jQuery Mobile

La bibliothèque JavaScript pour le Web mobile

Éric Sarrion

EYROLLES

jQuery Mobile

La bibliothèque JavaScript pour le Web mobile

Éric Sarrion

Formateur et développeur en tant que consultant indépendant, Éric Sarrion participe à toutes sortes de projets informatiques depuis plus de 25 ans. Il est l'auteur de plusieurs ouvrages sur le développement web aux éditions O'Reilly France, parmi lesquels *Pratique de CSS et JavaScript*, et aux éditions Eyrolles (*jQuery & jQuery UI, XHTML/CSS & JavaScript pour le Web mobile, Prototype et Scriptaculous*).

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 (iPhone, iPad, Android…).

L'hétérogénéité des écrans de terminaux mobiles est un véritable casse-tête technique pour le développeur de sites web mobiles. Grâce à jQuery Mobile, il peut désormais créer facilement des sites et applications web performantes, qui s'adaptent à tous types d'interfaces – y compris tactiles – pour une ergonomie optimale. Aujourd'hui en version 1.0, jQuery Mobile est déjà déclarée « Innovation de l'année » par les .Net Awards !

Complet et didactique, cet ouvrage explique méthodiquement toutes les facettes de la bibliothèque, illustrées par d'indispensables exemples d'utilisation. 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, il donne tous les éléments pour construire des sites qui fonctionneront sur la plupart des supports mobiles actuels. Enfin, une étude de cas détaille la réalisation d'une application de communication entre personnes (messagerie), afin de mettre en pratique les concepts et conseils techniques exposés dans le reste du livre.

Au sommaire

À qui s'adresse cet ouvrage ?

– À tous les développeurs et chefs de projet web qui visent les navigateurs de smartphones et de tablettes ;
– Aux intégrateurs HTML/CSS et JavaScript.

Code éditeur : G13388
ISBN : 978-2-212-13388-2

Conception : Nord Compo
© Image de couverture : Thinkstock

jQuery Mobile

La bibliothèque JavaScript pour le Web mobile

Du même auteur

E. Sarrion – **jQuery & jQuery UI.**
N°12892, 2011, 520 pages.

E. Sarrion. – **XHTML/CSS et JavaScript pour le Web mobile**. *Développement iPhone et Android avec et iUI et XUI.*
N°12775, 2010, 274 pages.

E. Sarrion. – **Prototype et Scriptaculous**. *Dynamiser ses sites web avec JavaScript*.
N°85408, 2010, 342 pages (e-book).

Dans la même collection

R. Goetter. – **CSS avancées.** *Vers HTML 5 et CSS 3*.
N°13405, 2e édition, 2012, 400 pages.

R. Rimelé. – **HTML 5.** *Une référence pour le développeur web*.
N°12982, 2011, 604 pages.

F. Daoust, D. Hazaël-Massieux. – **Relever le défi du Web mobile.** *Bonnes pratiques de conception et de développement*.
N°12828, 2011, 300 pages.

J. Chable, D. Guignard, E. Robles, N. Sorel. – **Programmation Android.**
N°13303, 2e édition, 2012, 520 pages environ.

T. Sarlandie, J.-M. Lacoste. – **Programmation IOS 5 pour iPhone et iPad.**
N°12799, 2e édition, 2012, 350 pages environ.

P.-Y. Chatelier. – **Objective-C pour le développeur avancé.**
N°12751, 2010, 224 pages.

J. Stark. – **Applications iPhone avec HTMl, CSS et JavaScript.** *Conversions en natifs avec PhoneGap*.
N°12745, 2010, 190 pages.

J.-M. Defrance. –**Ajax, jQuery et PHP**. *42 ateliers pour concevoir des applications web 2.0*.
N°13271, 3e édition, 2011, 482 pages.

C. Porteneuve. – **Bien développer pour le Web 2.0**. *Bonnes pratiques Ajax*.
N°12391, 2e édition, 2008, 674 pages.

S. Jaber. – **Programmation GWT 2**. *Développer des applications RIA et Ajax avec le Google Web Toolkit*.
N°12569, 2010, 484 pages.

E. Daspet, C. Pierre de Geyer. – **PHP 5 avancé**.
N°13435, 6e édition, 2012, 900 pages environ.

J. Pauli, G. Plessis, C. Pierre de Geyer. – **Audit et optimisation LAMP**.
N°12800, 2012, 300 pages environ.

D. Seguy, P. Gamache. – **Sécurité PHP 5 et MySQL**.
N°13339, 3e édition, 2011, 277 pages.

A. Vannieuwenhuyze. – **Programmation Flex 4**.
N°12725, 2e édition, 2010, 604 pages.

T. Ziadé. – **Programmation Python**. *Conception et implémentation*.
N°12483, 2e édition 2009, 586 pages.

P. Borghino, O. Dasini, A. Gadal. – **Audit et optimisation MySQL 5**.
N°12634, 2010, 282 pages.

Chez le même éditeur

E. Marcotte. – **Responsive Web Design**.
N°13331, 2011, 160 pages. (A Book Apart).

J. Keith. – **HTML5 pour les web designers**.
N°12861, 2010, 98 pages. (A Book Apart).

D. Cederholm. – **CSS3 pour les web designers**.
N°12987, 2011, 132 pages. (A Book Apart).

E. Kissane. – **Stratégie de contenu web**.
N°13279, 2011, 96 pages. (A Book Apart).

A. Walter. – **Design émotionnel**.
N°13398, 2011, 110 pages. (A Book Apart).

E. Sloïm. – **Mémento Sites web.** *Les bonnes pratiques*.
N°12802, 3e édition, 2010, 18 pages.

A. Boucher. – **Ergonomie web illustrée**. *60 sites à la loupe.*
N°12695, 2010, 302 pages. (Design & Interface).

A. Boucher. – **Ergonomie web**. *Pour des sites web efficaces.*
N°13215, 3e édition, 2011, 356 pages.

I. Canivet. – **Bien rédiger pour le Web**. *Stratégie de contenu pour améliorer son référencement naturel.*
N°12883, 2e édition, 2011, 552 pages.

O. Andrieu. – **Réussir son référencement web**. *Édition 2012*.
N°13396, 4e édition, 2011, 700 pages.

N. Chu. – **Réussir un projet de site web**.
N°12742, 6e édition, 2010, 256 pages.

S. Bordage, D. Thévenon, L. Dupaquier, F. Brousse. – **Conduite de projet Web**.
N°13308, 6e édition, 2011, 480 pages.

jQuery Mobile

La bibliothèque JavaScript pour le Web mobile

Éric Sarrion

Avec la contribution de Thomas Bertet

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Chaleureux remerciements à Thomas Berthet pour sa contribution et à Anne Rothé pour sa relecture.

Avant-propos

La date du 16 novembre 2011 est une date importante dans le monde du Web mobile. C'est ce jour-là que Todd Parker, responsable du projet jQuery Mobile, annonce la sortie de la version officielle 1.0 tant attendue de la bibliothèque, après presque dix versions intermédiaires. Ayant bénéficié du retour d'expérience de milliers d'utilisateurs à travers le monde durant l'année écoulée, elle fait preuve d'une maturité certaine.

S'appuyant sur la bibliothèque JavaScript jQuery, qui est une référence dans le domaine, jQuery Mobile est aujourd'hui une bibliothèque adaptée aussi bien pour les téléphones mobiles que pour les tablettes tactiles. Elle prend en compte toutes les spécificités de ces nouveaux supports, dont les fonctionnalités et les interfaces se démarquent clairement du Web traditionnel. Conçue pour être facile d'utilisation, tout en étant performante et surtout portable sur la plupart des téléphones ou tablettes, on peut donc supposer qu'il sera difficile de la concurrencer. D'ailleurs, une semaine à peine après la sortie officielle de la version 1.0, elle a déjà gagné le titre de « *Innovation of the year* » décerné par .Net Awards.

