
        <div class="col-sm-6">
          <input class="form-control" id="ville" placeholder="City" value="" name="ville"  type="text">
        </div>
(…)
```

Une correction pour le TP

- Voyons un exemple de correction du TP :
  - Premier fichier « **contacts.html** » (suite et fin) :

```
(…)
</div>
      <div class="form-group">
            <label for="inputName" class="col-sm-2 control-label">Phone</label>
            <div class="col-sm-5">
                  <input class="form-control" name="tel_modile" id="tel_modile" type="text">
            </div>
            <label for="inputName" class="col-sm-2 control-label">Phone</label>
            <div class="col-sm-5">
                  <input class="form-control" name="tel_fixe" id="tel_fixe" type="text">
            </div>
      </div>
      <div class="form-group">
            <label for="email" class="col-sm-2 control-label">Email</label>
            <div class="col-sm-5">
                  <div class="input-group"><span class="input-group-addon"><i class="fa fa-envelope"></i></span>
                        <input class="form-control" name="email" id="email" value=""  type="email">
                  </div>
                  </div>
                  <div class="col-sm-5">
                  <div class="input-group"><span class="input-group-addon"><i class="fa fa-envelope"></i></span>
                        <input class="form-control" name="email2" id="email2" value="" type="email">
                  </div>
            </div>
      </div>
      <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                  <button type="submit" name="button" id="btn_contact" class="btn btn-primary" >Enregistrer contact</button>
            </div>
      </div>

</form>
<!-- /.tab-pane -->
</div>
  <!--script type="text/javascript" src="cordova.js"></script-->
  <script type="text/javascript" src="js/contact.js"></script>
</body>
```

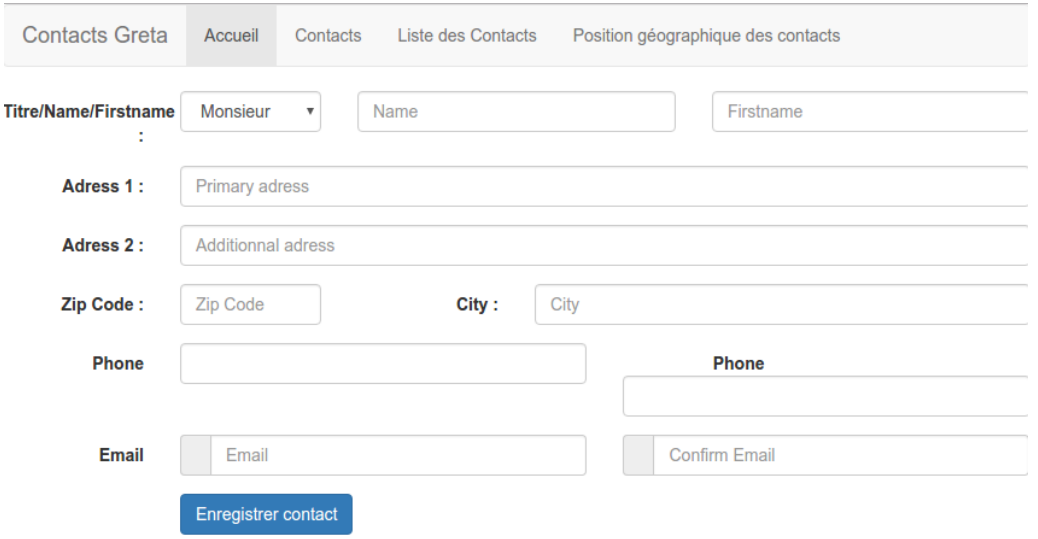
Une correction pour le TP

- Voyons un exemple de correction du TP :
  - Deuxième fichier « **contacts.js**» le fichier JavaScript associé stockant dans le **localStorage** :

