Développer avec jQuery Mobile

Framework de développement de sites web mobiles

Table des matières

1 . Introduction au Framework

1.1 . jQuery au cœur de l'internet Mobile

Pour le dire de façon sommaire, jQuery Mobile est une outil logiciel qui fonctionne de manière transparente sur toutes les plate-formes mobiles du moment. Aussi riche que sa parente jQuery, dont la renommée n'est plus à faire auprès des développeurs web, jQuery Mobile est la bibliothèque JavaScript la plus adaptée pour créer des sites web à destination des smartphones et tablettes tactiles.

Avec ses presque 1,2 Milliards d'utilisateurs, soit environ 17 % de la population mondiale, l'internet mobile continu de croître de façon considérable. On estime, grâce à un sondage de Médiamétrie, qu'en 2012 la moitié de la population française y sera connectée. La grande diversités des écrans de terminaux mobiles est devenue un véritable casse-tête pour les développeurs de sites web mobiles, soucieux de ne pas rater le passage à l'ère du tout mobile.
Grâce à jQuery Mobile, il est désormais facile de créer des sites et applications web performantes, qui s'adaptent à tous les types d'interfaces. Aujourd'hui en version 1.0, jQuery Mobile est déjà déclarée *"Innovation de l'année"* par les .Net Awards ! Qu'il s'agisse des fenêtres et composants graphiques d'interface HTML/CSS ou de l'interaction du site avec des données extérieures [base de données sur un serveur distant, géolocalisation avec Google Maps...) grâce à JavaScript, jQuery Mobile (jQM) donne tous les éléments pour construire, avec une agréable facilité, des sites qui fonctionneront sur la plupart des supports mobiles actuels.

L'explosion des ventes de smartphones et le besoin toujours plus grand des internautes d'être connecté au monde a favoriser la croissance de ce type de plates-formes de développement mobile. Pour de nombreux développeurs, jQuery Mobile est tout simplement en tête du classement. L'inclure dans ses projets est un donc choix absolument stratégique.

1.2 . Découvrir Le Framework

1.2.1 Installation de jQuery Mobile

Installer la plate-forme jQuery Mobile se fait aussi simplement que l'installation de jQuery. Il suffit en effet de soit :

- **Télécharger** le fichier JavaScript et de l'appeler dans son fichier html. Il faut dans ce cas télécharger également:
  - La feuille de style **jquerymobile.css**
  - Le dossier d'images à mettre dans le même répertoire que les fichiers.
- **Faire un lien** vers les fichiers JavaScript et CSS directement hébergés par les fondateurs du Framework. Cette méthode a l'avantage de nous éviter de télécharger des documents (JavaScript, images).

Étant donné que jQuery Mobile dépend du Framework jQuery, celui-ci doit être invoqué bien avant l'importation de jQM. Concrètement on aura avoir ceci :

```
<!doctype html>
<html >
</head>
  <title>Ma première Page</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="http://code.jquery.com/mobile/1.1.0-
  rc.1/jquery.mobile-1.1.0-rc.1.min.css" />

  <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.1.0-rc.1/jquery.mobile-1.1.0-
  rc.1.min.js"></script>

</head>
<body >
        ....
</body >
```

1.2.2 Une première application

Un template jQuery Mobile se définit très facilement. Il est presque toujours composé d'une ou plusieurs page, définies dans un document écrit en HTML 5 :

- Une page contenant généralement :
  - Un titre (entête, **header**)
  - Du contenu (**content**)
  - pied de page (**footer**)

Définition d'une page de base en jQuery Mobile :

```
<body >
        <div data-role="page">
                        <div data-role="header">
                    <h1>Un Titre</h1>
                </div><!-- /header →
                <div data-role="content">
                    <p>Hello world</p>
                </div><!-- /content -->
        </div><!-- /page -->
</body >
```

Et avec un exemple complet :

```
<!doctype html>
<html >
</head>
   <title>Ma première Page</title>
 <meta name="viewport" content="width=device-width, initial-scale=1">

 <link rel="stylesheet" href="http://code.jquery.com/mobile/1.1.0-
 rc.1/jquery.mobile-1.1.0-rc.1.min.css" />

 <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
 <script src="http://code.jquery.com/mobile/1.1.0-rc.1/jquery.mobile-
 1.1.0-rc.1.min.js"></script>

</head>
<body >
       <div data-role="page">

            <div data-role="header">
                <h1>Un Titre</h1>
            </div><!-- /header →

            <div data-role="content">
                <p>Bonjour le monde !</p>
            </div><!-- /content -->

       </div><!-- /page -->

</body >
```

2 . Rappels sur jQuery

Avant de se lancer dans le vif du sujet, il peut être intéressant de rappeler quelques éléments clés du Framework jQuery. En effet, comme son nom le laisse suggérer et comme nous l'avons souligné en début d'ouvrage, jQuery Mobile est basé sur la célèbre librairie JavaScript. Nous allons donc dans cette partie refaire le tour des principales fonctions et astuces de la librairie.

2.1 . Sélection d'éléments

La sélection d'éléments en jQuery se fait à l'aide du tag raccourci $(), qui est en fait l'équivalent du sélecteur jQuery(). Il prend en paramètre l'élément recherché.
En voici la syntaxe :

```
<script>
        jQuery('l_element_a_selectionne');
        //Ou en plus simple...
        $('l_element_a_selectionne');
</script>
```

Exemples de sélections intéressantes:

- Sélectionne tous les éléments du DOM
  - **$("*")** ;
- Sélection d'éléments HTML
  - Paragraphe :
    - **$('p')** ; //Sélectionne tous les paragraphes de la page html
  - Sélection tous les éléments de type hx (h1, h2, h3, h4, ...)
    - **$(":header")** ;
  - Liste à puces :
    - **$('li')** ; //Sélectionne tous les éléments de type liste.

- Sélection d'une classe et d'un id
  - **$('.demo')** ; //La classe s'appelle 'demo'
  - **$('#contenu')** ; //L'id s'appelle 'contenu'
- Sélectionne tous les éléments comportant le mot 'mobinaute'
  - **$(":contains('mobinaute')")** ;
- Renvoi tous les blocs d'éléments comportant des images
  - **$("a:has(img)")** ;
- Sélectionne tous les éléments de type submit
  - **$(input[type= "submit"])** ou **$(:submit)**;
- Sélectionne tous les éléments qui n'en contiennent pas d'autres. Ici on sélectionne les cases vides d 'un tableau
  - **$("td:empty")** ;

Une fois ces éléments sélectionnés on peut leur appliquer un certain nombre de changements gérés dynamiquement par jQuery.
Ainsi si nous souhaitons ajouter une classe 'rouge' à un élément ayant pour id le mot ***header*** et une classe 'noir' à un élément ayant pour id le mot ***footer,*** il suffira d'écrire :

```
$(document).ready( function() {
        $('#header').addClass('rouge');
        $('#footer').addClass('noir');
});
```

La fonction **$()** va donc nous permettre de sélectionner tout (ou presque tout) ce que l'on souhaite.

2.2 . Manipulation du DOM[1] (Document Object Model)

La manipulation du DOM est l'étape qui suit généralement la sélection. L'idée est donc de récupérer un éléments du DOM, puis de lui affecter un certain nombre de propriétés. Il existe de nombreuses fonctions jQuery facilitant la manipulation du DOM. Mais nous nous intéresserons qu'à un certain nombre d'entre eux, que nous pensons être les plus importants quand il s'agit de développer avec jQuery Mobile.

2.2.1 La fonction text()

Supposons que nous ayons le code html suivant :

```
<html>
<body>
      <div class="container">
            <h1 id="titre">Demo1 jQuery - Manipulation du DOM</h1>
            <p id="contenu">
              Hello <b>Mobinaute</b> <br />
              Je suis un contenu qui va s'afficher sous forme d'alerte. <br />
              Je suis un exemple pour montrer à quel point il est facile
              d'utiliser la fonction <b>text()</b>.
            </p>
      </div>
</body >
</html >
```

Il est très facile de récupérer le contenu du paragraphe ayant l'id **'contenu'** grâce à la fonction **text()** :

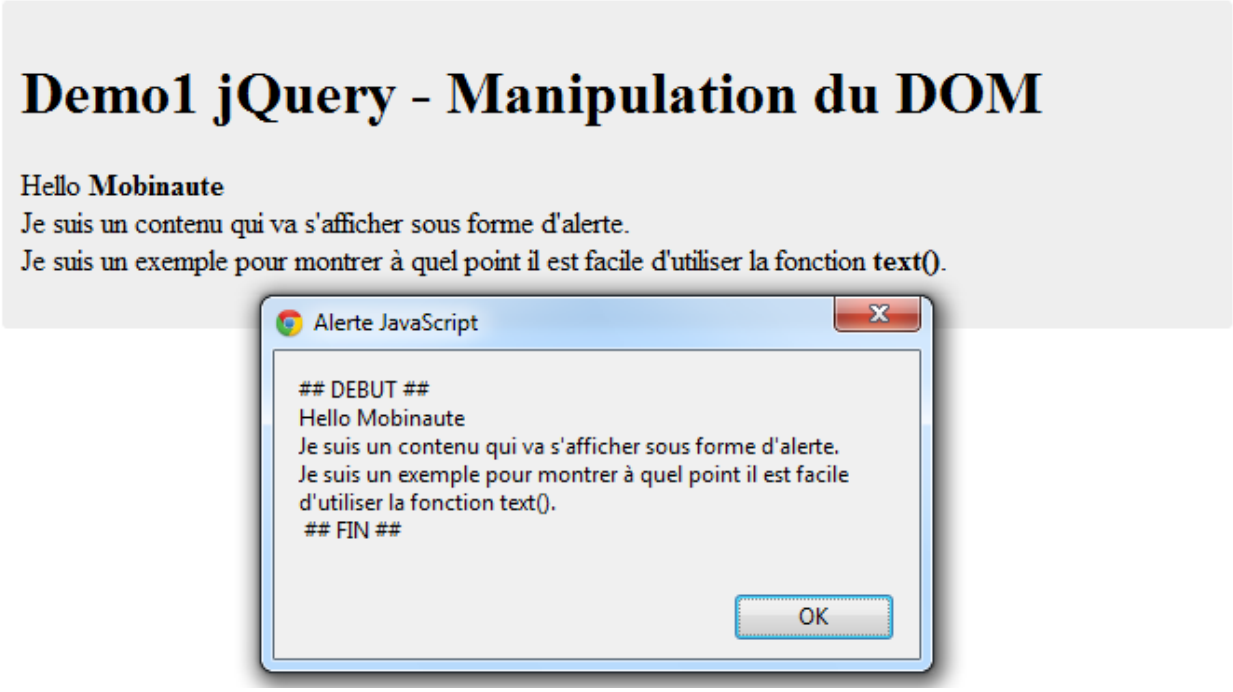
```
var contenu = $("#contenu").text();
```

La variable **contenu** contient le texte du paragraphe.

[1] Le **DOM** (Document Object Model) est une interface de programmation (API) qui décrit la structure d'un document et surtout la manière d'y accéder et de le manipuler. Plus d'informations sur Wikipédia (http://fr.wikipedia.org/wiki/Document_Object_Model)

Dans cet exemple, nous affichons le contenu du paragraphe dans une alerte

```
<script>
	var contenu = $('#contenu').text(); // Juste du texte
	alert('## DEBUT ## ' + contenu + ' ## FIN ##');// Affiche une alerte
</script>
```



Voir la Démo 1 (Dossier demos/chapitre2)

Si l'on souhaite à présent changer le contenu du paragraphe, il suffit de faire :

```
$("#contenu").text("Bonjour le Monde");
```

Qui a pour effet de remplacer le contenu actuel du paragraphe par « Bonjour le Monde » .

De façon générale, La fonction text() sans argument permet de récupérer le contenu d'un élément du DOM, tandis qu'avec un argument de type chaîne de caractère, celle-ci va remplacer le contenu courant par celui indiqué en paramètre.

Le texte du paragraphe a été remplacé par « Bonjour le Monde »

Voir la Démo 1 bis (Dossier demos/chapitre2)

2.2.2 La fonction html()

La fonction html() permet de récupérer tout le contenu d'un élément (aussi bien du texte que des balises html).

Pour la voir en action, modifions légèrement notre code html:

```
<html>
<body>
            <div class="container">
                  <h1 id="titre">Demo2 jQuery - Manipulation du DOM</h1>
                  <div id="contenu1">
                        <p>Voici du beau contenu html qui sera entièrement
récupéré.</p>
                        <ul>
                              <li>1 - Un</li>
                              <li>2 - Deux</li>
                        </ul>
                  </div><!--- Contenu html qui sera récupéré-->

            </div><!--- Container -->
      </body >
</html >
```

Pour récupérer le contenu HTML, il suffit de faire :

```
var contenuHTML = $("#contenu2").html();
```

La variable ***contenuHTML*** contient le code html qui se trouve à l'intérieur de la div (id='contenu').