Adressé aussi bien aux développeurs, intégrateurs et chefs de projet, qu'aux étudiants, ce livre a pour objectif d'expliquer toutes les facettes de cette bibliothèque, afin de permettre la construction de sites ou d'applications qui fonctionneront sur la plupart des supports mobiles actuels.

Alors en route pour découvrir cette fabuleuse bibliothèque qu'est jQuery Mobile !

À LIRE **Le guide zen du développeur mobile**

Avec le Web mobile, de nouveaux enjeux, aussi bien fonctionnels qu'ergonomiques apparaissent, sans compter la nécessité d'adaptation à la variété des terminaux disponibles sur le marché. Voici un ouvrage qui fait le point sur la situation, prodiguant conseils et bonnes pratiques pour éviter les écueils et répondre intelligemment aux problématiques mobiles.

📖 F. Daoust, D. Hazaël-Massieux, *Relever le défi du Web mobile : Bonnes pratiques de conception et de développement*, Eyrolles, 2011

À LIRE jQuery et jQuery UI

Pour maîtriser toutes les fonctionnalités de la bibliothèque jQuery, ainsi que de son module jQuery UI, permettant de créer des composants graphiques avancés, référez-vous à l'ouvrage suivant :

📖 É. Sarrion, *jQuery & jQuery UI*, Eyrolles, 2011

Structure de l'ouvrage

Ce livre est découpé en trois parties.

- La première concerne l'étude de jQuery Mobile d'un point de vue « design de l'application ». Vous y apprendrez à utiliser les composants HTML et les attributs correspondants permettant d'afficher les fenêtres et leur contenu à l'écran. À l'issue de cette partie, l'aspect de votre site ou de votre application sera adapté au monde mobile, vraiment différent d'un site web traditionnel.
- La deuxième partie est une partie plus technique qui traite de l'utilisation de JavaScript avec jQuery Mobile. Elle vous montre comment faire communiquer votre application avec le monde extérieur, par exemple pour afficher des données d'un serveur ou afficher une carte Google Maps.
- Enfin, la troisième partie propose une étude de cas, afin de mettre en application les concepts exposés dans les parties précédentes. Nous y détaillons la réalisation d'une application de communication entre personnes (messagerie). Bien sûr, cette partie ne peut être comprise que si les précédentes ont été assimilées.

Table des matières

PREMIÈRE PARTIE

Afficher les composants graphiques

1

Installation de jQuery Mobile

Le but de ce livre étant d'utiliser jQuery Mobile afin de créer des sites web accessibles depuis les téléphones mobiles ou des tablettes graphiques comme l'iPad, nous devons pour cela commencer par installer un serveur web qui contiendra les pages HTML de notre site. Nous verrons ensuite comment installer la bibliothèque, en ajoutant quelques précisions quant aux paramétrages spécifiques au développement pour iPhone.

Installation d'un serveur web

Pour héberger notre site, n'importe quel type de serveur web fait l'affaire (PHP, .Net, Java, Ruby on Rails, etc.). Dans cet ouvrage, nous prenons en exemple un serveur PHP.

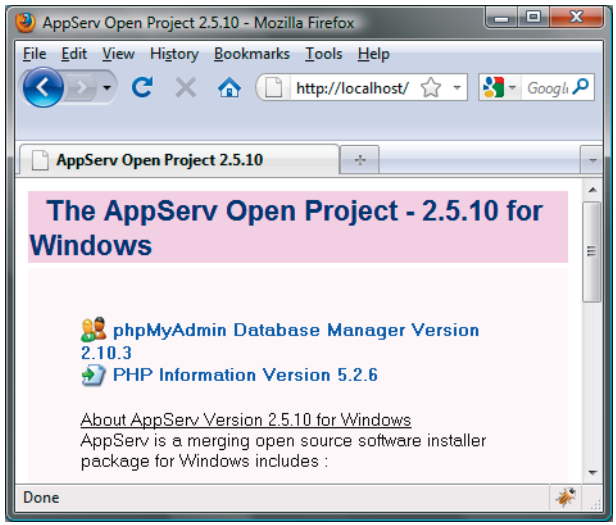
Le serveur PHP installé sera différent selon que l'on est sous Windows ou sous Mac OS :

- sous Windows, on installera AppServ (http://www.appservnetwork.com) ;
- sous Mac OS, on installera MAMP (Mac, Apache, MySQL, PHP, http://www.mamp.info).

Une fois le serveur installé et lancé, vérifiez que tout est correct en saisissant l'URL http://localhost dans la barre d'adresse de votre navigateur. Vous devriez voir s'afficher la

page d'accueil du serveur. La figure 1-1 représente cette fenêtre pour AppServ sous Windows, dans un navigateur Firefox.

Figure 1–1
Page d'accueil du serveur PHP



> **REMARQUE** **Emplacement des fichiers sur le serveur**
>
> Sous AppServ, les fichiers sont situés dans `appserv/www`.
> Sous MAMP, ils sont situés sous `MAMP/htdocs`.
> Des sous-répertoires peuvent être créés à ces emplacements pour contenir les pages de notre site (avec toutes ses composantes : images, etc.).

Si aucun nom de fichier n'est indiqué à la fin de l'URL, il correspond au fichier par défaut `index.html`. Donc l'URL http://localhost désigne en fait le fichier `index.html` situé dans `appserv/www` (sous Windows) ou `MAMP/htdocs` (sous Mac OS).

Installation de jQuery Mobile

Vous pouvez télécharger le fichier ZIP contenant les sources de la bibliothèque, à l'adresse http://jquerymobile.com/download/. Vous pouvez également utiliser les fichiers se trouvant sur le serveur code.jquery.com, comme cela est indiqué dans la page de téléchargement affichée (voir ci-après la section « Installation indépendante d'un type de serveur »).

Quelle que soit l'installation que vous choisissez (PHP, Ruby On Rails ou autre), vous devez utiliser la bibliothèque jQuery standard (à partir de la version 1.6), que

vous pouvez télécharger sur http://jquery.com. Elle se trouve également dans le répertoire `demos` de jQuery Mobile.

Installation sous un serveur PHP

Décompressez le fichier ZIP dans le répertoire du serveur (ou un sous-répertoire que vous créez dans celui-ci) :

- répertoire `appserv/www` sous Windows ;
- répertoire `MAMP/htdocs` sous Mac OS.

Ici, nous décompressons les fichiers dans un répertoire `test` du serveur qui contiendra les fichiers de notre application. Après décompression, ce répertoire contient un sous-répertoire contenant jQuery Mobile (ici `jquery.mobile-1.0`, que l'on renomme en `jquery.mobile`). Le répertoire contenant jQuery Mobile contient une liste de fichiers :

- `jquery.mobile-1.0.css`, que l'on renomme en `jquery.mobile.css` : il correspond au fichier CSS *(Cascading Style Sheet)* de jQuery Mobile, en version non compressée. Ce fichier servira à styler les pages HTML affichées dans le navigateur du téléphone ;
- `jquery.mobile-1.0.js`, que l'on renomme en `jquery.mobile.js` : il correspond au fichier JavaScript de jQuery Mobile, en version non compressée. Ce fichier servira à utiliser du code JavaScript pouvant s'exécuter dans le navigateur du téléphone ;
- `jquery.mobile-1.0.min.css` : version compressée de `jquery.mobile-1.0.css` ;
- `jquery.mobile-1.0.min.js` : version compressée de `jquery.mobile-1.0.js` ;
- enfin, le répertoire `images`, contenant certaines images qui seront affichées dans les pages HTML à l'aide de directives CSS (fonctionnement géré en interne par jQuery Mobile).

Un exemple de code d'une application basique sous PHP serait le suivant (il correspond au fichier `index.html`).