```
function initialize() {
    var btn_contact = document.getElementById('btn_contact');
    btn_contact.addEventListener('click', writeContact);
}
function writeContact(evt) {
        // todo : il faut parser le formulaire pour en faire du json.
        var form = document.getElementById('contactform');
        var formData = new FormData(form);
        var contact = {}; // initialisation de la variable contact qui contient l'objet a insérer
        for (var i = 0; i < form.length; i++) {
            console.log('form[' + i + '].name : ' + form[i].name);
            console.log('value : ' + formData.get(form[i].name));
            contact[form[i].name] = formData.get(form[i].name);
        }
        delete contact.button; // le bouton submit est envoyé dans un POST?
        console.log('contact : ', contact);
        var contacts = JSON.parse(localStorage.getItem('contacts')) || [];
        contacts.push(contact);
        localStorage.setItem('contacts', JSON.stringify(contacts));
        evt.preventDefault(); // uniquemement si le bouton est de type Submit.
        form.reset();
    }
/** au chargement de la page contacts.html appel de la fonction initialize() **/
window.onload = initialize();
```

TP : Corrigé

- Voici les screenshots de notre application « contacts »



Une connexion sur une serveur

- Voyons une connexion à un serveur (machine formateur): IP
  - Créez un fichier « **countries.js** » dans le dossier « **www/js** » :

```
window.onload = function (evt) {

    var listCountries = document.querySelector('#listeCountries');
    var xhr = new XMLHttpRequest();
    xhr.timeout = 4000;
    xhr.open("GET", 'http://<IP Machine Formateur>/countriesJSON', true);
    xhr.send(null);
    xhr.onabort = function(e) {
        alert('Error : ' + e);
        window.location = "ma_box.html";
    };
    xhr.onreadystatechange = function () {
        var DONE = 4; // readyState 4 means the request is done.
        var OK = 200; // status 200 is a successful return.
        if (xhr.readyState === DONE) {
            if (xhr.status === OK) {
                var html = "";
                console.log('data here : \n'  + xhr.responseText);
                var data = JSON.parse(xhr.responseText);
                // on construit la chaîne de caractère contenant le code HTML des options de la liste
                for (var i=0;i<data.length;i++) {
                    html += '<option value="'+ data[i]._id +'">'+ data[i].name +'</option>';
                }
                // on intègre de cette façon les données dans la page HTML
                listCountries.innerHTML = html;
            }
        }
    }
}
```

Une connexion distante vers un serveur

- Voyons une connexion à un serveur (machine formateur): IP
  - Créez un fichier « **countries.html** » dans les dossier « **www** » :
    - Les entêtes : <head> ajoutez « **connect-src http://*** » comme ci-dessous : vous pouvez utiliser les entêtes de index.html de l'application générée par cordova.

```
<html>
<head>
    <meta http-equiv="Content-Security-Policy" content="default-src 'self' data: gap: https://ssl.gstatic.com
                                                        'unsafe-eval'; style-src 'self' 'unsafe-inline'; media-src *;
                                                        img-src 'self' data: content:; connect-src http://*">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="UTF-8">
    <meta name="format-detection" content="telephone=no">
    <meta name="msapplication-tap-highlight" content="no">
    <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1, width=device-
    <link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.css">
    <!--link rel="stylesheet" type="text/css" href="css/index.css"-->
    <title>Countries</title>
</head>
(…)
```

Une connexion distante vers un serveur

- Voyons une connexion à un serveur (machine formateur): IP
  - Complétez le fichier « **countries.html** » dans les dossier « **www** » :
    - Les entêtes : <head> ajoutez « **`connect-src http://*`** » comme ci-dessous :

```
<body>
    <nav class="navbar navbar-default">
      <div class="container-fluid">
        <div class="navbar-header">
          <a class="navbar-brand" href="#">Countries from server</a>
        </div>
        <ul class="nav navbar-nav">
          <li class="active"><a href="index.html">Accueil</a></li>
          <li><a href="contacts.html">Contacts</a></li>
          <li><a href="countries.html">Liste des Pays</a></li>
          <li><a href="#"></a></li>
        </ul>
      </div>
    </nav>
        <div class="tab-content">
            <select name="countries" id="listeCountries"></select>
        </div>
  <!--script type="text/javascript" src="cordova.js"></script-->
  <script type="text/javascript" src="js/countries.js"></script>
</body>
</html>
```

Une connexion distante vers un serveur

- Voyons une connexion à un serveur (machine formateur):
  - Normalement avec l'adresse de la machine formateur cela fonctionne car dans la configuration Nginx ont été ajouté les Directives pour le CORS
  - A ajouter dans un vhost Nginx dans « **/etc/nginx/sites-available/default** » :