Si l'on souhaite par contre modifier le contenu du bloc en y insérant autre chose , il suffit de faire :

```
$('#secondaire').html(autre_contenuHTML);
```

Insertion d'un nouveau contenu html

```
<script>
      var autre_contenuHTML = "<p>Voici un nouveau contenu qui montrer
            la facilité d'utilisation de la fonction <b>html()</b><p>";
      var contenuHTML = $('#contenu').html(autre_contenuHTML);
      alert('## DEBUT ## ' + contenuHTML + ' ## FIN ## ');
</script>
```

Voir les Démos 2 et 2 bis (Dossier demos/chapitre2)

2.2.3 La fonction remove

Supprime tous les éléments du DOM répondant aux critères de sélection. Attention, cette fonctionne ne supprime PAS les éléments de l'objet jQuery, ce qui permet une utilisation de ces éléments même si ceux ci ne figurent plus dans le document.

Considérons le contenu html suivant :

```
<div class="container">
    <div class="hello">Hello</div>
    <div class="world">World</div>
</div>
```

Nous choisissons de supprimer le mot 'World' du contenu. Il suffit de faire :

```
$('.hello').remove();
```

Qui aura pour résultat la suppression du bloc contenant la classe 'world'. On aura finalement :

```
<div class="container">
    <div class="world">World</div>
</div>
```

Si plusieurs blocs possèdent la classe sélectionnée, ils seront également supprimer.
Cette fonction est particulièrement intéressante dans le cas où l'on souhaiterait supprimer du contenu, sans affecter la hiérarchie du DOM

Autre exemple : Suppression de tous les paragraphes du DOM

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
 <style>p { background:blue; }</style>
</head>
<body>
  <p>Hello mobinaute ! </p>
  <p>Comment vas-tu</p>
  Ce texte ne sera PAS supprimé.
  <p>aujourd'hui?</p>
  <button>Supprimer les paragraphes</button>
<script>
        $("button").click(function () {
                $("p").remove();
        });
</script>
</body>
</html>
```

On peut aussi choisir de ne supprimer que des paragraphes possédant une certaine classe.

```
<!DOCTYPE html>
<html>
<head>
        <style>p { background:blue; }</style>
        <script src="http://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
        <p class="hello">Hello mobinaute ! </p>
        <p>Comment vas-tu</p>
        Ce texte ne sera PAS supprimé.
        <p>aujourd'hui?</p>
        <button>Supprimer les paragraphes</button>
<script>
        $("button").click(function () {
                $("p").remove(":contains('Hello')");
        });
</script>
</body>
</html>
```

2.3 . Gestion des événements

Les événements font partis intégrante de jQuery. Ils permettent de gérer les actions utilisateurs, comme le clic sur un bouton. Voici 2 d'événements très utilisés :

- **document.ready()** permet de déclencher l'exécution du code, ceci remplace la fonction onload() de JavaScript.

- **$('#events').click(function() {**

  });

L'exemple ci-dessus permet de simuler le clic d'un utilisateur.

D'autres événements se fondent sur le même principe : **blur, bind, change, mouseon, mouseup** …

Dans l'exemple précédents, nous avons sans le signaler utilisé des événements. Il en existe un certain nombre, plusieurs dizaines pris en charges par jQuery, mais les plus utilisés sont très certainement **bind, click**, **toggle, change, submit , ready** et **load**.

Un événement est presque toujours associé à une sélection, et donc affecté à un objet du DOM.

2.3.1 L'événement bind

Permet de lier une action à un événement particulier. Le gestionnaire d'événements peut être passé en argument de la fonction à appeler . Pour arrêter l'action par défaut et le bouillonnement d'événement, votre fonction doit retourner "false". Le bouillonnement d'événement est le phénomène qui fait remonter un événement chez tous les parents de l'objet touché, ce qui peut dans certains cas, considérablement diminuer les performances du système.

Exemple 1 : Annule une action par défaut, ici la soumission d'un formulaire.

```
$("form").bind("submit", function() { return false; })
```

Exemple 2 : Le contenu du paragraphe est affiché au clic dans une alerte.

```
$("p").bind("click", function(){
   alert( $(this).text() );
 });
```

Code de test:

- `<p>Hello</p>`

Résultat:

- `alert("Hello")`

2.3.2 L'événement click

Considérons le code html suivant :

```
<p id="clic">
      <button>Cliquez-moi !</button>
      <span class="no-see">Contenu qui s'affichera au clic...</span>
</p><!--- Contenu secondaire →
```

La class **'no-see '** a un display à **'none'**. Le contenu du span n'est donc pas visible au chargement de la page.

```
<style type="text/css">
.no-see {
      display:none;
}
</style>
```

Nous allons en jQuery afficher ce contenu au clic sur le bouton.

```
<script>
            $('#clic').click( function() {
                  $(this).find('span').removeClass('no-see');
            });
</script>
```

L'élément **this** se réfère à l'objet courant du DOM, et dans notre exemple il s'agit du paragraphe ayant pour id 'clic'. La fonction find() permet de trouver un élément contenu dans le paragraphe (**this**). Puis une fois cet élément trouvé, on supprime la class **'no-see '** avec la fonction removeClass().

Voir la Démos 3 (Dossier demos/chapitre2)

2.3.3 L'événement toggle

Supposons que nous souhaiterions afficher/cacher le contenu d'un paragraphe au clic sur un bouton.

Reprenons notre code source en le modifiant très légèrement :

```
<html >
<body>
      <div class="container">

            <h1 id="titre">Demo2 jQuery – Gestion des événements</h1>
            <div id="contenu">
                  <p>Un contenu qui apparait et disparait, apparait,
disparait,                        apparait, …
                  </p>
                  <ul>
                        <li>jQuery UI</li>
                        <li>jQuery Mobile</li>
                  </ul>
            </div><!--- Contenu-->
            <p id="secondaire">
                  <button>Toggle-me !</button>
            </p><!--- Contenu secondaire →

      </div><!--- Container -->
</body >
</html>
```

L'action se fait au clic sur un bouton. L'événement est définie de façon relativement simple en jQuery :

```
<script>
            $("#secondaire").toggle(
              function () {
                  $('#contenu').addClass("toggle");
              },
              function () {
                  $('#contenu').removeClass("toggle");
              }
            );
</script>
```

La fonction toggle() permet d'alterner entre deux action par clic successifs. Ainsi au premier clic la première fonction est invoquée, tandis qu'au second, c'est la deuxième qui est invoquée. Le cycle recommence au clic.

Les fonctions addClass() et removeClass() permettent respectivement d'ajouter ou de supprimer une classe donnée en paramètre.

Code complet :

```
<html >
<head>
	<title>Démos jQuery</title>
	<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></scrip
t>
	<style type="text/css">
	.container {
		width:40em;
		margin:0px auto;
		padding:10px;
		background:#eee;
		border-radius:3px;
	}
	.toggle {
		display:none;
	}
	</style>
</head>
<body>
	<div class="container">
		<h1 id="titre">Demo2 jQuery – Gestion des événements</h1>
		<div id="contenu">
			<p>Un contenu qui apparait et disparait, apparait,
			   disparait, apparait, …
			</p>
			<ul>
				<li>jQuery UI</li>
				<li>jQuery Mobile</li>
			</ul>
			</div><!--- Contenu-->
```

```
                <p id="secondaire">
                        <button>Toggle-me !</button>
                </p><!--- Contenu secondaire →
        </div><!--- Container →
        <script>
                $("#secondaire").toggle(
                        function () {
                    $('#contenu').addClass("toggle");
                        },
                        function () {
                    $('#contenu').removeClass("toggle");
                  });
        </script>
    </body >
</html >
```

Le contenu disparaît au clic sur le bouton **'Toggle-me'** :

Avant :

Après :

2.3.4 L'événement submit

Intervient lors de l'envoi des éléments d'un formulaire.

Exemple :

Cet exemple permet de vérifier que le champ input contient bien une valeur. Si ce n'est pas le cas, le formulaire ne sera pas envoyé.

```
$("#formulaire").submit( function() {
   return $("input", this).val().length > 0;
 } );
```

Code de test:

```
<form id="formulaire"><input type=''text'' value='' ''/></form>
```

2.4 . Ajouter des effets

Il existe un certain nombre d'effets intéressants en jQuery. Nous intéresserons à ceux qui apparaissent le plus lors de développements avec jQuery Mobile.

2.4.1 hide() et show()

Les fonctions hide() et show() comme le laissent présager leurs noms, permettent d'afficher ou de masquer du contenu. Sans aucun paramètre l'affichage/masquage se fait

2.5 . Autres fonctions intéressantes

2.5.1 $.browser

Contient des drapeaux indiquant le useragent courant (équivalent du navigator.userAgent). Les drapeaux disponibles sont : safari, opera, msie et mozilla.

Exemple:

Affiche une alerte seulement sur les navigateurs de type Safari

```
if ($.browser.safari) {
      $( function() {
            alert("Ceci est un navigateur Safari!");
      } );
}
```

2.5.2 each

La fonction. each () lorsqu'elle est appelée permet d'effectuer une itération sur les éléments du DOM qui font partie de l'objet jQuery.

Supposons que nous ayons la liste non ordonnée suivante:

```
<ul>
    <li> un </ li>
    <li> deux </ li>
</ul>
```

Nous pouvons choisir les éléments de la liste et effectuer une itération sur chacun d'eux:

```
$ ('li'). each (function (index) {
   alert (index + ':'. + $ (this).text());
});
```

Affichera le contenu de chaque élément de liste précédé du texte *'index* +'.

2.5.3 after

Insère du contenu après chaque élément de la sélection.

Considérons le code html suivant :

```
<div class="container">
 <h2>Greetings</h2>
 <div class="inner">Hello</div>
 <div class="inner">Goodbye</div>
</div>
```

Le script :

```
$('.inner').after('<p>Test</p>');
```

générera le contenu suivant :

```
<div class="container">
 <h2>Greetings</h2>
 <div class="inner">Hello</div>
 <p>Test</p>
 <div class="inner">Goodbye</div>
 <p>Test</p>
</div>
```

2.6 . Développer avec Ajax

Ajax est apparu il y a déjà quelques années et est une technologie permettant d'envoyer des requêtes à un serveur et de récupérer ses réponses sans rafraîchir sa page. On l'utilise souvent pour rafraîchir des petits bouts de HTML, dans le but d'éviter des temps de chargement longs et des requêtes vers les serveurs trop lourdes à gérer.

Pour illustrer son fonctionnement, nous allons créer un mini-formulaire de recherche avec un effet similaire à ***Google-instant***.

```
<form class="ajax" action="search.php" method="get">
	<p>
		<label for="query">Rechercher un article</label>
		<input type="text" name="query" id="q" />
	</p>
</form> <!--Formulaire-->

<div id="results"></div><!-- Résultats-->
```

Avec jQuery, l'utilisation d'Ajax se fait simplement :

```
$(document).ready( function() {

 // détection de la saisie dans le champ de recherche
 $('#q').keyup( function(){

   $field = $(this);

   $('#results').html(''); // on vide les resultats

   $('#ajax-loader').remove(); // on retire le loader


   // on commence à traiter à partir du 2ème caractère saisie
   if( $field.val().length > 1 )
   {
    // on envoie la valeur recherché en GET au fichier de traitement
    $.ajax({
     type : 'GET', // envoi des données en GET ou POST
     url : 'ajax-search.php' , // url du fichier de traitement
     data : 'q='+$(this).val() , // données à envoyer en  GET ou POST
     beforeSend : function() { // traitements JS à faire AVANT l'envoi
		$field.after('<img src="ajax-loader.gif" alt="loader" id="ajax-loader" />');
// ajout d'un loader pour signifier l'action
     },
     success : function(data){ // traitements JS à faire APRES le retour d'ajax-search.php
		$('#ajax-loader').remove(); // on enleve le loader
		$('#results').html(data); // affichage des résultats dans le bloc
     }
    });
   }
 });
});
```

2.7 . Travaux pratiques

- **Manipulation de tableaux**

Considérons un tableau contenant une liste de contacts, avec différentes informations par colonne :

- nom, prénom, adresse mail, téléphone bureau, mobile, nom de l'entreprise, plus quelques informations supplémentaires au choix.

La dernière colonne sera obligatoirement un bouton ayant pour valeur le mot **'Supprimer '**. Ce bouton permettra donc de supprimer un contact du tableau. Cependant au lieu de faire disparaître la ligne choisie, sa couleur d'arrière-plan deviendra rouge, et le texte du bouton passera de **'Supprimer'** à **'Annuler'**.

- **Afficher/Masquer**

Utiliser les fonctions addClass()/Remove() pour afficher/masquer le contenu d'un texte.

- **Validation de formulaires**

Utiliser la fonction submit() pour valider un formulaire.

- **Affichage de commentaires sans rafraîchissement de la page (Ajax)**

3 . Utilisation des composants jQuery Mobile

3.1 . Créer une page avec jQuery Mobile

3.1.1 Structure d'une page

Une page jQuery Mobile doit nécessairement déclarer un doctype HTML5, ce afin de bénéficier de tous les avantages qu'apporte la nouvelle version du langage à balises, d'ailleurs très bien implémentés dans le Framework mobile.

Déclaration d'un doctype HTML5 *:*