Code d'une application sous PHP (fichier index.html)

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>
```

```
<body>

<div data-role=page>
  <div data-role=header>
    <h1>Titre de la fenêtre</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre </p>
  </div>
</div>

</body>
</html>
```

Nous incluons le fichier CSS de jQuery Mobile, puis les deux fichiers JavaScript correspondant à jQuery et jQuery Mobile. Notez que le fichier de jQuery doit être inclus avant celui de jQuery Mobile, le second ayant besoin du premier pour fonctionner. Le fichier `jquery.js` est supposé ici être situé dans le même répertoire que le fichier `index.html`.

La directive `<!DOCTYPE html>` permet de s'assurer que le navigateur utilisé dans le téléphone prendra en compte certaines spécificités incluses dans jQuery Mobile (qui, autrement, ne le seraient peut être pas). On inclura donc cette directive dans chacune de nos pages HTML.

La directive `<meta>` et ses attributs permettent d'indiquer que l'affichage peut s'effectuer sur un écran dont les dimensions sont inférieures aux écrans traditionnels (par exemple, un écran de téléphone mobile). Ainsi, la taille des caractères est ajustée en conséquence.

Les éléments `<div>` inclus dans la page HTML correspondent aux éléments qui seront affichés dans la fenêtre du navigateur. Ces éléments sont expliqués en détail dans le chapitre suivant.

Vérifions que la page s'affiche correctement dans le navigateur d'un téléphone mobile (ici, sur la figure 1-2, un iPhone, mais le résultat est identique pour les autres types de téléphones pris en charge par jQuery Mobile). Le test doit s'effectuer en indiquant l'adresse IP du serveur (ici http://192.168.1.30/test, car notre application est dans le répertoire `test` du serveur).

> **REMARQUE** **Connaître l'adresse IP du serveur**
>
> Pour connaître l'adresse IP du serveur, il suffit de taper, dans une fenêtre de commandes, l'instruction `ipconfig` sous Windows, ou `ifconfig` dans un environnement Unix (Mac OS ou Linux).

Figure 1–2
Test d'une application minimale

Installation indépendante d'un type de serveur

Dans le cas où l'on ne souhaite pas avoir les fichiers JavaScript et CSS sur son serveur, il est possible de les inclure depuis un serveur externe. Ils sont présents sur le serveur code.jquery.com. Le fichier `index.html` contient dans ce cas le code suivant.

Fichier index.html

```
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet
       href=http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css />
  <script src=http://code.jquery.com/jquery-1.6.min.js></script>
  <script
      src=http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js>
  </script>
</head>

<body>
<div data-role=page>
  <div data-role=header>
    <h1>Titre de la fenêtre</h1>
  </div>
```

```
  <div data-role=content>
    <p> Contenu de la fenêtre </p>
  </div>
</div>
</body>
</html>
```

L'affichage est identique au précédent (figure 1-2).

À SAVOIR Attributs HTML utilisés par jQuery Mobile

Vous remarquerez dans les exemples précédents que nous avons introduit de nouveaux attributs dans certains éléments HTML, en particulier l'attribut `data-role`, pouvant valoir `"page"`, `"header"`, etc.
Ces attributs (et bien d'autres) ont été créés par jQuery Mobile afin de faciliter l'écriture du code HTML. Ils sont interprétés par jQuery Mobile afin de donner une nouvelle apparence à la page HTML, et de permettre ainsi d'afficher des fenêtres sur l'écran. Nous verrons plus loin que l'on peut donner une nouvelle apparence à la plupart des éléments classiques d'une page HTML, comme par exemple les boutons, les cases à cocher, les listes de sélection, etc.

Paramétrages spécifiques à l'iPhone

Rendre l'application accessible depuis le bureau de l'iPhone

Pour l'instant, notre application est utilisable à travers le navigateur Safari, en donnant l'URL du serveur dans sa barre d'adresse. Une application web peut être aussi accessible comme une autre application iPhone, en ayant sa propre icône sur le bureau. Voici comment procéder :

1. Il faut d'abord qu'une icône puisse être associée à cette application. Pour cela, il faut l'indiquer dans le code HTML :

```
<link rel="apple-touch-icon" href="nom_fichier.png" />
```

   Sur un serveur PHP, l'emplacement du fichier sera relatif à celui de la page HTML qui contient l'instruction HTML précédente.

2. Ensuite, une fois la page d'accueil de notre site affichée dans le navigateur Safari de l'iPhone, il suffit de cliquer sur le bouton + de la barre de boutons située dans la partie basse de l'écran. Un menu s'affiche, dans lequel on choisit *Ajouter à*

l'écran d'accueil. Une nouvelle page s'affiche, dans laquelle on retrouve l'icône indiquée dans notre code HTML (si elle n'est pas visible, c'est que le chemin d'accès indiqué dans le code HTML n'est pas correct), et un libellé modifiable pour indiquer le nom de notre application. Après avoir cliqué sur le bouton *Ajouter*, l'icône s'affiche sur le bureau de l'iPhone et permet un accès direct à notre application.

> **REMARQUE Taille de l'icône**
>
> L'icône doit être de 57 x 57 pixels, sans bords arrondis (elles seront automatiquement affichées de façon arrondie par le système d'exploitation de l'iPhone).

Supprimer l'affichage de la barre d'adresse du navigateur

Une fois l'application accessible via l'icône sur le bureau de l'iPhone, il subsiste un léger problème : le lancement de l'application via l'icône affiche d'abord la barre d'adresse du navigateur, qui disparaît ensuite pour laisser place à notre page HTML. En fait, cette barre d'adresse n'a pas complètement disparu, elle est simplement remontée vers le haut. Si l'on descend la page, elle réapparaît.

Comment la faire disparaître de façon définitive, de façon à ce que l'utilisateur ait vraiment l'impression d'utiliser une application native, et non pas de naviguer sur un site web ? Il suffit d'indiquer dans le code HTML que l'application doit être ouverte en plein écran, grâce à la balise `<meta>` suivante (ajoutée dans la partie `<head>` de la page HTML) :

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

> **ATTENTION Quand insérer cette balise <meta> ?**
>
> Ce tag doit être inscrit dans la page HTML *avant* que l'icône ne soit créée sur le bureau de l'iPhone, sinon cette instruction n'est pas prise en compte. Pensez donc à l'insérer dès le début de la création de votre application et, surtout, avant que les premiers utilisateurs y aient accédé.
> Vous remarquerez également que cette instruction fait disparaître la barre de boutons du bas de l'écran de Safari. Nous avons ainsi vraiment l'impression d'être dans une application native comme celles téléchargées sur l'App Store !

Définir une image affichée au démarrage

Pour donner à l'utilisateur encore davantage l'impression que l'application qu'il utilise est native, il peut être intéressant d'afficher une page au démarrage de l'application. Cette page sera en fait une image de 320 × 460 pixels, destinée à couvrir la zone

d'affichage de l'iPhone. On utilise pour cela la balise `<link>` de la façon suivante (dans la partie `<head>` de la page HTML) :

```
<link rel="apple-touch-startup-image" href="nom_fichier.png" />
```

Sur un serveur PHP, l'emplacement du fichier sera relatif à celui de la page HTML qui contient l'instruction HTML précédente.

ATTENTION Quand insérer cette balise <link> ?

Comme précédemment, ce tag doit être inscrit dans la page HTML *avant* que l'icône ne soit créée sur le bureau de l'iPhone, sinon cette instruction n'est pas prise en compte. Pensez donc à l'insérer dès le début de la création de votre application, et surtout avant que les premiers utilisateurs y aient accédé.

2

Afficher des fenêtres

Dans ce chapitre, nous expliquons la structure minimale des différentes fenêtres, ainsi que les différentes façons de les afficher, de passer de l'une à l'autre (transitions) ou de modifier leur aspect graphique via les feuilles de styles CSS.

Une première fenêtre

Reprenons l'exemple du chapitre 1, qui consistait à afficher une première page HTML dans le navigateur, en utilisant les fonctionnalités de jQuery Mobile.

Une première page HTML utilisant jQuery Mobile

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>
```

```
<div data-role=page>
  <div data-role=header>
    <h1>Titre de la fenêtre</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre </p>
  </div>
</div>

</body>
</html>
```

Notre page HTML inclut principalement un élément `<div>` possédant l'attribut `data-role` de valeur `"page"`. Il correspond à une fenêtre qui sera affichée à l'écran (figure 2-1).