```
server {
    listen 80;

    server_name 192.168.0.11;  # IP de la machine qui héberge Nginx

    location / {
        proxy_pass http://localhost:3000;  # redirige les requêtes HTTP sur le PORT de l'application NodeJS
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Credentials' 'true';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers'
                           'DNT,X-CustomHeader,Keep-Alive,User-Agent,
                            X-Requested-With,If-Modified-Since,Cache-Control,Content-Type';
                add_header 'Access-Control-Max-Age' 1728000;
                add_header 'Content-Type' 'text/plain charset=UTF-8';
                add_header 'Content-Length' 0;
                return 204;
        }
       (…) // suite slide suivante (…)
```

Une connexion distante vers un serveur

- Voyons une connexion à un serveur (machine formateur):
  - Normalement avec l'adresse de la machine formateur cela fonctionne car dans la configuration Nginx ont été ajouté les Directives pour le CORS
  - A compléter dans le fichier de conf Nginx dans « **/etc/nginx/sites-available/default** »:

```
(…) // suite de la slide précédente (…)
 if ($request_method = 'POST') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Credentials' 'true';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers' 'DNT,X-CustomHeader,
                           Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type';
        }
        if ($request_method = 'GET') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Credentials' 'true';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers' 'DNT,X-CustomHeader,
                           Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type';
                add_header 'Cache-Control' 'no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0';
                expires off;
        }
    }
}
```

- NB : Une bonne habitude, avant de modifier un fichier de configuration toujours en faire une sauvegarder :

  $ sudo mv default defaultOLD
  $ sudo gedit default

Une connexion distante vers un serveur

- Le code pour enregistrer le contact dans MongoDB depuis NodeJS : Voyons le code modifié de « **insert.js** » :

```
var express = require('express');
var router = express.Router();
var mongoose = require('mongoose');
var ObjectId = mongoose.Types.ObjectId;

/* Insert one new user into database. */
router.route('/').post(function (req, res) {
    var path = req.originalUrl.split('?')[0];
       var type = req.method;
    console.log('req.originalUrl : ', req.originalUrl);
    req.body._id = new ObjectId();
    global.schemas[global.actions_json[type+path].model].create([req.body],
        function (err, result) {
            if (err) {
                throw err;
            }
            global.schemas[global.actions_json[type+path].model].find({ _id : req.body._id}, function(err, result2) {
                console.log('Inserted data : ', result);
                if (global.actions_json[type+path].return_type == null) {
                    res.render(global.actions_json[type+path].view, { title: 'Creating User without error with datas below :',
                        data: result2
                    });
                } else {
                    res.send(result2); // renvoie les données insérée pour confirmation
                }
            });
        } // fin callback de l'insert
    ); // fin de l'insert()
}); // fin de la gestion de la route
module.exports = router;
```

Cordova-plugin-camera

- Pour utiliser le plugin camera on l'installe via cordova install :

cordova install cordova-plugin-camera

  - Ensuite on a un peu de HTML à écrire, une balise img qui sera le réceptacle de la photo (en base64).

```
<div class="row">
    <div class="col-md-12">
        <img src="" id="my-photo">
        <button type="button" id="btn_photo"
                    class="btn btn-default">prendre une photo</button>
    </div>
</div>
```

  - Ensuite on a un petit morceau de JavaScript qui va utiliser le plugin et lancer l'appareil photo d'Android

```
window.onload = function(evt) {
  document.querySelector('#btn_photo').on('click', function (evt) {
    navigator.camera.getPicture(onSuccess, onFail, { quality: 50,
      destinationType: Camera.DestinationType.DATA_URL
    });
  });
  function onSuccess(imageData) {
    var image = document.querySelector('#my-photo');
    image.src = "" + imageData;
    // on peut stocker la chaîne base64 qui représente l'image dans le localStorage
  }
});
```

TP : suite

- Vous allez ajouter à votre application Cordava Contacts la possibilité de les visualiser sous la forme d'un tableau HTML : **<table>...</table>.**

  1) Votre tableau devra permettre le tri sur les colonnes (petite image de tri orde croissant uniquement pour l'exercice.)
  2) Votre tableau permettra la sélection d'une ligne (bouton radio sur une colonne) et permettra de connaître l'index du contact dans le tableau du localStorage.
  3) Deux boutons seront à afficher pour supprimer un contact de la base de données, et un pour modifier les données d'un contact : ouverture d'une popup pour afficher le formulaire de modification.

- *NB : La contrainte est d'essayer de créer un **objet réutilisable** sans librairie externe (Jquery DataTable, DHTMLX, etc..) juste du Vanilla.js ;-) permettant d'**afficher un tableau HTML** à partir d'un Tableau JavaScript d'objets.*