```
<!DOCTYPE html>
  <html>
```

L'entête html (<head>...</head>) doit ensuite importer les éléments nécessaires au fonctionnement de jQM en suivant cet ordre:

- La feuille de style jQuery Mobile (CSS)
- Le fichier jQuery (JavaScript)
- Le fichier jQuery Mobile (JavaScript)

```
<head>
      <title>Ma première page</title>
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="http://code.jquery.com/mobile/1.1.0-
rc.1/jquery.mobile-1.1.0-rc.1.min.css" />
      <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>

      <script src="http://code.jquery.com/mobile/1.1.0-rc.1/jquery.mobile-1.1.0-
rc.1.min.js"></script>


</head>
```

Remarques :

- **jQuery Mobile** est compatible à la fois avec les versions 1.6.4 et 1.7.1 de jQuery.
- Sur le site du Framework, il est recommandé de faire un lien vers les sources stockées directement sur les serveurs jQuery (code.jquery.com).

Notons au passage la présence de la métadonnée **viewport** qui spécifie le niveau de zoom de la page au sein du navigateur, ainsi que les dimensions de celle-ci. Nous précisons ici que la page s'affichera normalement, sans zoom (valeur à 1), et que sa largeur sera égale à celle du device[2] (width=device-width) depuis lequel la page est consultée Si l'on souhaite afficher la page avec un zoom de 2, il suffit simplement de mettre la valeur du initial-scale à 2.

Il faut noter que ce choix de configuration n'empêche pas l'utilisateur de **zoomer/dézoomer** sur la page qu'il consulte.

3.1.2 Contenu d'une page

Dans jQuery Mobile, chaque page est associée à une vue de l'application. Définie au sein du tag <body>, une page (vue) est identifiée par l'attribut data-role="page" .

Définition d'une page (Vue) jQuery Mobile

```
<div data-role="page">
    ...
</div>
```

[2] Smartphone, tablette, mobile wap,...

Au sein ensuite d'une page, il est possible d'ajouter n'importe quel élément html (lien hypertexte, image, liste à puce,...). Cependant et dans une logique de développement mobile, on utilise toujours strictement la structure suivante :

```
<div data-role="page">
    <div data-role="header">...</div>
    <div data-role="content">...</div>
    <div data-role="footer">...</div>
</div>
```

Une page est ainsi composée :

- D'un **entête** où apparaîtra le titre de la page (Vue) courante
- D'un **contenu**, où l'on affichera les éléments visibles de la page
- D'un **pied de page** où l'on pourra par exemple rajouter un copyright ou même un menu sous forme d'onglets (Navigation tab).

Un exemple complet :

```
<!DOCTYPE html>
<html>
      <head>
          <title>Ma première page</title>

          <meta name="viewport" content="width=device-width, initial-scale=1">
          <link rel="stylesheet" href="http://code.jquery.com/mobile/1.1.0-rc.1/jquery.mobile-1.1.0-rc.1.min.css" />

          <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>

          <script src="http://code.jquery.com/mobile/1.1.0-rc.1/jquery.mobile-1.1.0-rc.1.min.js"></script>

      </head>
```

```
<body>
<div data-role="page">
	<div data-role="header">
		<h1>Ma première vue</h1>
	</div><!-- /header -->

	<div data-role="content">
		<p>Le contenu de la page s'affichera ici.</p>
	</div><!-- /content -->

	<div data-role="footer">
		<h4>&copy; 2012 Mobile-tuts! – Un beau pied de page</h4>
	</div><!-- /footer -->
</div><!-- /page -->
</body>
</html>
```

Voir Démo 1 (dossier seance2)

3.1.3 Structure multi-page

Il est possible de définir plusieurs pages au sein d'un même document. Elles sont toutes définies à l'aide de l'attribut **data-role = "page"** . Chaque bloc de page nécessite d'être identifié à l'aide d'un unique ID, qui sera utilisé pour lier les pages entre elles.

Définition d'un document multi-page

```
<body>
<div data-role="page" id="page1">

	<div data-role="header">
		<h1>Titre - Page 1</h1>
```

```
	</div><!-- /header -->

	<div data-role="content">
		<p>Contenu de la première page.</p>
		<p>Afficher la <a href="#page2">Page n°2</a></p>
	</div><!-- /content -->

	<div data-role="footer">
		<h4>&copy; 2012 Mobile-tuts! – Un beau pied de page</h4>
	</div><!-- /footer -->
</div><!-- /page 1 -->

<div data-role="page" id="page2">

	<div data-role="header">
		<h1>Titre - Page 2</h1>
	</div><!-- /header -->

	<div data-role="content">
		<p>Contenu de la seconde page.</p>
		<p><a href="#page1">Retour à la page 1</a></p>
	</div><!-- /content -->

	<div data-role="footer">
		<h4>&copy; 2012 Mobile-tuts! – Un beau pied de page</h4>
	</div><!-- /footer -->
</div><!-- /page 2 -->
</body>
```

La transition d'une page à l'autre se fait en Ajax, qui se sert des différents ID pour trouver la page recherchée (href="#ID_de_la_page_recherchée") .

Voir Démo 2 (dossier chapitre3)

Remarque

- Il n'est actuellement pas possible de créer un ancre (lien hypertexte interne) au sein d'une même page, car jQuery Mobile utilise le caractère dièse pour la recherche de pages.

- Il est recommandé dans un contexte multi-page de définir le titre de chaque vue (page) à l'aide de l'attribut data-title , plutôt que d'utiliser l'attribut data-role="header" dans un div au sein duquel on définit le titre (h1). On passe ainsi de :

```
<div data-role="page" id="page1">
      <div data-role="header">
                  <h1>Un titre de page</h1>
      </div><!-- /header -->
…
        <div data-role="content">
                <p>Contenu de la seconde page.</p>
                <p><a href="#page1">Retour à la page 1</a></p>
        </div><!-- /content -->

        <div data-role="footer">
                <h4>&copy; 2012 Mobile-tuts!</h4>
        </div><!-- /footer -->
</div>
```

à ceci :

```
<div data-role="page" id="page1" data-title="Un titre de page">
       <div data-role="content">
                <p>Contenu de la seconde page.</p>
                <p><a href="#page1">Retour à la page 1</a></p>
        </div><!-- /content -->

        <div data-role="footer">
                <h4>&copy; 2012 Mobile-tuts!</h4>
        </div><!-- /footer -->

</div>
```

3.1.4 Passage d'une page à l'autre

Le passage d'une page à une autre peut se faire de différentes manières. Dans le cas d'un document multi-page, le passage, comme on l'a vu précédemment, se fait en cliquant sur un ancre possédant l'ID de la page que l'on souhaite atteindre. Ainsi, en cliquant sur le lien ci-dessous, on passera de la page courante vers la page 1 :

```
<p><a href="#page1">Aller à la page 1</a></p>
```

Si l'on souhaite par contre atteindre un contenu qui n'est pas dans le document courant, jQuery Mobile utilise par défaut la technologie Ajax, pour passer d'une vue à une autre. Le contenu est importé et rajouté au DOM.

```
<a href="externe.html" rel="external">Une page externe</a>
```

L'attribut rel="external" permet de préciser qu'il s'agit bien d'une page externe au document courant. Si l'on ne souhaite pas utiliser Ajax pour passer d'un contenu à un autre, il faut mettre l'attribut data-ajax à **false**. Soit avec l'exemple précédent :

```
<a href="externe.html" rel="external" data-ajax="false">Une page externe</a>
```

3.1.5 Utiliser les transitions entre les vues

jQuery Mobile propose plusieurs effets de transitions inspirés des transitions disponible au sein d'applications natives.

Il existe actuellement (jQuery Mobile 1.1) 8 transitions : **fade**, **pop**, **flip**, **turn**, **flow**, **slide**, **slideup** et **slidedown**.

L'effet de transition doit être définit dans le lien qui servira au passage de la vue courante vers une autre :

```
<a href="index.html" data-transition="pop">Effet pop</a>
```

On chargera donc le contenu du fichier index.html avec l'effet **pop**.

3.1.6 Les boites de dialogues

Les transitions citées précédemment sont également utilisées dans la définition d'une boite de dialogue. L'effet **"pop"** est utilisé par défaut.

```
<a href="index.html" data-rel="dialog">Effet pop</a>
```

On peut aussi définir son effet :

```
<a href="index.html" data-rel="dialog" data-transition="flip">Flip</a>
```

Voir démo 3 (chapitre 3).

3.1.7 Quelques astuces pour les pages made in Apple

Apple a définie un certain nombre de métadonnées qui permettent d'enrichir le développement d'applications web mobile. Ces spécificités n'agissent bien sûr que sur des devices sous iOS (iPod, iPhone et iPad).

Définir l'icône d'une application

```
<link rel="apple-touch-icon" href="apple-touch-icon.png">
<link rel="apple-touch-icon" sizes="72x72" href="apple-touch-icon-72x72.png">
<link rel="apple-touch-icon" sizes="114x114" href="apple-touch-icon-114x114.png">
```

Cette icône s'affichera à l'enregistrement d'une page en favoris sur votre iPhone ou iPad. Cette option est également reconnu sur Android (Voir l'illustration ci-dessous)

Illustration 1: Icône à l'enregistrement d'un favoris

Définir un splash screen

```
<!-- iOS 3+: dimensions 320x460 -->
<link rel="apple-touch-startup-image" href="startup.png">
```

Le splash screen est l'image qui s'affiche en grand écran au démarrage d'une application.

Affichage en mode plein écran

```
<meta name="apple-mobile-web-app-capable" content="yes" />
```

Changer la couleur de la barre d'état en mode plein écran

```
<!-- La barre d'état aura un fond noir -->
<meta name="apple-mobile-web-app-status-bar-style" content="black" />
```

3.2 . Créer des listes

3.2.1 Définir une liste simple

Une liste se définit à l'aide de l'attribut data-role="listview" :

```
<ul data-role="listview" data-theme="g">
        <li data-role="list-divider">Catégorie 1</li>
        <li><a href="lien1.html">Mon lien 1.1</a></li>
        <li><a href="lien2.html">Mon lien 1.2</a></li>
        <li><a href="lien3.html">Mon lien 1.3</a></li>

        <li data-role="list-divider">Catégorie 2</li>
        <li><a href="lien1.html">Mon lien 2.1</a></li>
        <li><a href="lien2.html">Mon lien 2.2</a></li>
        <li><a href="lien3.html">Mon lien 2.3</a></li>
</ul>
```

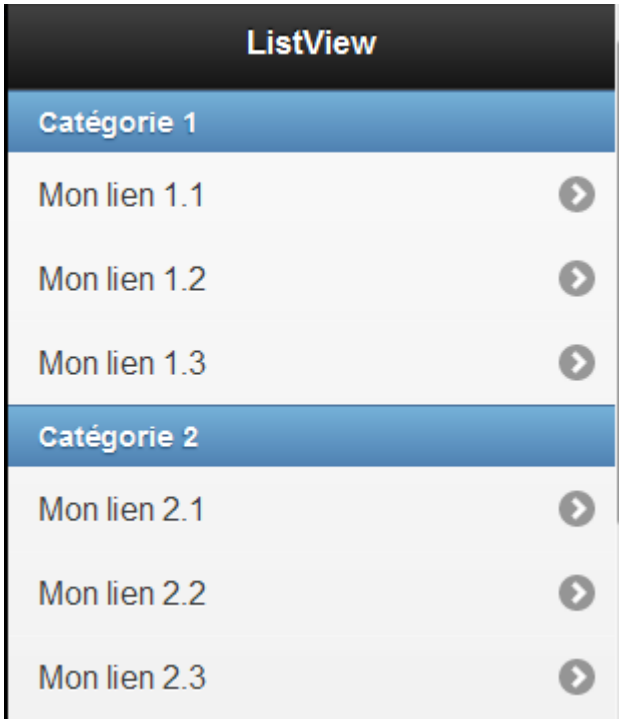
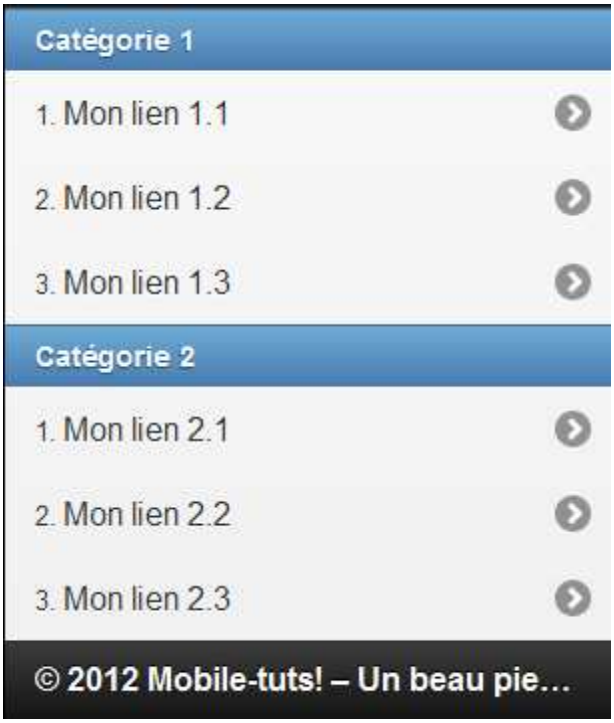


Illustration 2: ListView Simple

3.2.2 Liste numérotée

Une liste numérotée est définit à l'aide de l'élément ol (liste ordonnée)