La fenêtre possède, dans notre exemple, deux éléments `<div>` principaux :

- Le premier correspond à la barre de titre de l'application (*header*, en anglais) et possède pour cela l'attribut `data-role` de valeur `"header"`. Il servira à contenir le titre de la fenêtre, ici inclus dans un élément `<h1>`.
- Le second correspond au contenu propre de la fenêtre (*content*, en anglais), situé en dessous de la barre de titre. Il correspond à un élément `<div>` contenant l'attribut `data-role` de valeur `"content"` et pouvant contenir n'importe quels éléments HTML, qui seront affichés dans la fenêtre.

Figure 2–1
Une première fenêtre

Il est également possible de définir une barre située en bas de page. Cette barre est appelée *footer* (pied de page). On l'indique dans le code HTML au moyen de l'attribut `data-role="footer"`. Par exemple :

Une fenêtre contenant un header et un footer

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page>
  <div data-role=header>
    <h1>Titre de la fenêtre</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre </p>
  </div>

  <div data-role=footer>
    <h1>Bas de la fenêtre</h1>
  </div>
</div>

</body>
</html>
```

Comme le montre la figure 2-2, une barre est apparue en bas de la page (à la suite du contenu). Pour placer ce pied de page (ou *footer*) en bas de la fenêtre, il faut que le contenu de la fenêtre soit plus conséquent ou que l'élément `<div>` associé à la barre possède l'attribut `data-position="fixed"`. Cela est étudié dans le chapitre 7, concernant les barres d'outils.

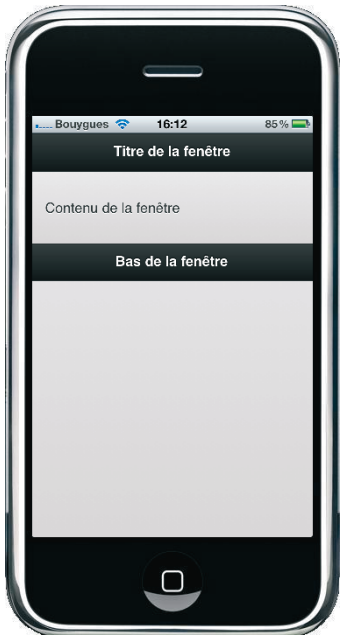


Figure 2–2
Une fenêtre comportant un footer

IMPORTANT **Pages et fenêtres : définissons les termes employés**

jQuery Mobile utilise principalement le terme de *page*, par exemple dans l'attribut `data-role="page"`. La page correspond à la fenêtre affichée à l'écran. On ne doit pas confondre avec la page HTML elle-même, qui, en fait, peut contenir plusieurs fenêtres (ou pages, au sens de jQuery Mobile).
Pour éviter toute confusion, nous emploierons le terme *fenêtre* lorsqu'il s'agit de fenêtre à l'affichage (correspondant aux éléments `<div>` possédant l'attribut `data-role` de valeur `"page"`), et le terme *page HTML* lorsqu'il s'agit de la page HTML incluant une ou plusieurs fenêtres. Nous éviterons d'utiliser le terme *page* employé sans autre qualificatif, trop ambigu selon nous.

Et si on n'utilise pas de fenêtres ?

jQuery Mobile a prévu que vous puissiez écrire une page HTML sans utiliser les conventions précédentes. Cette facilité permet de ne pas bloquer l'affichage si le code HTML n'est pas écrit de façon attendue.

Écrivons maintenant une page HTML basique ne contenant pas les éléments `<div>` tels que précédemment décrits.

Une page HTML s'affichant comme une fenêtre

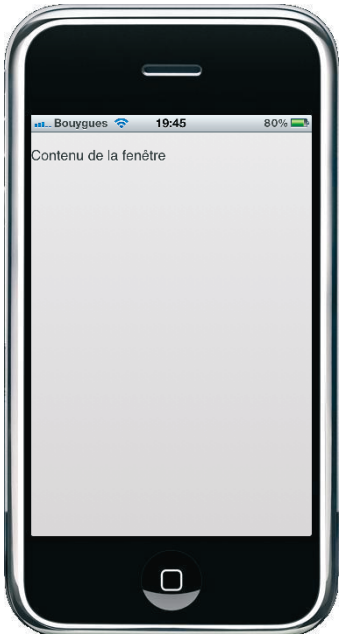
```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<p> Contenu de la fenêtre </p>

</body>
</html>
```

Figure 2–3
Une page HTML sans fenêtre



Nous n'indiquons plus ici les éléments `<div>` possédant les attributs `data-role` de valeur `"page"`, `"header"` ou `"content"`. Pourtant, grâce à jQuery Mobile, cette page HTML s'affiche en tant que fenêtre : le code HTML d'origine a été automatiquement englobé dans un élément `<div>` possédant l'attribut `data-role="page"`.

Passer d'une fenêtre à l'autre

Pour l'instant, notre application ne contient qu'une seule fenêtre, en fait un seul élément `<div>` possédant l'attribut `data-role="page"`. Écrivons une page HTML contenant deux éléments `<div>` possédant cette caractéristique.

Une page HTML contenant deux fenêtres

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win1>
  <div data-role=header>
    <h1>Fenêtre 1</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 1</p>
    <a href=#win2> Aller sur la fenêtre 2 </a>
  </div>
</div>

<div data-role=page id=win2>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 2</p>
  </div>
</div>

</body>
</html>
```

Vous remarquerez que nous avons affecté un attribut `id` aux éléments `<div>` correspondant aux deux fenêtres. La première fenêtre contient un lien permettant d'aller vers la seconde fenêtre, au moyen de l'attribut `href="#win2"`, `win2` étant l'identifiant de cette fenêtre (figure 2-4). Vérifions que cela fonctionne en cliquant sur le lien.

Figure 2–4 Lien dans une fenêtre

Figure 2–5 Seconde fenêtre de la page HTML

La nouvelle fenêtre apparaît (figure 2-5), mais il peut parfois être utile d'insérer un bouton *Back* permettant de revenir automatiquement à la fenêtre précédente. Ce bouton s'affiche normalement dans la barre de titre de la nouvelle fenêtre, dans sa partie gauche.

Pour cela, jQuery Mobile demande d'utiliser l'attribut `data-add-back-btn` de valeur `"true"`. Cet attribut doit être positionné sur l'élément `<div>` correspondant à la fenêtre dans laquelle sera affiché le bouton. On aura donc :

Afficher le bouton Back dans la seconde fenêtre

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>
```

```
<div data-role=page id=win1>
  <div data-role=header>
    <h1>Fenêtre 1</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 1</p>
    <a href=#win2> Aller sur la fenêtre 2 </a>
  </div>
</div>

<div data-role=page id=win2 data-add-back-btn=true>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 2</p>
  </div>
</div>

</body>
</html>
```

On obtient maintenant l'écran représenté sur la figure 2-6.

Figure 2–6
Affichage du bouton Back

Cas des fenêtres situées dans des pages HTML différentes

Les fenêtres précédentes étaient toutes incluses dans un seul fichier. La transition entre les deux fenêtres s'effectue sans requête au serveur, vu que les deux fenêtres sont présentes en mémoire lors du chargement de la page HTML (même si à un instant donné, une seule fenêtre est visible à l'affichage).

jQuery Mobile permet d'inscrire les fenêtres dans des fichiers séparés. Pour charger la seconde fenêtre, il effectue un appel Ajax au serveur, permettant de récupérer le code HTML de la fenêtre. Elle est alors insérée dans l'arborescence du DOM *(Document Object Model)* puis affichée.

Fichier contenant la première fenêtre (index.html)

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=home>
  <div data-role=header>
    <h1>Home</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la Fenêtre 1 </p>
    <a href=index2.html> Aller dans la fenêtre située dans index2.html </a>
  </div>
</div>

</body>
</html>
```