```
<ol data-role="listview" data-theme="g">
        <li><a href="lien1.html">Mon lien 1.1</a></li>
        <li><a href="lien2.html">Mon lien 1.2</a></li>
        <li><a href="lien3.html">Mon lien 1.3</a></li>

           ...
</ol>
```



3.2.3 Insérer des séparateurs dans les listes

Pour séparer plusieurs listview, il suffit de rajouter l'élément suivant en tête de liste :

```
<li data-role="list-divider">Titre liste 1</li>
```

Soit avec un exemple complet :

```
<div data-role="content">
          <ul data-role="listview" data-theme="g">
               <li data-role="list-divider">Titre liste 1</li>
               <li><a href="lien1.html">Mon lien 1</a></li>
               <li><a href="lien2.html">Mon lien 2</a></li>
               <li><a href="lien3.html">Mon lien 3</a></li>
          </ul>
          <ol data-role="listview" data-theme="g">
               <li data-role="list-divider">Titre liste 2</li>
               <li><a href="lien1.html">Mon lien 1</a></li>
               <li><a href="lien2.html">Mon lien 2</a></li>
               <li><a href="lien3.html">Mon lien 3</a></li>
          </ol>
</div><!-- /content -->
```

Voir démo 4 (chapitre 3)

3.2.4 Afficher un compteur dans un élément de liste

Pour afficher un compteur dans une liste, il faut rajouter un élément contenant la class spécial ''ui-li-count ''

```
<ol data-role="listview" data-theme="g">
        <li>
           <a href="lien1.html">
                Boite de réception <span class="ui-li-count">0</span>
           </a>
       </li>
        <li>
           <a href="lien2.html">
                Brouillon <span class="ui-li-count">4</span>
           </a>
       </li>
</ol>
```

Voir démo 5 (chapitre 3)

3.2.5 Rechercher dans une liste

jQuery Mobile permet de rechercher dans une liste. Pour ce faire, il faut mettre à true l'attribut data-filter de la Listview :

```
<ul data-role="listview" data-theme="g" data-filter= "true" data-
filter-placeholder="Rechercher...">
        <li>
           <a href="lien.html">Google</a>
        </li>
        <li>
           <a href="lien.html">Microsoft</a>
        </li>
        <li>
           <a href="lien.html">Apple</a>
        </li>
        <li>
           <a href="lien.html">Oracle</a>
        </li>
</ul>
```

Voir démo 6 (chapitre 3)

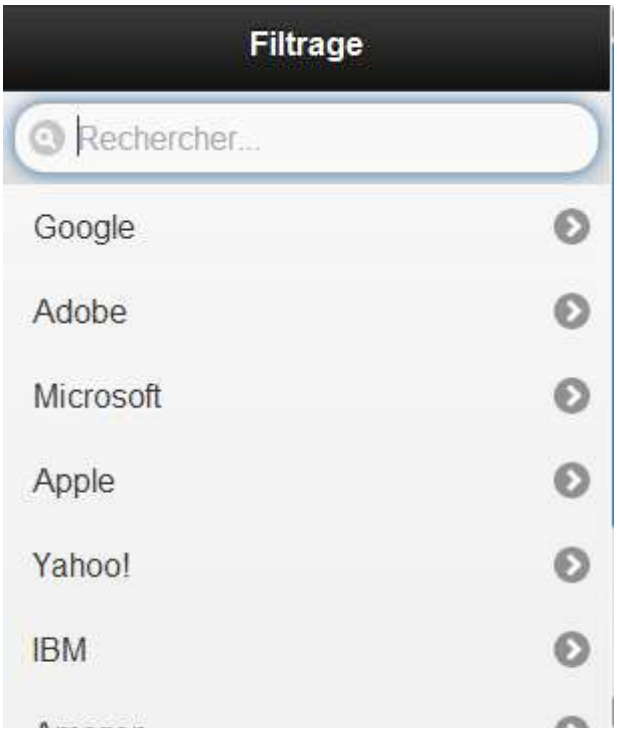


Illustration 3: Avant filtrage

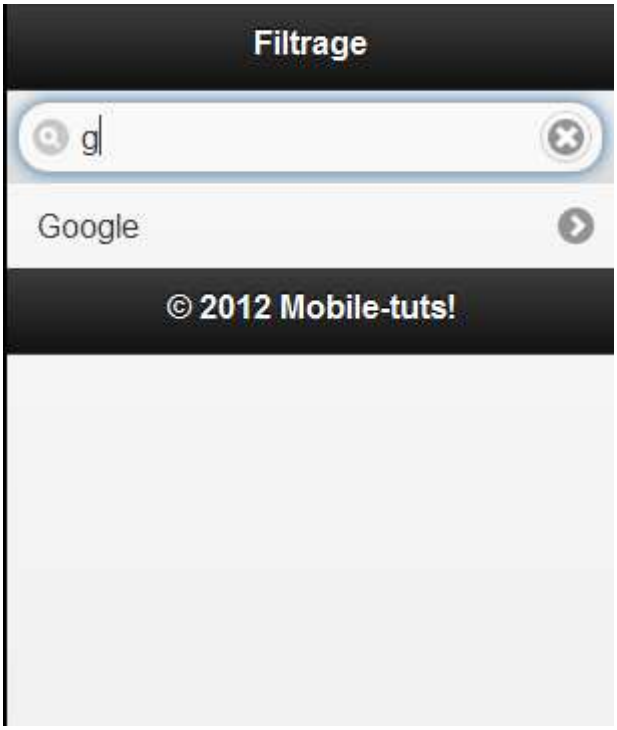


Illustration 4: Après filtrage

3.2.6 Listes à lecture seule

Si une liste possède du contenu non attaché à un lien hypertexte, alors la listview si elle est définie sera en lecture seule.

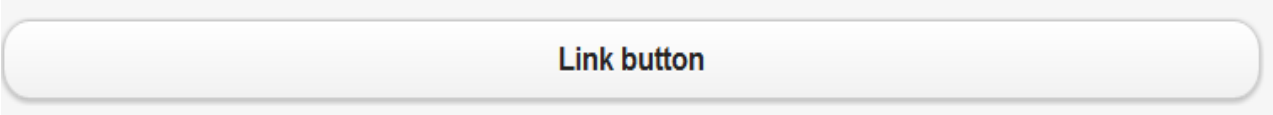
3.3 . Utiliser des boutons

3.3.1 Définir un bouton

Les boutons sont codés avec des liens hypertextes standards. Un bouton est définie grâce à l'attribut data-role=''button'' :

```
<a href="index.html" data-role="button">Link button</a>
```

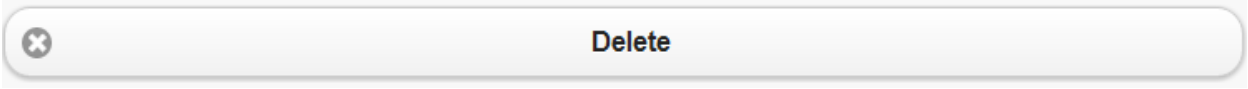
Qui produit le résultat ci-dessous :



3.3.2 Associer une icône à un bouton

JQuery Mobile possède un nombre important d'icônes. Ces derniers peuvent être ajouté à des boutons en donnant une valeur à l'attribut data-icon.

```
<a href="index.html" data-role="button" data-icon="delete" >
      Bouton qui possède sa propre icône
</a>
```



Vous pouvez voir ici la liste des icônes actuellement disponibles sur : **jquerymobile.com/demos/1.1.0-rc.1/docs/buttons/buttons-icons.html**

Il reste tout fait possible de personnaliser son icône, comme nous le verrons dans un prochain chapitre.

3.3.3 Juxtaposer les boutons horizontalement (inline)

Par défaut, tous les boutons d'une page sont définis pour occuper toute la largeur de l'écran . Pour aligner verticalement des boutons, il suffit de mettre l'attribut data-inline à true, soit :

```
<a href="index.html" data-role="button" data-inline="true">Annuler</a>
<a href="index.html" data-role="button" data-inline="true" data-
theme="b">Sauvegarder</a>
```

3.4 . Éléments de formulaires

jQuery Mobile apporte améliore considérablement le design des éléments de formulaire déjà existants. La plupart d'entre eux comme les champs de saisie, les cases à cocher et les boutons radio, ont vu leur apparence complètement changé.

Depuis peu, jQuery Mobile a introduit un nouvel attribut, **data-mini="true"** , qui permet un rendu beaucoup plus petit de ou des éléments de formulaire.

3.4.1 Boutons radio

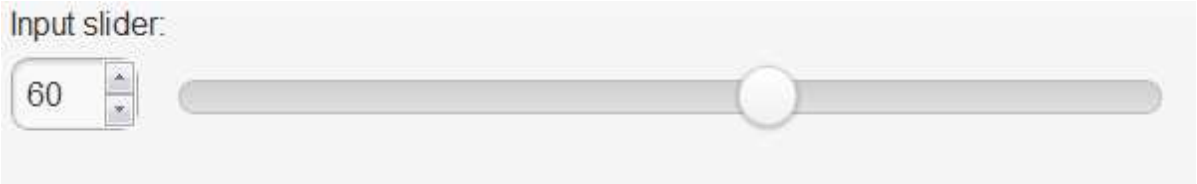
La définition d'un bouton radio reste la même qu'en HTML

```
<fieldset data-role="controlgroup">
      <legend>Choisir un animal domestique :</legend>
            <input  type="radio"  name="radio-choix-1"  id="radio-choix-1"  value="choix-1"
checked="checked" />
      <label for="radio-choix-1">Chat</label>

      <input type="radio" name="radio-choix-1" id="radio-choix-2" value="choix-2"  />
      <label for="radio-choix-2">Chien</label>

      <input type="radio" name="radio-choix-1" id="radio-choix-3" value="choix-3"  />
      <label for="radio-choix-3">Hamster</label>

      <input type="radio" name="radio-choix-1" id="radio-choix-4" value="choix-4"  />
      <label for="radio-choix-4">Souris</label>
</fieldset>
```

3.4.2 Sliders

Pour ajouter un slider à notre page, il suffit simplement d'intégrer un élément input ayant pour type ''range'' :

```
<label for="slider-0">Input slider:</label>
<input type="range" name="slider" id="slider-0" value="60" min="0"
max="100" />
```



3.4.3 Élément de recherche.

Ce élément fait partie des nouveaux éléments HTML, et permet d'effectuer des recherches. Il est définit simplement (classique):

```
<label for="search-basic">Search Input:</label>
<input type="search" name="search" id="searc-basic" value="" />
```

3.4.4 Mieux disposer les éléments à l'écran

Afin d'améliorer le style des éléments du formulaire et leur affichage à l'écran, jQuery Mobile un a créé l'attribut **data-role="fieldcontain"** .

```
<div data-role="fieldcontain">
    <label for="name">Text Input:</label>
    <input type="text" name="name" id="name" value=""  />
</div>
```

3.5 . Les barres d'outils

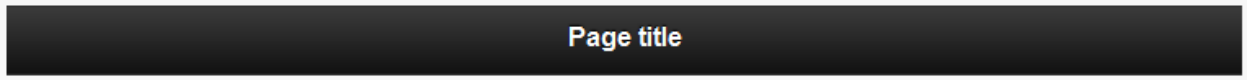
3.5.1 Barres d'outils "header" et "footer"

En jQuery Mobile, il existe deux types standards de barre d'outils :

- La barre l'entête : est utilisée comme le titre d'une vue, elle est généralement le premier élément à l'intérieur d'une page

Une entête est une barre d'outil contenant généralement le titre de la vue courante, ainsi qu'un certain nombre de boutons optionnels et disposés à gauche ou à droite de la barre.