Fichier contenant la seconde fenêtre (index2.html)

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv=Content-Type content=text/html;charset=iso-8859-1 />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
```

```
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win2 data-add-back-btn=true>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la Fenêtre 2 </p>
  </div>
</div>

</body>
</html>
```

Conserver les fenêtres en mémoire via l'attribut data-dom-cache

Dans l'exemple précédent, vous remarquerez que si vous naviguez entre les deux fenêtres, la seconde fenêtre s'affiche en provoquant l'apparition d'un message d'attente (*Loading*). Cela signifie que jQuery Mobile interroge le serveur via Ajax pour récupérer le code HTML de cette fenêtre. Cet appel Ajax est effectué avant l'affichage de la seconde fenêtre, et chaque fois que cette fenêtre sera affichée. La raison est que la seconde fenêtre est supprimée de l'arborescence du DOM lorsqu'elle devient cachée, ce qui permet à jQuery Mobile d'optimiser la place en mémoire. Un appel Ajax est alors nécessaire pour réafficher ultérieurement la fenêtre, étant donné que la fenêtre a été supprimée de la mémoire.

Il est possible d'indiquer à jQuery Mobile de n'effectuer le chargement de la fenêtre qu'une seule fois (la première), puis de conserver cette fenêtre constamment dans l'arborescence du DOM. Un seul appel Ajax sera alors effectué (le premier), permettant ainsi l'affichage plus rapide de la seconde fenêtre lors des affichages suivants. On utilise pour cela l'attribut `data-dom-cache=true` positionné sur la fenêtre que l'on désire conserver en mémoire.

La seconde fenêtre peut alors être codée de la façon suivante (le code de la première fenêtre n'est pas modifié).

Conserver la fenêtre en mémoire avec l'attribut data-dom-cache=true (fichier index2.html)

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv=Content-Type content=text/html;charset=iso-8859-1 />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win2 data-add-back-btn=true data-dom-cache=true>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la Fenêtre 2 </p>
  </div>
</div>

</body>
</html>
```

> **REMARQUE** **Appel Ajax**
>
> L'attribut `data-dom-cache` ne s'utilise que pour des fenêtres situées dans des fichiers externes, c'est-à-dire récupérées par Ajax. Lorsque les deux fenêtres sont situées dans le même fichier HTML, aucun appel Ajax n'est effectué par jQuery Mobile, donc les fenêtres restent en mémoire dans tous les cas.

Anticiper le chargement des fenêtres via l'attribut data-prefetch

Reprenons l'exemple précédent concernant les deux fenêtres situées dans des fichiers différents. Il est possible d'effectuer le chargement de la seconde fenêtre en tâche de fond, sans attendre d'avoir cliqué sur le lien permettant de l'afficher. Il suffit pour cela d'indiquer l'attribut `data-prefetch` (sans valeur associée) dans le lien `<a>` contenant la référence à cette fenêtre. Lors du chargement de la première fenêtre (celle contenant le lien), jQuery Mobile effectue une recherche de tous les liens contenant l'attribut `data-prefetch` et charge en tâche de fond les fenêtres correspondantes. Ces fenêtres seront alors immédiatement disponibles lorsqu'on cliquera sur le lien associé.

Code HTML de la première fenêtre

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=home>
  <div data-role=header>
    <h1>Home</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la Fenêtre 1 </p>
    <a href=index2.html data-prefetch>
          Aller dans la fenêtre située dans index2.html </a>
  </div>
</div>

</body>
</html>
```

L'utilisation de l'attribut `data-prefetch` permet de charger automatiquement la fenêtre située dans `index2.html` sans attendre un clic sur le lien.

Code HTML de la seconde fenêtre (chargée automatiquement en mémoire lors de l'affichage de la première fenêtre)

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv=Content-Type content=text/html;charset=iso-8859-1 />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win2 data-add-back-btn=true>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
```

```
  </div>

  <div data-role=content>
    <p> Contenu de la Fenêtre 2 </p>
  </div>
</div>

</body>
</html>
```

Nous constatons maintenant que le clic sur le lien affichant la seconde fenêtre ne provoque plus l'affichage du message d'attente (*Loading*). La seconde fenêtre est affichée directement, suite au clic sur le lien.

Transitions entre les fenêtres

La transition entre les deux fenêtres s'effectue pour l'instant par un glissement de la droite vers la gauche, la deuxième fenêtre poussant la première vers la gauche au fur et à mesure de son apparition. jQuery Mobile permet de modifier l'effet produit lors de la transition entre les deux fenêtres.

Utilisons par exemple une transition `flip`, consistant en un effet de rotation horizontale entre les deux fenêtres. Il suffit pour cela d'ajouter l'attribut `data-transition="flip"` dans le code HTML du lien vers la seconde fenêtre.

Utiliser une transition flip entre les deux fenêtres

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win1>
  <div data-role=header>
    <h1>Fenêtre 1</h1>
  </div>
```

```
  <div data-role=content>
    <p> Contenu de la fenêtre 1</p>
    <a href=#win2 data-transition=flip> Aller sur la fenêtre 2 </a>
  </div>
</div>

<div data-role=page id=win2 data-add-back-btn=true>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 2</p>
  </div>
</div>

</body>
</html>
```

Lors du clic sur le lien, la transition entre les deux fenêtres s'effectue maintenant par rotation. Lors de l'appui sur le bouton *Back*, la transition produit le même effet, mais en sens inverse.

Voici la liste des transitions possibles entre deux fenêtres, utilisables au moyen de l'attribut `data-transition`.

Tableau 2–1 Valeurs possibles de l'attribut data-transition

data-transition	Signification
`slide`	On passe d'une fenêtre à l'autre par un déplacement horizontal de la droite vers la gauche. C'est la valeur par défaut.
`slideup`	La seconde fenêtre apparaît par le bas, en recouvrant progressivement la première.
`slidedown`	La seconde fenêtre apparaît par le haut, en recouvrant progressivement la première.
`pop`	La seconde fenêtre apparaît par le centre de la première, en s'élargissant jusqu'à la recouvrir.
`fade`	La première fenêtre disparaît grâce à une diminution de son opacité (de 1 vers 0), tandis que la seconde apparaît grâce une augmentation de son opacité (de 0 vers 1).
`flip`	La seconde fenêtre apparaît par un effet de rotation sur un axe vertical, faisant disparaître ainsi la première fenêtre.
`none`	Pas de transition entre les deux fenêtres. La seconde fenêtre s'affiche imédiatement.

Fenêtres superposées

Les transitions précédentes permettent de passer d'une fenêtre à l'autre : la deuxième fenêtre remplace la précédente à l'affichage et le retour à la fenêtre précédente s'effectue par le bouton *Back* situé dans la barre de titre.

jQuery Mobile a prévu la possibilité d'afficher une fenêtre par-dessus l'autre, au lieu de la remplacer à l'écran. On a donc des fenêtres superposées, par exemple une boîte de dialogue qui s'ouvre par-dessus une fenêtre affichée. Deux façons de procéder sont proposées par jQuery Mobile :

- soit on indique dans l'attribut du lien que l'on désire ouvrir une nouvelle fenêtre qui se superposera à la précédente ;
- soit on indique dans les attributs de la nouvelle fenêtre que c'est une fenêtre superposée, et chaque visualisation de cette fenêtre l'affichera comme telle.

Nous examinons ci-après ces deux possibilités.

La fenêtre superposée est définie par les attributs du lien

Cela correspond à la première façon de créer une fenêtre superposée. Pour la créer (au lieu d'une simple fenêtre), il suffit d'indiquer dans le lien du bouton l'attribut `data-rel="dialog"`. Le clic sur le lien ouvrira une nouvelle fenêtre (comme pour tous les liens), mais cette fenêtre se superposera au-dessus de la précédente, sans la faire disparaître (figure 2-7, page suivante).

Afficher une fenêtre superposée par l'attribut du lien

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win1>
  <div data-role=header>
    <h1>Fenêtre 1</h1>
  </div>
```

```
  <div data-role=content>
    <p> Contenu de la fenêtre 1</p>
    <a href=#win2 data-rel=dialog data-transition=pop>
          Aller sur la fenêtre 2 </a>
  </div>
</div>

<div data-role=page id=win2>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 2</p>
  </div>
</div>

</body>
</html>
```

Figure 2–7
Une fenêtre superposée

jQuery Mobile a automatiquement ajouté, dans la barre de titre de la fenêtre superposée, un bouton de fermeture symbolisé par une croix (voir figure 2-7). Un clic sur ce bouton permet de fermer cette fenêtre et de revenir à la précédente (située sous celle-ci).