```
<div data-role="header">
      <h1>Page Title</h1>
</div>
```



Il est très facile d'ajouter des boutons à une barre d'outils d'en-tête ou de pied de page :

```
<div data-role="header">
      <a href="index.html" data-icon="delete">Annuler</a>
   <h1>Éditer un contact</h1>
    <a href="index.html" data-icon="check" data-theme="b">Sauvegarder</a>
</div>
```

- La barre de pied de page :qui est généralement le dernier élément d'une page et se construit exactement comme une barre d'outil d'entete :

```
<div data-role="footer" class= ''ui-bar''>
        <div data-role="controlgroup" data-type="horizontal">
        <a href="index.html" data-role="button" data-icon="delete">Supprimer</a>
        <a href="index.html" data-role="button" data-icon="plus">Ajouter</a>
```

```
        <a href="index.html" data-role="button" data-icon="arrow-u">Haut</a>
        <a href="index.html" data-role="button" data-icon="arrow-d">Bas</a>
        </div>
</div>
```

Voir la démo 9 (chapitre 3).

3.5.2 Barres d'outils de type fixe

Il est possible de rendre fixe une barre d 'outil en rajouter simplement l'attribut data-position="fixed" .

3.5.3 Utiliser les barres de navigation

jQuery dispose d'un widget qui gère la navigation par onglet, qui se définit là aussi très facilement

```
<div data-role="navbar">
    <ul>
        <li><a href="a.html" class="ui-btn-active">Un</a></li>
        <li><a href="b.html">Deux</a></li>
    </ul>
</div><!-- /navbar -->
```

On peut choisir d'insérer la navigation par onglet dans notre pied de page :

```
<div data-role="footer">
    <div data-role="navbar">
        <ul>
            <li><a href="#">Un</a></li>
            <li><a href="#">Deux</a></li>
            <li><a href="#">Trois</a></li>
        </ul>
    </div><!-- /navbar -->
</div><!-- /footer -->
```

3.6 . Travaux pratiques

- **Premiers pas avec jQuery Mobile**
  - Installer un environnement de travail : importation des librairies javascript, documents CSS, ... Faire valider votre installation.

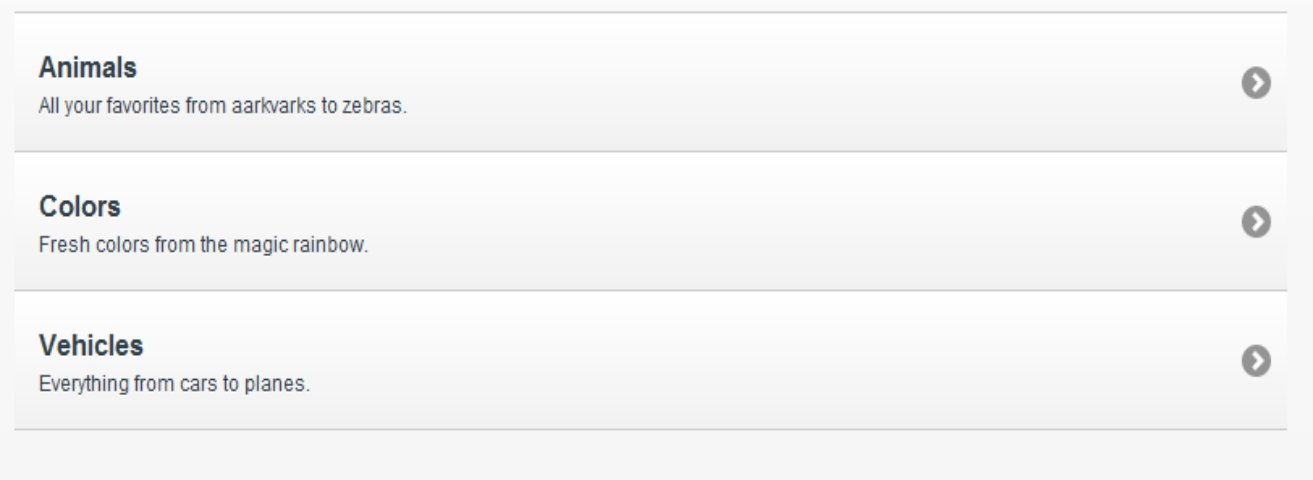
- **Création de pages**
  - Créer une vue comportant un titre, un contenu et un pied de page
  - Ajouter quatre vues différentes à votre application, de façon à ce que l'on puisse naviguer de la page 1 (#page1) vers la page 2(#page2), de la page 2 vers la page 3(#page3), de la page 3 vers page 4(#page4) et de la page 4 vers la page 1.
    - **page 1 ↔ page 2 ↔ page 3 ↔ page 4**
  - Remplacer l'ID de la page 2 par **"#page22"**. Ne changer rien d'autre.
    Depuis la page 1 cliquez sur le lien menant à la page 2. Que se passe t-il ?

- **Embellir son application**
  - Définir des icônes pour différents formats d'application.
  - Ajouter un splashpage (iOS) à votre application.

- **Utiliser des formulaires**
  - Créer une page de connexion : Login, mot de passe. Le formulaire aura un id=''formulaire''.
  - Valider le formulaire en ajoutant le script jQuery suivant juste avant la balise </body> :

```
<script>
  $("#formulaire").submit( function() {
   if ( $("input", this).val().length > 0 ) {
         $('#formulaire').after('<p>Erreur </p>').css('background','red');
               return false ;// Le formulaire n'est pas envoyé.
         }
  } );
<script>
```

- **Utiliser des listes**
    - Voir dans la documentation comment créer les deux types de listes ci-dessous



Et :

Indice : faites afficher le source.

- **Utiliser des boutons**

Trouver dans la documentation officiel commet faire pour afficher les boutons de cette façon.
Indice : faites afficher le source.

- **Barre de navigation**
  - Comment insérer des icônes dans une barre de navigation (d'en-tête ou de pied de page?)
  - Rechercher dans la documentation comment changer le thème d'une barre navigation.

4 . Programmer avec l'API jQuery Mobile

4.1 . Initialisation du Framework

Comme la plupart des outils JavaScript existants, jQuery Mobile exécute un certain nombre d'actions aussitôt qu'il démarre, et même bien avant l'événement **document.ready**. Ces actions sont exécutées en fonction de ce qui est écrit dans la configuration de base de jQM. Nous verrons plus tard comment customiser ces actions et les adapter à nos besoins.

4.1.1 L'événement mobileinit.

Cet événement est exécuté au démarrage immédiat de votre l'application. C'est donc l'endroit rêvé pour appliquer sa propre configuration par défaut.

"Enregistrement de l'événement mobileinit"

```
$(document).bind("mobileinit", function(){
     //Mettre ici des actions à exécuter au démarrage
});
```

Cet événement doit être enregistré (bind) bien avant le chargement de jQuery Mobile. Pour ce faire, il est impératif d'importer les fichiers JavaScript dans cet ordre là :

```
<script src="jquery.js"></script>
<script src="mon-script-perso.js"></script>
<script src="jquery-mobile.js"></script>
```

Et c'est dans le fichier **"mon-script-perso.js"** que l'on écrira l'enregistrement JavaScript de l'événement.
A partir de là, on pourra personnaliser notre environnement d'exécution en ajoutant nos propres paramètres par défaut.

4.1.2 Options de configuration

Comme on l'a précisé plus haut, les options de configuration sont à préciser lors de l'enregistrement de l'événement ***mobileinit*** **.** L'assignation d'une option se fait à l'aide de l'objet $.mobile :

```
$(document).bind("mobileinit", function(){
     $.mobile.une_option = valeur;
});
```

une_option : l'option que l'on souhaite modifier. Ces options sont proposées par le framework (Voir tableau plus bas)

valeur : ce que l'on souhaite ajouter à l'option

Supposons que l'on souhaite désactiver Ajax. jQuery Mobile met à disposition l'attribut **ajaxEnabled.** Il suffit donc, pour le désactiver, de faire:

```
$(document).bind("mobileinit", function(){
     $.mobile.ajaxEnabled = false;//Désactive Ajax dans l'application
});
```

Il existe plusieurs options de configuration. Nous en listons quelques unes dans ce tableau :

Nom de l'option	Type	Valeur par défaut	Description
activeBtnClass	String	"ui-btn-active"	La classe CSS utilisée lorsqu'un bouton est dans son état "actif"
activePageClass	String	"ui-page-active"	La classe CSS assignée à la page courante
ajaxEnabled	Boolée n	true	Gère l'activation/désactivation d'Ajax
autoInitializePage	Boolée n	true	Initialisation automatique ou non d'une page.
defaultDialogTransition	String	'pop'	Transition par défaut d'une boite de dialogue.
defaultPageTransition	String	'slide'	Transition par défaut d'une page
loadingMessage	String	false	Texte affiché lors d'un chargement.
loadingMessageTheme	String	"a"	Thème (style) du message

Une liste complète est disponible à l'adresse suivante :
http://jquerymobile.com/demos/1.1.0-rc.1/docs/api/globalconfig.html

4.1.3 Événements jQuery Mobile

jQuery Mobile propose la gestion de plusieurs événements liés au clic ou au toucher. On pourra donc implémenter des actions qui seront associées à des événements déclenchés depuis un téléphone mobile.

La déclaration d'un événement se fait à l'aide de la fonction JavaScript **bind()** ou **live()**, au choix.

Remarque :

- *Il faut utiliser la fonction **pageInit()** propre à jQuery Mobile et qui permet d'exécuter du code avant le chargement d'une quelconque page (vue), au lieu de **$(document).ready**.*

Nous listons dans le tableau ci-dessous quelques-uns des événements par type :

Type d'événement	Événement	Description
Toucher	**tap**	Événement au clic rapide (tap)
Toucher	**taphold**	Toucher à l'écran d'environ 1s
Toucher	**swipe**	Drag horizontal sur 30px ou plus et moins de 20px en verticale.
Toucher	**swipeleft**	Swipe à gauche
Toucher	**swiperight**	Swipe à droite
Souris (virtuelle)	**vmouseover**	Evenement au toucher.
Souris (virtuelle)	**vmousedown**	Evenement au début du toucher
Souris (virtuelle)	**vmousemove**	Événement au mouvement du doigt
Souris (virtuelle)	**vmouseup**	Événement à la fin du toucher
Orientation	**orientationchange**	Événement lors d'un changement d'orientation
Scroll	**scrollstart**	Événement au démarrage du scrolling
Scroll	**scrollstop**	Événement à la fin du scrolling
Chargement de la page	**pagebeforeload**	Événement avant le chargement de la page

Type d'événement	Événement	Description
Chargement de la page	**pageloadfailed**	Événement au chargement non réussi d'une page
Transition entre page	**pagebeforeshow**	Événement avant l'affichage de la page de destination

Une liste complète est disponible à l'adresse suivante :
http://jquerymobile.com/demos/1.1.0-rc.1/docs/api/events.html

Exemples d'utilisation d'événements jQuery Mobile

```
// pageshow : le paramètre data est un objet du DOM. On pourra donc le manipuler
$( 'div' ).live( 'pageshow',function(event, data){
     console.log( 'Le contenu de  "'+
     data.prevPage.find('h1').text() + '" vient juste de
     disparaitre');
});

// pagehide : le paramètre data est également un objet du DOM.
$( 'div' ).live( 'pagehide',function(event, data){
     console.log( 'Le contenu de  "'+
     data.nextPage.find('h1').text() + '" vient juste d\'apparaitre');
});
```

4.1.4 Fonctions utilitaires

jQuery Mobile propose un certain nombre de fonctions et de propriétés dans l'objet **$.mobile**, que nous pourrons utiliser dans nos applications.

Nom de la fonction	Description	Arguments
$.mobile.changePage	permet de passer d'une page à une autre de façon dynamique et programmatique	destination, options
$.mobile.loadPage	Charge un page externe au document courant	url, options
$.mobile.showPageLoadingMsg	Permet l'affichage d'un texte au chargement d'une page	Theme (string), msgText (string), textonly(boolean)
$.mobile.path.parseUrl	Permet de parser une URL	url relative ou absolue.
$.mobile.path.makePathAbsolute	Converti une url relative en absolue	url
$.mobile.path.isSameDomain	Compare les domaines de 2 URLs	url1, url2
$.mobile.silentScroll	Permet de scroller à endroit de la page, sans déclencher un événement de type scroll	

Exemples d'utilisation des fonctions

```
//Passage à la page "apropos" avec un slideup comme transition
$.mobile.changePage( "about/apropos.html", { transition: "slideup"} );

//Passage à la page "resultats" , en utilisant les données d'un formulaire
$.mobile.changePage( "resultats.php", {
	type: "post",
	data: $("form#search").serialize()
});

//Active l'affichage d'un texte au chargement
$.mobile.showPageLoadingMsg();

//On utilise le thème "b", un message personnalisé, et que du texte
$.mobile.showPageLoadingMsg("b", "Chargement en cours ...", true);
```

4.2 . Manipuler des pages

4.2.1 Gérer les attributs d'une page

jQuery Mobile met actuellement à disposition 12 attributs pour manipuler des pages sans avoir à écrire du code JavaScript. Le tableau ci-dessous les recense tous :

Nom de l'attribut	Description	Valeurs possibles
data-add-back-btn	Permet d'ajouter un bouton "retour". Disponible uniquement dans l'**header** de la page.	true, false
data-back-btn-text	Le texte du bouton retour arrière	String
data-back-btn-theme	Thème du bouton retour arrière	a-z
data-close-btn-text	Texte associé à l'icône de fermeture d'une boite de dialogue.	String
data-dom-cache	Gestion du cache du DOM	true, false
data-fullscreen	Affichage en mode plein écran. A combiner avec les barres de navigation fixes.	
data-overlay-theme		
data-theme	Thème de la page	a-z
data-title	Titre de la page	String

Voir démo 3 (chapitre 4)

4.2.2 Pré-chargement et gestion du cache

Supposez que vous ayez une page contenant un lien hypertexte pointant vers une autre page assez lourd. Avec jQM il sera possible d'effectuer le pré-chargement de

cette page, de façon à ce qu'au clic sur le lien, le chargement se fasse presque instantanément. On utilise l'attribut **data-prefetch**.

Exemple de pré-chargement d'une page

```
<a href="lourde_page.html" data-prefetch> ... </a>
```

Il est également possible de stocker les pages de son application directement dans le DOM. On peut effectuer le stockage de façon programmatique (1) ou alors depuis le html (2) :

(1) Stock l'ensemble des pages déjà visitées

```
$.mobile.page.prototype.options.domCache = true;
```

(2) Ne stock que cette page et toutes celles qui auront l'attribut à **true**

```
<div data-role="page" id="cacheMe" data-dom-cache="true">
```

Les pages stockées sont celles qui ont été visitées avant la page courante.

4.2.3 Gérer ses pages de façon programmatique

- Changement de page

```
// Transite vers la page articles-jqm
$.mobile.changePage( "blog/articles-jqm.html", { transition: "slideup"} );
```

- Chargement d'une page

```
// Rajoute la page au DOM .
$.mobile.loadPage( "blog/articles-android.html" );
```

- Scroller à une position précise dans une page

```
// On scroll à la coordonnée Y = 300px
$.mobile.silentScroll(300);
```

4.3 . Manipuler des listes

4.3.1 Gérer les attributs d'une page

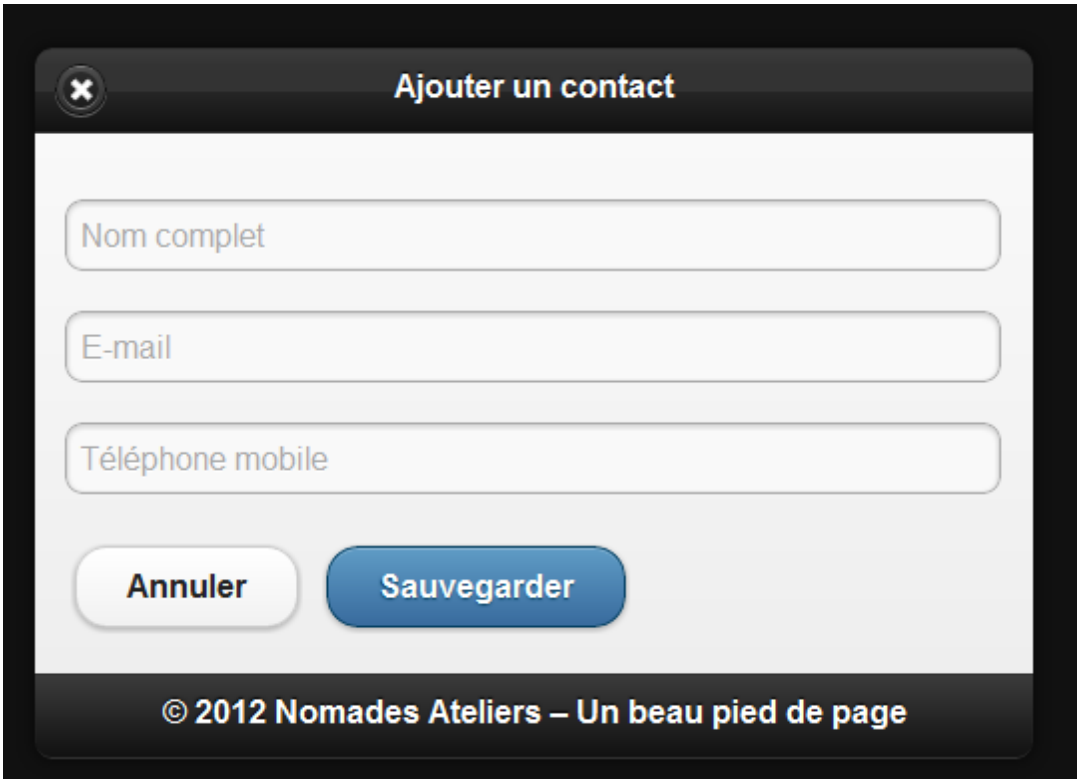
jQuery Mobile propose 9 attributs pour manipuler les listes sans avoir à écrire du code JavaScript. Le tableau ci-dessous les recense tous :

Nom de l'attribut	Description	Valeurs possibles
data-count-theme	Thème du compteur	a-z
data-dividertheme	Thème du séparateur de catégories	a-z
data-filter	Filtrage de données	true, false
data-filter-placeholder	Texte associé au filtre.	String
data-filter-theme	Thème du filtre	a-z
data-inset	Recadrage du listview	true, false
data-split-icon	home | delete | plus | arrow-u | arrow-d | check | gear | grid | star | custom | arrow-r | arrow-l | minus | refresh | forward | back | alert | info | search	**c.f.** description
data-split-theme	Thème de l'icône (split)	a-z
data-theme	Thème de la liste	String

Voir démo 4 (chapitre 4)

4.3.2 Créer une liste dynamiquement

On souhaite créer une application qui puisse nous permettre de créer un contact et de l'ajouter automatiquement à une liste.



Voir démo 4_bis (chapitre 4)

On souhaite qu'à la validation du formulaire, une liste contenant les données du contact soit créée.

4.3.3 Supprimer un élément dans une liste

Considérons la liste suivante :

Code HTML

```
<div data-role="content">
            <ul data-role="listview" data-filter="true" data-filter-
placeholder="Rechercher..." data-inset="true">
                  <li data-role="list-divider">Web & Software</li>
                  <li><a href="#">Javascript</a></li>
                  <li><a href="#">PHP</a></li>
                  <li><a href="#">HTML 5</a></li>
                  <li><a href="#">CSS 3</a></li>
                  <li><a href="#">Linux</a></li>
                  <li><a href="#">Putty</a></li>
                  <li><a href="#">Tor</a></li>

                  <li data-role="list-divider">Hardware</li>
                  <li><a href="#">Smartphone (Android, iOS, Windows
Phone)</a></li>
                  <li><a href="#">Robotique (Arduino)</a></li>
                  <li><a href="#">Serveurs - Box</a></li>
            </ul>
</div><!-- /content -->
```

On souhaiterait qu'au clic sur un élément de la liste, celui-ci soit supprimer. On utilise dans ce cas la puissance des sélectionneurs jQuery. L'action se fait très facilement en se rappelant des étapes à suivre lorsque l'on souhaite manipuler un élément avec jQuery :

- **Sélection** de l'élément : $(), find(), has, :contains(), ...
- Associer l'élément sélectionné à un **listener** (un événement)
- Appliquer l'**action** recherchée

Code JavaScript

```
<script>
        // Suppression d'éléments
        $('ul[data-role="listview"]').find('li').click(function() {
         $(this).remove(); //Supprime l'élément courant
        });
</script>
```

Voir démo 5 (chapitre 4)

4.3.4 Gérer les événements sur les listes

De façon générale, la gestion d'événement dans une liste se fait en sélectionnant celle-ci:

```
<script>
      // Sélection et application d'un événement
     $('ul[data-role="listview"]').find('li').mon_evenement(function() {
            // Définir l'action ici
     });
</script>
```

4.4 . Manipuler des boutons

4.4.1 Gérer les attributs d'un bouton

jQuery Mobile propose 8 attributs pour manipuler les boutons sans avoir à écrire du code JavaScript. Le tableau ci-dessous les recense tous :

Nom de l'attribut	Description	Valeurs possibles
data-corners		true, false
data-icon	home | delete | plus | arrow-u | arrow-d | check | gear | grid | star | custom | arrow-r | arrow-l | minus | refresh | forward | back | alert | info | search	c.f. Description
data-iconpos	Filtrage de données	true, false
data-iconshadow	Texte associé au filtre.	String
data-inline	Alignement horizontal	a-z
data-mini	Version compact du bouton	true, false
data-shadow	Ombrage autour du bouton	True false
data-theme	Thème du bouton	a-z

4.4.2 Agir dynamiquement sur un bouton

Considérons un formulaire. Un bouton sert à valider la saisie. On souhaiterait que le bouton ne soit actif qu'une fois l'ensemble des champs remplit.

Pour afficher un bouton dynamiquement, il suffit d'écrire :

```
$('[type='submit']').button();
```

4.4.3 Gérer les événements sur les boutons

De façon générale, la gestion d'événements dans un bouton se fait en sélectionnant celui-ci :

```
<script>
       // Sélection et application d'un événement
      $('a[data-role="button"]').mon_evenement(function() {
                    // Définir l'action ici
      });
</script>
```

4.5 . Manipuler des champs de saisie

4.5.1 Affecter et récupérer la valeur inscrite dans un champ de saisie

```
<script>
      // Sélection et application d'un événement
     $('textarea').mon_evenement(function() {
           var contenuInput =  $(this).text() ;
      //     Ensuite il faut ....
      // ...Traiter les données
        });
</script>
```

4.5.2 Gérer les événements sur les champs de saisie

De façon générale, la gestion d'événements dans un champs de saisie se fait en sélectionnant celui :

```
<script>
      // Sélection et application d'un événement
      $('textarea').mon_evenement(function() {
      // Définir l'action ici
      });
</script>
```

4.6 . Manipuler des cases à cocher

4.6.1 Gérer les événements sur les cases à cocher

De façon générale, la gestion d'événements dans une case à cocher se fait en sélectionnant celle-ci :

```
<script>
       // Sélection et application d'un événement
      $('input[type="checkbox"]').mon_evenement(function() {
       // Définir l'action ici
      });
</script>
```

4.6.2 Fonction de gestion des cases à cocher

- **Enable** *:* Rend actif un bouton radio

  $("input[type='checkbox']").**checkboxradio**('enable');

- **disable** : désactive un bouton radio

  $("input[type='checkbox']").**checkboxradio**('disable');

- **refresh** : Mets à jour le bouton radio.

$("input[type='checkbox']:first").attr("checked",true).checkboxradio("refresh");

4.7 . Manipuler des boutons radio

4.7.1 Gérer les événements sur les boutons radio

De façon générale, la gestion d'événement dans une liste se fait en sélectionnant celui-ci :

```
<script>
    // Sélection et application d'un événement
    $('input[type="radio"]').mon_evenement(function() {
        // Définir l'action ici
    });
</script>
```

4.7.2 Fonction de gestion des boutons radio

- **Enable** : Rend actif un bouton radio

  $("input[type='radio']").**checkboxradio**('enable');

- **disable** : désactive un bouton radio

  $("input[type='radio']").**checkboxradio**('disable');

- **refresh** : Mets à jour le bouton radio.

4.8 . Manipuler des sliders

4.8.1 Créer dynamiquement un slider

De façon générale, la gestion d'événement dans un slider se fait en sélectionnant celui-ci :

```
<script>
        // Sélection et application d'un événement
            $('input[type="range"]').mon_evenement(function() {
        // Définir les actions ...
        });
 </script>
```

Il est possible d'appeler dynamiquement le plugin Slider au sein de sa page. Cela se fait en une seule ligne :