> **REMARQUE** **Bouton Back**
>
> Il est possible d'insérer soi-même un bouton de fermeture dans le contenu de la fenêtre. Pour cela, il suffit de simuler le bouton *Back* (voir chapitre 7, « Simuler le bouton Back »).

La fenêtre superposée est définie par ses propres attributs

Plutôt que d'indiquer dans le lien que l'on souhaite ouvrir une fenêtre superposée, on indique dans les attributs de la fenêtre qu'il s'agit d'une fenêtre superposée. L'ouverture de cette fenêtre s'effectuera donc toujours au-dessus de la précédente.

Pour cela, on indique dans les attributs de la fenêtre `data-role="dialog"` (au lieu de `data-role="page"`). L'exemple précédent devient :

Afficher une fenêtre superposée au moyen de l'attribut de la fenêtre

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win1>
  <div data-role=header>
    <h1>Fenêtre 1</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 1</p>
    <a href=#win2 data-transition=pop>
          Aller sur la fenêtre 2 </a>
  </div>
</div>

<div data-role=dialog id=win2>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
  </div>
```

```
  <div data-role=content>
    <p> Contenu de la fenêtre 2</p>
  </div>
</div>

</body>
</html>
```

Le résultat est le même que celui de l'exemple précédent.

Utiliser les thèmes CSS

jQuery Mobile permet d'utiliser des styles différents pour chaque fenêtre créée dans les pages HTML. Par défaut, un style est déjà appliqué à celles-ci, mais il est très simple d'en utiliser un autre.

Les styles CSS définis par jQuery Mobile correspondent aux lettres `a, b, c d` et `e`. Ces styles sont définis dans le fichier `jquery.mobile.css`, en tête du fichier. Voici par exemple un extrait du fichier concernant les définitions de styles associées au thème `"a"` :

Définitions du thème "a" dans le fichier jquery.mobile.css

```
/* A
-----------------------------------------------------------------------------
----------------------------*/

.ui-bar-a {
  border: 1px solid #2A2A2A;
  background:        #111111;
  color:             #ffffff;
  font-weight: bold;
  text-shadow: 0 -1px 1px #000000;
  background-image: -webkit-gradient(linear, left top, left bottom,
          ➥ from(#3c3c3c), to(#111)); /* Saf4+, Chrome */
  background-image: -webkit-linear-gradient(top, #3c3c3c, #111); /* Chrome 10+,
          ➥ Saf5.1+ */
  background-image: -moz-linear-gradient(top, #3c3c3c, #111); /* FF3.6 */
  background-image:  -ms-linear-gradient(top, #3c3c3c, #111); /* IE10 */
  background-image:   -o-linear-gradient(top, #3c3c3c, #111); /* Opera 11.10+ */
  background-image:      linear-gradient(top, #3c3c3c, #111);
}
.ui-bar-a,
.ui-bar-a input,
```

```
.ui-bar-a select,
.ui-bar-a textarea,
.ui-bar-a button {
    font-family: Helvetica, Arial, sans-serif;
}
.ui-bar-a .ui-link-inherit {
  color:              #fff;
}
.ui-bar-a .ui-link {
  color:              #7cc4e7;
  font-weight: bold;
}
.ui-body-a {
  border: 1px solid   #2A2A2A;
  background:         #222222;
  color:              #fff;
  text-shadow: 0 1px 0 #000;
  font-weight: normal;
  background-image: -webkit-gradient(linear, left top, left bottom,
          ➥ from(#666), to(#222)); /* Saf4+, Chrome */
    background-image: -webkit-linear-gradient(top, #666, #222); /* Chrome 10+,
          ➥ Saf5.1+ */
  background-image: -moz-linear-gradient(top, #666, #222); /* FF3.6 */
  background-image:  -ms-linear-gradient(top, #666, #222); /* IE10 */
  background-image:   -o-linear-gradient(top, #666, #222); /* Opera 11.10+ */
  background-image:      linear-gradient(top, #666, #222);
}
.ui-body-a,
.ui-body-a input,
.ui-body-a select,
.ui-body-a textarea,
.ui-body-a button {
    font-family: Helvetica, Arial, sans-serif;
}
.ui-body-a .ui-link-inherit {
  color:              #fff;
}
.ui-body-a .ui-link {
  color:              #2489CE;
  font-weight: bold;
}
.ui-br {
  border-bottom: rgb(130,130,130);
  border-bottom: rgba(130,130,130,.3);
  border-bottom-width: 1px;
vborder-bottom-style: solid;
}
.ui-btn-up-a {
  border: 1px solid     #222;
  background:           #333333;
```

```
  font-weight: bold;
  color:                #fff;
  text-shadow: 0 -1px 1px #000;
  background-image: -webkit-gradient(linear, left top, left bottom,
          ➥ from(#555), to(#333)); /* Saf4+, Chrome */
    background-image: -webkit-linear-gradient(top, #555, #333); /* Chrome 10+,
          ➥ Saf5.1+ */
  background-image: -moz-linear-gradient(top, #555, #333); /* FF3.6 */
  background-image:  -ms-linear-gradient(top, #555, #333); /* IE10 */
  background-image:   -o-linear-gradient(top, #555, #333); /* Opera 11.10+ */
  background-image:      linear-gradient(top, #555, #333);
}
.ui-btn-up-a a.ui-link-inherit {
  color:                #fff;
}
.ui-btn-hover-a {
  border: 1px solid     #000;
  background:           #444444;
  font-weight: bold;
  color:                #fff;
  text-shadow: 0 -1px 1px #000;
  background-image: -webkit-gradient(linear, left top, left bottom,
          ➥ from(#666), to(#444)); /* Saf4+, Chrome */
    background-image: -webkit-linear-gradient(top, #666, #444); /* Chrome 10+,
          ➥ Saf5.1+ */
  background-image: -moz-linear-gradient(top, #666, #444); /* FF3.6 */
  background-image:  -ms-linear-gradient(top, #666, #444); /* IE10 */
  background-image:   -o-linear-gradient(top, #666, #444); /* Opera 11.10+ */
  background-image:      linear-gradient(top, #666, #444);
}
.ui-btn-hover-a a.ui-link-inherit {
  color:                 #fff;
}
.ui-btn-down-a {
  border: 1px solid     #000;
  background:           #3d3d3d;
  font-weight: bold;
  color:                #fff;
vtext-shadow: 0 -1px 1px #000;
  background-image: -webkit-gradient(linear, left top, left bottom,
          ➥ from(#333), to(#5a5a5a)); /* Saf4+, Chrome */
  background-image: -webkit-linear-gradient(top, #333, #5a5a5a); /* Chrome 10+,
          ➥ Saf5.1+ */
  background-image: -moz-linear-gradient(top, #333, #5a5a5a); /* FF3.6 */
  background-image:  -ms-linear-gradient(top, #333, #5a5a5a); /* IE10 */
  background-image:   -o-linear-gradient(top, #333, #5a5a5a); /* Opera 11.10+ */
  background-image:      linear-gradient(top, #333, #5a5a5a);
}
```