```
$('input').slider();
```

4.9 . Travaux pratiques

1. **JQMotion** From Scratch

Le but de l'exercice est de développer de A-Z l'application jQMotion.

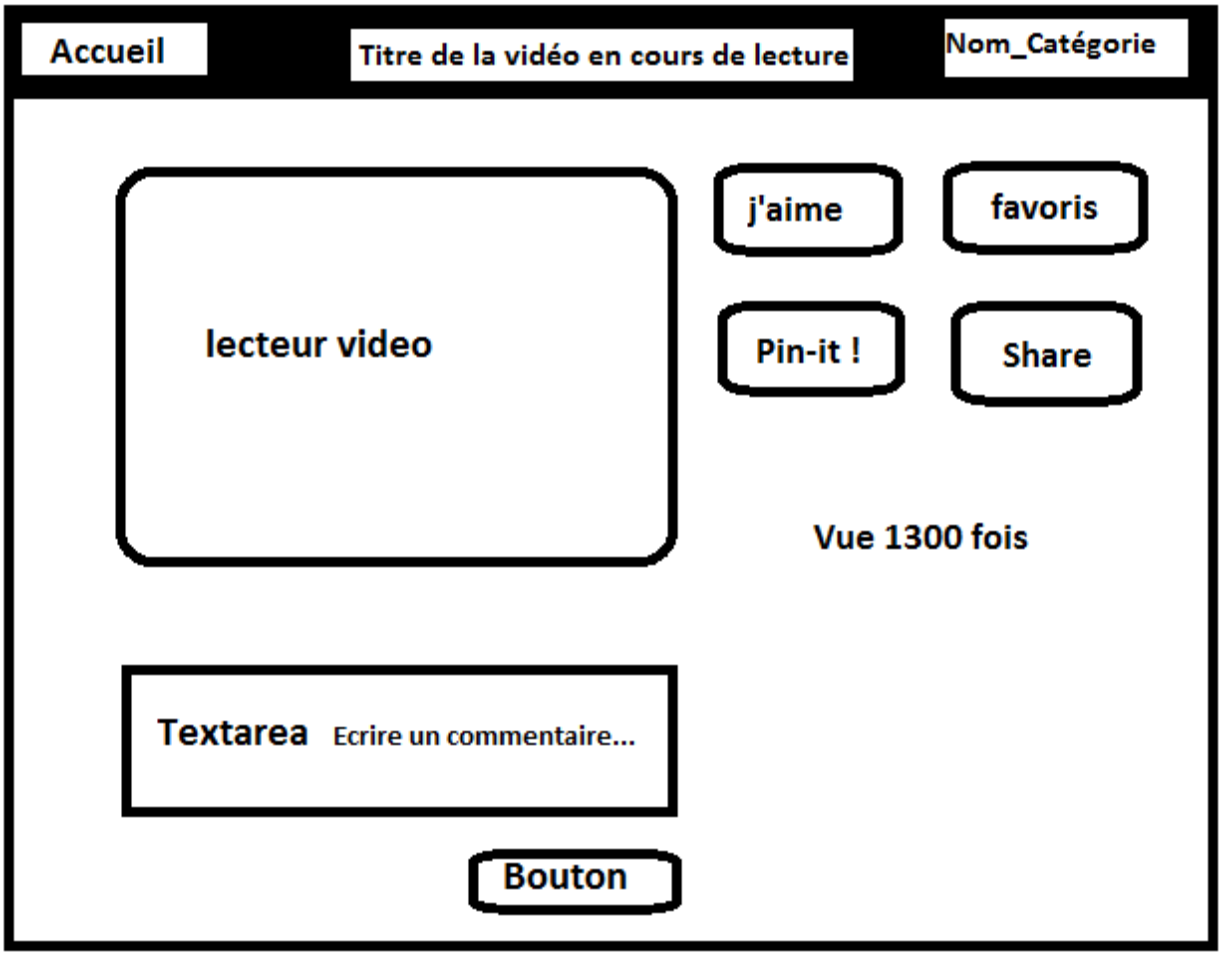
JQMotion est un site mobile qui permet à des utilisateurs de visionner des vidéos poster par des administrateurs. On n'implémentera pas la publication des vidéos. Voici cependant les fonctions qui devront impérativement apparaître dans l'application :

- **F1 - Page de connexion** (On supposera avoir déjà créé son compte). Valider le formulaire en vérifiant que tous les champs sont remplis. Les champs sont :
    - **Nom d'utilisateur :** Chaîne de caractère (String)
    - **Mot de passe :** Chaîne de caractère

Une fois le formulaire validé, on devra, en cliquant sur le bouton de connexion, arriver à la page d'accueil de l'application mobile.

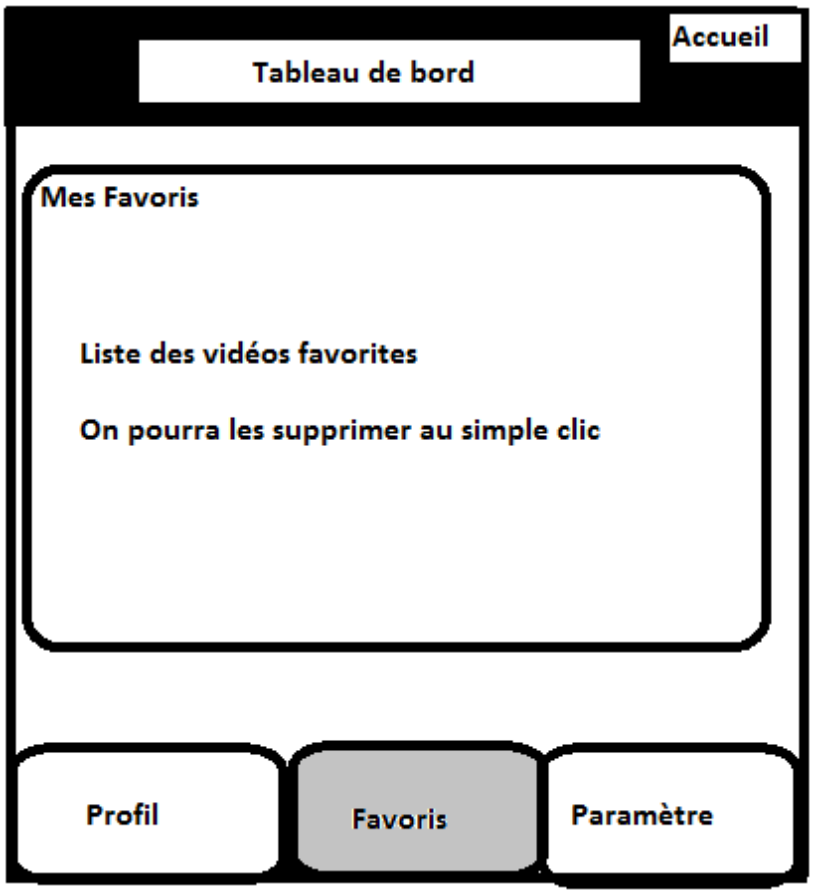
- **F2 – Affichage des vidéos récentes** sous forme de liste. On devra pouvoir distingué les différentes catégories dans la liste. Le nombre et les noms des catégories sont au choix, tout comme le nombre de vidéo par liste.

En cliquant sur une vidéo de la liste, on doit pouvoir se retrouver sur une page comportant la vidéo (Voir figure ci-dessous).



Un clic sur le bouton "Catégorie" doit nous permettre d'afficher toutes les vidéos de la catégorie courante.

- **F3 - Affichage du tableau de bord de l'utilisateur**. Les données seront évidemment brutes. De là il pourra naviguer dans son profil, qui comportera des onglets (navigation bar) :



  - Ajouter des icônes aux différents onglets

- **Fn - Pour aller plus loin :** Insérer un système d'auto-complétion aux input de l'application.
- **Fn +1** - Sentez-vous libre de rajouter autant de fonctions que vous souhaitez.

5 . jQuery Mobile et HTML 5

5.1 . Customisation des styles et formatage de contenus

5.1.1 Créer son propre thème

jQuery Mobile offre la possibilité d'assigner une couleur (thème) à différents types d'éléments grâce à l'attribut spécial **data-theme**, qui peut prendre une valeur allant de **a** à **z**.

Exemple : cette page utilise le thème "a".

```
<div data-role= "page" data-theme="a">
                …
</div>
```

La feuille de style jQuery Mobile (**jquery.mobile.css**) implémente déjà les thèmes **a**, **b**, **c**, **d** et **e**. Les autres ne le sont pas. On peut donc sans aucune crainte les implémenter, en leur assignant nos propres styles (couleur, ombrage,...).

Supposons que l'ont souhaite créer un thème dont la couleur d'arrière-plan sera **#a7a7f7**. On pourrait créer son propre fichier CSS et écraser les styles jQuery Mobile (couleur d'arrière-plan). C'est une solution intéressante, cependant jQuery Mobile en propose une autre qui est beaucoup plus élégante et respectueuse des conventions de développement propres aux Framework.
Voici donc comment faire :

- Définir dans un fichier CSS tout ou partie des classes suivantes :
  - **ui-bar-[a-z]** : style appliqué dans les toolbars (header et footer)
  - **ui-body-[a-z]** : style appliqué au body de la page
  - **ui-btn-up-[a-z]** : s'applique aux boutons et à tous les éléments cliquables.

Si l'on décide de désigner notre thème par la lettre **'w'** par exemple, on devra concrètement définir les classes **ui-bar-w**, **ui-body-w**, **ui-btn-up-w**.

On créé un fichier CSS où l'on définira nos nouvelles classes

```
/** == Mon style personnalisé == **/
.ui-bar-w, .ui-body-w, .ui-btn-up-w {
        background:#a7a7f7;
}
```

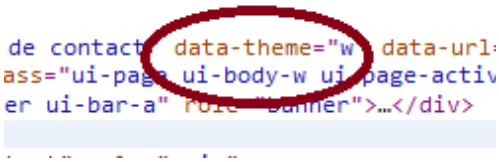
Ensuite, l'appel du thème se fera en ajoutant simplement un **data-theme= "w"** à l'endroit où l'on souhaite appliquer notre style. Pour jQuery Mobile il s'agira simplement d'injecter une classe portantant le suffixe **'w'**.

On insère le thème dans une page

```
<div data-role="page" data-theme="w">
```

Cette solution a l'avantage d'être générique, de plus on a pas se demander si l'on doit nécessairement importer notre feuille de style avant ou après celle de jQuery, car ils n'entreront pas de toute façon en conflit.

Dans l'image ci-dessous on peut effectivement voir que jQuery a automatique injecté une classe **ui-body-w** après qu'on est précédemment précisé dans la page un **data-theme="w"** :



Visualisation avec Firebug

Voir démo 1 et 1_bis (chapitre 5)

5.1.2 Créer une icône personnalisée

La customisation d'une icône suit la même logique que celle d'un thème. Dans sa convention de nom, jQuery Mobile s'est arrangé à ce qu'il y est des classes comportant le même préfixe pour une certaine catégorie de styles.

Supposons que l'on souhaite insérer dans notre application une icône que l'on décide de nommer "funny". En fait cela revient à créer une classe qui devra s'appeler **ui-icon-funny**. On a donc simplement rajouté le nom de notre icône à la suite du suffixe ***ui-icon-***.

Il faut ensuite dans une feuille de style déclarer notre icône :

```
/** == Mon style personnalisé == **/
.ui-icon-funny {
        background-image: url("mon_image.png");
}
```

Pour respecter le style visuel de l'ensemble de l'application, il faut que l'image soit en 18x18 pixels. Il s'agit de la résolution standard.
Cependant, de plus en plus de smartphones sont à très haute résolution. Il faut donc prendre cela en compte. Pour créer une icône haute résolution, il faut créer une autre image en 36x36 pixels (le double) et utiliser la puissance des media queries :

```
@media only screen and (-webkit-min-device-pixel-ratio: 2) {
        .ui-icon-funny {
                background-image: url("mon_image.png");
                background-size: 18px 18px;
        }
        ...
}
```

L'appel de l'icône se fera en faisant simplement data-icon="funny".

5.1.3 Regrouper des éléments par colonnes

jQuery Mobile permet de construire très facilement des blocs de contenus grâce un ensemble de classes dont le suffix est **ui-grid-**. 4 styles de colonnes ont déjà été définis par le Framework. :

- **ui-grid-a** : affiche deux blocs de contenus
- **ui-grid-b** : affiche trois blocs de contenus
- **ui-grid-c** : affiche quatre blocs de contenus
- **ui-grid-d** : affiche cinq blocs de contenus

Exemple : avec deux blocs de contenus :

```
<div class="ui-grid-a">
      <div class="ui-block-a"><strong>Un bloc A</strong> et un peu de
texte</div>
      <div class="ui-block-b"><strong>Un bloc B</strong> et un peu de
texte</div>
</div><!-- /grid-a -->
```

Voir démo 2 (chapitre 5)

5.1.4 Contenus coulissants et accordéons

Pour créer un contenu coulissant, il suffit de rajouter **data-role= "collapsible"** à un bloc div dans le **data-role="content"** :

```
<div data-role="content">
      <div data-role="collapsible" data-theme="b" data-content-theme="d">
            <h3>[Un titre accrocheur] Apple va faire faillite !</h3>
```

```
            <p>...non c'est pas vrai...du moins pas avant un bon bout de
temps</p>
      </div>
      <div data-role="collapsible" data-theme="b" data-content-theme="d">
            <h3>La terre est ronde !</h3>
            <p>...et non pas plate...C'est une révolution !</p>
      </div>
</div><!-- /content -->
```

Par défaut le contenu ne s'affiche pas, et il faut donc cliquer sur le bouton pour l'ouvrir.

Voir démo 3 (chapitre 5)

On peut ensuite regrouper ces éléments coulissants pour former des accordéons. jQuery Mobile propose l'attribut **data-role="collapsible-set"** pour le générer :

```
<div data-role="content">
      <div data-role="collapsible" data-theme="b" data-content-theme="d">
            <h3>[Un titre accrocheur] Apple va faire faillite !</h3>
            <p>...non c'est pas vrai...</p>
      </div>
      <div data-role="collapsible" data-theme="b" data-content-theme="d">
            <h3>La terre est ronde !</h3>
            <p>...et non pas plate...C'est une révolution !</p>
      </div>
</div><!-- /content -->
```

Voir démo 3_bis (chapitre 4)

5.2 . Balises HTML 5 spéciales

5.2.1 Insérer du contenu multimédia : les balises audio et video

La sortie de HTML5 a été l'occasion d'introduire de nouvelles balises. Longtemps gérés par des plugins et autres outils comme Flash ou Silverlight, les lecteurs audio et vidéo sont désormais disponible en HTML. Comparé à Flash, la déclaration d'une vidéo ou d'un fichier audio en HTML 5 peut se faire en 2 lignes. La plupart du temps on rajoute un peu plus de lignes que ça, mais cela est optionnel.

```
<p>
      <video width="320" height="240" controls="controls">
     <source src="http://www.w3schools.com/html5/movie.ogg"
type="video/ogg" />
     <source src="http://fa.cotecine.fr/pod/7601%20vf%20fa3.mp4"
type="video/mp4" />
            Votre mobile semble ne pas être compatible le format video
html5.
      </video>
</p>
```

Voir la démo 4 (chapitre 5)

5.2.2 Interagir avec son téléphone: les attributs email et tel

Avec HTML5 il est désormais possible d'interagir plus ou moins facilement avec son appareil mobile. De nouveaux attributs comme **email** et **tel** ont ainsi été ajoutés aux champs de saisies :

```
<input type= "email" value="charles@charlesen.fr" />
<input type= "tel" value="+336xxxxxxxx" />
```

(Les croix sont bien sûr à remplacer par des chiffres)

Ces deux input, plus sémantiques, permettent au téléphone d'adapter son clavier à la saisie en cours. Ainsi, un utilisateur qui serait en train de saisir son adresse mail dans le champs de saisie email verra son clavier mettre plus avant des mots ou caractères comme l'arobase, le '.' ou encore les extensions de domaines (.com, .fr).

Par ailleurs, ces même attributs existent aussi pour les liens hypertextes. Si le mail en lien hypertexte existe depuis longtemps, la possibilité d'écrire son numéro de téléphone dans un lien hypertexte est beaucoup plus récente.

Un clic sur le lien déclenchera, depuis un téléphone mobile, un appel vers le numéro indiqué

```
<a href= "tel :+336xxxxxxxx" />
```

5.2.3 Autres attribut HTML 5 : search, url

Ces attributs contribuent eux aussi à améliorer l'expérience utilisateur. Ils sont particulièrement utiles dans un contexte de mobilité car en mettant en déclenchant la mise en avant de certaines touches du clavier, ils améliorent considérablement la saisie de données.

Exemple : les attributs search et url facilitent eux aussi la saisie au clavier.

```
<input type= "search" value="" />
<input type= "url" value="charlesen.fr" />
```

5.3 . Persistance de données

5.3.1 Stockage dans la session

HTML5 met à disposition la fonction sessionStorage, qui permet de stocker des données pour le temps d'un succès.
sessionStorage est un objet de type storage qui représente un espace de stockage défini comme attribut de chaque fenêtre. Il est créé pour un nouveau visiteur et supprimé quand l'utilisateur se déconnecte, ou à la demande d'un script.

Il est implémenté dans les navigateurs depuis Firefox 3, Internet Explorer 8 et sur Chrome, Safari. Le stockage se fait simplement :

```
// La sauvegarde en Session
sessionStorage.login = "utilisateur";

// Récupérer en Session
var login = sessionStorage.login
```

5.3.2 Stockage permanent côté client

Avec la venue du HTML5, une nouvelle méthode de stockage des données est introduite: Le stockage local (localStorage).C'est sans doute l'une des prorpiétés les plus intéressantes du HTML5: Le local storage, ou le stockage local.

En effet, grâce à cette propriété, il sera désormais possible de stocker des données localement dans votre navigateur (supportant la propriété bien sûr).

Les données sont stockées pour un domaine précis et peuvent récupérées dans n'importe quelle page du domaine en question. La propriété s'annonce comme le remplacement des cookies à la différence qu'il est possible de stocker bien plus d'informations (Environ 5 Mb) et que seul le client peut accéder à ces données.

Stocker une valeur en local se fait de deux façons différentes :

- **localeStorage**['une_cle'] = 'une_valeur' ;
- **localeStorage.setitem**('ma_cle', 'une_valeur');

Récupérer des valeurs se fait presque aussi simplement que le stockage :

- **localeStorage**['une_cle']
- **localeStorage.getitem**['ma_cle'];

Supprimer un élément :

- **removeItem**('ma_cle')

5.4 . Utilisation de la Géolocalisation

5.4.1 Connaître ses coordonnées géographiques

La géolocalisation est un point de plus en plus pris en compte dans le développement d'applications mobiles. Avec HTML5 il est assez facile de récupérer les coordonnées géographiques, un plus à ajouter à nos applications jQuery Mobile.
La géolocalisation, souvent critiquée dans ses débuts, est désormais présent dans de nombreux navigateurs. On peut donc l'utiliser à priori sans craindre de mauvaises surprises. Le tableau ci-dessous nous présente sa prise en charge par différents navigateurs (desktop et mobile) :

IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser	
		4.0							
		5.0							
		6.0							
	2.0	7.0							
	3.0	8.0							
	3.5	9.0							
	3.6	10.0		9.0					
	4.0	11.0		9.5-9.6					
	5.0	12.0		10.0-10.1					
	6.0	13.0		10.5					
5.5	7.0	14.0	3.1	10.6			10.0		
6.0	8.0	15.0	3.2	11.0	3.2		11.0	2.1	
7.0	9.0	16.0	4.0	11.1	4.0-4.1		11.1	2.2	
8.0	10.0	17.0	5.0	11.5	4.2-4.3		11.5	2.3	3.0
9.0	11.0	18.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0	
10.0	12.0	19.0	6.0	12.0					
	13.0	20.0							

Nous allons voir comment il est possible de récupérer les coordonnées d'un utilisateur et comment gérer les cas de succès et d'erreur lors du démarrage du processus de géolocalisation grâce à des fonctions appellées callbacks.

Nous verrons aussi comment suivre ses déplacements, connaître sa vitesse et même sa direction.

La première chose à faire est de tester la prise en charge de la géolocalisation par le device client (ordinateur, smartphone, tablette, …). Il suffit de faire un simple test conditionnel :

```
// Vérifie la prise en charge de la géolocalisation par le navigateur courant
if (navigator.geolocation) { … }
```

le navigator.geolocation renvoi vrai si le navigateur prend en charge la géolocalisation, et faux sinon.

Si ce test est positif, nous pourrons donc, après validation de l'utilisateur d'être géolocalisé, afficher tranquillement nos coordonnées géographiques. C'est ce que nous précisons dans le code ci-dessous :

```
<script>
	if (navigator.geolocation) {
		var watchId =
navigator.geolocation.watchPosition(successCallback, errorCallback,
{enableHighAccuracy:true});
	}
	else {
		alert("Votre navigateur ne prend pas en compte la
géolocalisation HTML5");
	}
</script>
```

Le paramètre **enableHighAccuracy** permet d'activer le GPS

successCallback et **errorCallback** sont appelées fonctions de Callback et permettent de gérer les cas de succès et d'échec du processus de géolocalisation. Trois types d'échecs sont possible :

- Permission refusée par l'utilisateur
- Indétermination de la position de l'utilisateur
- Timeout dans l'exécution de la requête

Nous définissons ensuite les fonctions de Callback :

```
function successCallback(position){
      $('div[data-role="content"]').after('<p>Latitude : ' +
position.coords.latitude + '</p> <br />' + '<p> Longitude :' +
position.coords.longitude + '</p>');
           };
function errorCallback(error){
      switch(error.code){
            case error.PERMISSION_DENIED:
                  alert("L'utilisateur n'a pas autorisé l'accès à sa
position");
                  break;
            case error.POSITION_UNAVAILABLE:
                  alert("L'emplacement de l'utilisateur n'a pas pu
être déterminé");
                  break;
            case error.TIMEOUT:
                  alert("Le service n'a pas répondu à temps");
                  break;
      }
};
```

Voir démo 6 (chapitre 5)

5.4.2 Afficher une fenêtre Google Maps

Reprenons notre exemple précédent. On souhaite afficher une Google Maps en fonction de sa position géographique. Google fournit une API permanent de récupérer un lien vers une carte dont on aura renseigner les coordonnées géographiques.

Voir démo 6 bis (chapitre 5)

6 . Développer avec le Framework PhoneGap

6.1 . jQuery Mobile et PhoneGap

PhoneGap est un framework de développement d'applications natives en Javascript, CSS et HTML. Elle a la particularité de pouvoir exploiter les fonctionnalités et les ressources du mobile sur lequel elle se lance (géolocalisation, accéléromètre, etc.).. Les applications sont construites comme de simples pages HTML avant d'être packagées pour s'exécuter dans un UIWebView (iOS) ou une WebView (Android).

6.2 . Développer avec PhoneGap

6.2.1 Pré-requis

Suivant que l'on souhaite développer pour l'un ou l'autre des OS mobiles existants, il faut respecter plusieurs conditions :

- Pour développer avec **iOS**, il faut :
  - MAC OS X Snow Leopard
  - La certification iOS développeur (pour publier votre app)
  - Xcode
  - Le SDK (kit de développement) de PhoneGap

- Pour développer sous **Android**, il faut :
  - Eclipse 3.4+
  - Le SDK Android
  - Le plugin ADT pour Eclipse (Outils de developpement Android)
  - Le SDK de PhoneGap

- Pour développer pour **Windows Phone**, il faut :
  - Windows 7 ou Vista SP2
  - Le SDK des Windows Phone
  - Le SDK de PhoneGap

6.2.2 Installation de PhoneGap

Une explication détaillée pour chaque plate-forme est disponible à cette adresse : http://phonegap.com/start

6.2.3 Importer son projet jQuery Mobile

Le développement PhoneGap se fait en html, JavaScript et css. Les fichiers nécessaires au développement son stockés dans le dossier www. C'est dans ce dossier qu'il faudra importer les fichiers jQuery Mobile.

Annexes :

Émulateurs mobiles

- **iPhone Tester :** http://iphonetester.com
- Vous trouverez une liste plus complète à cette adresse : http://www.mobilexweb.com/emulators

Sites et blogs

- **Site et documentation officiel de jQuery Mobile :** http://jquerymobile.com
- **Documentation officiel de jQuery :** http://api.jquery.com
- **Mobile-tuts! :** site d'actualités et de tutoriels autour des technologies mobiles
- **Sélecteurs jQuery :** Une liste complète de sélecteurs jQuery est disponible à l'adresse : http://jquery.developpeurweb2.com/documentation/selecteurs.php
- **Fonctions de manipulation du DOM** : Une liste complète de fonctions de manipulation du DOM est disponible à l'adresse : http://jquery.developpeur-web2.com/documentation/manipulation.php
- **jQM Offline :** http://www.raymondcamden.com/index.cfm/2011/3/12/Building-an-offline-capable-mobile-web-site-with-jQuery-Mobile
- **jQM Developement tips:** http://www.jquery4u.com/mobile/50-jquery-mobile-development/
- **jQuery Mobile Running** : http://www.amazon.com/jQuery-Mobile-Running-Maximiliano-Firtman/dp/1449397654

Références

1. **International Telecommunication Union (ITU)**. *The World in 2011 – ICT Facts and Figures*, Novembre 2011 : http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf

2. J. Chaffer & K. Swedberg. ***JQuery : Simplifiez et enrichissez vos développements JavaScript***, du titre original **Learning jQuery 1.3**. Traduit de l'américain par Hervé Soulard. ISBN original : 978-1-847196-70-5. ISBN de la traduction : 978-2-7440-2381-1