```
.ui-btn-down-a a.ui-link-inherit {
  color:                #fff;
}
.ui-btn-up-a,
.ui-btn-hover-a,
.ui-btn-down-a {
  font-family: Helvetica, Arial, sans-serif;
  text-decoration: none;
}
```

La classe CSS `ui-bar-a` définit par exemple le style de la barre de titre dans le thème `"a"`, tandis que `ui-body-a` définit le style du contenu de la fenêtre dans ce même thème. Les autres classes CSS utilisées permettent de styler chacun des éléments de la page HTML selon ce thème (principalement, les différents états des boutons : `up`, `down`, `hover`).

Indiquer un nouveau thème pour une fenêtre

jQuery Mobile permet d'associer un thème indépendant à chacun des composants de la fenêtre (barre de titre, barre de bas de page et contenu de la fenêtre). Il fournit pour cela les attributs `data-theme`, `data-header-theme`, `data-footer-theme` et `data-content-theme.` Chacun de ces attributs peut posséder l'un des thèmes définis par jQuery Mobile (`"a", "b", … "e"`) ou un thème défini par notre application (voir section suivante).

Tableau 2–2 Utilisation des attributs associés aux thèmes

Attributs	Signification
`data-theme`	Cet attribut peut s'utiliser pour tous les éléments d'une fenêtre (y compris l'élément `<div>` associé à la fenêtre). Si on l'applique sur le `<div>` possédant l'attribut `data-role="page"` : le thème permet de définir alors le contenu de la fenêtre. Par défaut, le contenu de la fenêtre utilise le thème `"c"`. Si on l'applique sur le `<div>` possédant l'attribut `data-role="header"` : le thème permet de définir alors la barre de titre. Par défaut, la barre de titre de la fenêtre utilise le thème `"a"`. Si on l'applique sur le `<div>` possédant l'attribut `data-role="footer"` : le thème permet de définir alors la barre du bas de page. Par défaut, la barre du bas de page de la fenêtre utilise le thème `"a"`. Si on l'applique sur le `<div>` possédant l'attribut `data-role="content"` : le thème permet de définir alors le contenu de la fenêtre. Par défaut, le contenu de la fenêtre utilise le thème associé à la fenêtre.
`data-header-theme`	Cet attribut s'utilise sur l'élément `<div>` associé à la fenêtre (`data-role="page"`). Il permet de définir le thème de la barre de titre (par défaut, le thème `"a"`).

Tableau 2–2 Utilisation des attributs associés aux thèmes (suite)

Attributs	Signification
`data-footer-theme`	Cet attribut s'utilise sur l'élément `<div>` associé à la fenêtre `(data-role="page")`. Il permet de définir le thème de la barre du bas de page (par défaut, le thème `"a"`).
`data-content-theme`	Cet attribut s'utilise sur l'élément `<div>` associé à la fenêtre `(data-role="page")`. Il permet de définir le thème du contenu de la fenêtre (par défaut, le thème associé à la fenêtre).

Comme le thème d'un élément de la fenêtre peut être défini à plusieurs endroits (par exemple, le thème de la barre de titre peut se définir par l'attribut `data-theme` utilisé sur celle-ci ou par l'attribut `data-header-theme` utilisé sur la fenêtre), jQuery Mobile a défini un ordre de priorité pour la prise en compte du thème final.

Pour la barre de titre *(header)*, les attributs suivants sont pris en compte par ordre de priorité *décroissante* :

1. attribut `data-theme` défini sur la barre de titre ;
2. attribut `data-header-theme` défini sur la fenêtre. Si cet attribut n'est pas défini dans le code HTML, il vaut `"a"` par défaut ;
3. attribut `data-theme` défini sur la fenêtre. Si cet attribut n'est pas défini dans le code HTML, il vaut `"c"` par défaut. Pour que la valeur de cet attribut soit prise en compte, il faut que l'attribut `data-header-theme` soit défini à `""`.

Pour la barre du bas de page *(footer)*, les attributs suivants sont pris en compte par ordre de priorité *décroissante* :

1. attribut `data-theme` défini sur la barre de bas de page ;
2. attribut `data-footer-theme` défini sur la fenêtre. Si cet attribut n'est pas défini dans le code HTML, il vaut `"a"` par défaut ;
3. attribut `data-theme` défini sur la fenêtre. Si cet attribut n'est pas défini dans le code HTML, il vaut `"c"` par défaut. Pour que la valeur de cet attribut soit prise en compte, il faut que l'attribut `data-footer-theme` soit défini à `""`.

Pour le contenu de la fenêtre, les attributs suivants sont pris en compte par ordre de priorité *décroissante* :

1. attribut `data-theme` défini sur le contenu de la fenêtre ;
2. attribut `data-theme` défini sur la fenêtre. Si cet attribut n'est pas défini dans le code HTML, il vaut `"c"` par défaut ;
3. attribut `data-content-theme` défini sur la fenêtre. La valeur par défaut est `""` (pas de valeur définie).

> **À RETENIR L'attribut data-theme est prioritaire**
>
> On retiendra que l'attribut `data-theme` défini sur un élément de la fenêtre est prioritaire sur les autres attributs définis sur les autres éléments. L'attribut `data-theme` sera celui que nous utiliserons principalement.

Pour voir l'impact de l'utilisation de cet attribut dans nos pages HTML, définissons quelques fenêtres utilisant chacune un des thèmes, et s'enchaînant les unes aux autres au moyen d'un lien `<a>` dans chacune de celles-ci.

Utiliser un thème différent dans chaque fenêtre

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win>                <!-- thème par défaut -->
  <div data-role=header>
    <h1>Défaut</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre</p>
    <a href=#wina> Aller sur le thème a </a>
  </div>
</div>

<div data-role=page id=wina data-add-back-btn=true>
  <div data-role=header data-theme=a>
    <h1>Thème a</h1>
  </div>

  <div data-role=content data-theme=a>
    <p> Contenu de la fenêtre</p>
    <a href=#winb> Aller sur le thème b </a>
  </div>
</div>

<div data-role=page id=winb data-add-back-btn=true>
  <div data-role=header data-theme=b>
    <h1>Thème b</h1>
  </div>
```

```
  <div data-role=content data-theme=b>
    <p> Contenu de la fenêtre</p>
    <a href=#winc> Aller sur le thème c </a>
  </div>
</div>

<div data-role=page id=winc data-add-back-btn=true>
  <div data-role=header data-theme=c>
    <h1>Thème c</h1>
  </div>

  <div data-role=content data-theme=c>
    <p> Contenu de la fenêtre</p>
    <a href=#wind> Aller sur le thème d </a>
  </div>
</div>

<div data-role=page id=wind data-add-back-btn=true>
  <div data-role=header data-theme=d>
    <h1>Thème d</h1>
  </div>

  <div data-role=content data-theme=d>
    <p> Contenu de la fenêtre</p>
    <a href=#wine> Aller sur le thème e </a>
  </div>
</div>

<div data-role=page id=wine data-add-back-btn=true>
  <div data-role=header data-theme=e>
    <h1>Thème e</h1>
  </div>

  <div data-role=content data-theme=e>
    <p> Contenu de la fenêtre</p>
    <p> Fin des thèmes</p>
  </div>
</div>

</body>
</html>
```

Enchaînons chacune des fenêtres de la page HTML afin de visualiser les thèmes correspondants (figures 2-8 à 2-13).

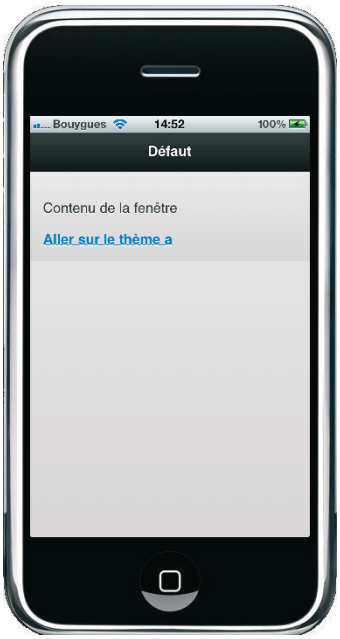


Figure 2–8 Thème par défaut

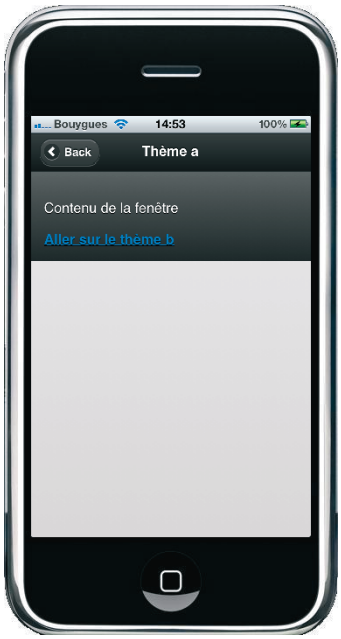


Figure 2–9 Thème "a"

Figure 2–10 Thème "b"

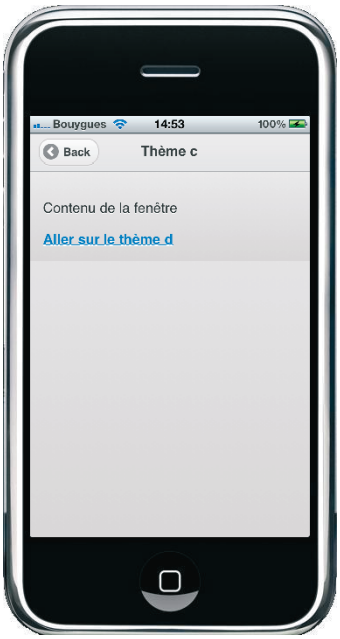


Figure 2–11 Thème "c"

Figure 2–12 Thème "d"

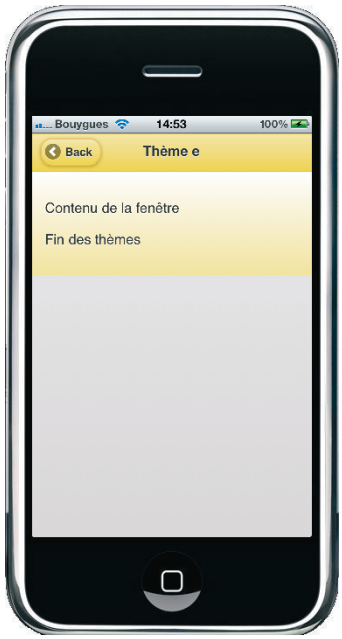


Figure 2–13 Thème "e"

Créer ses propres thèmes

jQuery Mobile a créé une application permettant de créer ses propres thèmes. Elle est accessible à l'URL http://jquerymobile.com/themeroller. Elle permet de créer de nouveaux thèmes en vous construisant un fichier de thèmes CSS correspondant aux choix que vous aurez effectués dans l'application. Ce fichier devra ensuite être inclus dans votre page HTML.

Toutefois, en respectant les conventions de jQuery Mobile concernant l'écriture des styles CSS, nous pouvons écrire nous-même nos propres thèmes. Par exemple, écrivons le thème `"z"`, inexistant à ce jour.

Création et utilisation du thème "z"

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>

  <style type=text/css>
    .ui-bar-z {
      color : red;
      background-color : gainsboro;
    }
    .ui-body-z {
      font-style : italic;
      font-family : comic sans MS;
    }
  </style>
</head>

<body>

<div data-role=page id=win>
  <div data-role=header data-theme=z>
    <h1>Thème z</h1>
  </div>

  <div data-role=content data-theme=z>
    <p> Le nouveau thème "z" est utilisé dans cette fenêtre !</p>
  </div>
</div>

</body>
</html>
```

Nous définissons ici le thème `"z"` de façon rudimentaire, en définissant uniquement l'aspect de la barre de titre et le contenu de la fenêtre, via respectivement les classes CSS `ui-bar-z` et `ui-body-z`. La fenêtre aura maintenant l'aspect suivant :

Figure 2–14
Application d'un nouveau thème

REMARQUE Nommage des thèmes

jQuery Mobile conseille d'utiliser une seule lettre pour indiquer le nom du thème (`"a"`, `"b"`, … `"z"`).

À LIRE Feuilles de styles CSS

Si vous souhaitez maîtriser les CSS, l'ouvrage suivant vous en livre toutes les subtilités.

📖 R. Goetter, *CSS avancées : Vers HTML 5 et CSS 3*, Eyrolles, 2e édition, 2012

3

Afficher des listes

Nous avons décrit dans le précédent chapitre la structure minimale de chacune de nos fenêtres. Ce chapitre permet d'étudier comment y ajouter un contenu plus élaboré, en particulier des listes. Les listes sont en effet un élément de base pour l'affichage sur les terminaux mobiles, du fait de la taille réduite de ces derniers (en particulier pour les smartphones).

Afficher une liste simple

Pour afficher une liste d'éléments, on utilise les balises HTML `<ul>` ou `<ol>`, contenant les éléments de liste `<li>`. L'attribut `data-role="listview"` spécifié dans la balise `<ul>` ou `<ol>` permet de styler la liste selon les conventions définies dans les styles jQuery Mobile.

Une liste d'éléments affichée selon les conventions jQuery Mobile (figure 3-1)

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
```

```
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=win1>
  <div data-role=header>
    <h1>Fenêtre 1</h1>
  </div>

  <div data-role=content>
    <h3> Liste d'éléments</h3>
    <ul data-role=listview>
      <li> Elément 1 </li>
      <li> Elément 2 </li>
      <li> Elément 3 </li>
      <li> Elément 4 </li>
      <li> Elément 5 </li>
    </ul>
  </div>
</div>

</body>
</html>
```

Figure 3–1 Une liste d'éléments

Figure 3–2 Une liste d'éléments sans styles

Si l'attribut `data-role="listview"` n'est pas indiqué, la liste s'affiche, mais ne profite pas des styles CSS définis par jQuery Mobile (figure 3-2).

Ajouter des liens

Supposons que l'on désire afficher une liste de menus pour notre application. Chaque élément de liste est un lien vers une autre fenêtre (situé dans la même page HTML ou non). On écrit alors :

Une liste contenant des liens vers d'autres fenêtres

```
<!DOCTYPE html>
<html>
<head>
  <meta name=viewport content="user-scalable=no,width=device-width" />
  <link rel=stylesheet href=jquery.mobile/jquery.mobile.css />
  <script src=jquery.js></script>
  <script src=jquery.mobile/jquery.mobile.js></script>
</head>

<body>

<div data-role=page id=home>
  <div data-role=header>
    <h1>Home</h1>
  </div>

  <div data-role=content>
    <h3> Menu principal</h3>
    <ul data-role=listview>
      <li><a href=#win1> Fenêtre 1 </a></li>
      <li><a href=#win2> Fenêtre 2 </a></li>
    </ul>
  </div>
</div>

<div data-role=page id=win1 data-add-back-btn=true>
  <div data-role=header>
    <h1>Fenêtre 1</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 1 </p>
  </div>
</div>
```

```
<div data-role=page id=win2 data-add-back-btn=true>
  <div data-role=header>
    <h1>Fenêtre 2</h1>
  </div>

  <div data-role=content>
    <p> Contenu de la fenêtre 2 </p>
  </div>
</div>

</body>
</html>
```

Les liens vers les autres fenêtres sont indiqués dans chacune des balises `<li>`. La fenêtre s'affiche comme sur la figure 3-3.

Figure 3–3
Une liste avec des liens

REMARQUE **Affichage des éléments cliquables**

jQuery Mobile a augmenté la hauteur de chaque élément de liste (afin que l'on puisse plus facilement le sélectionner sur l'écran tactile) et a ajouté, à droite de chacun d'eux, une icône représentant une flèche pointant à droite (symbolisant le fait qu'une autre fenêtre apparaît si l'on clique sur l'élément de liste). Bien sûr, si l'on clique sur chacun des éléments de liste, la fenêtre correspondante apparaît.