Chapter

8

Mobile Web

CHAPTER AUTHORS

Jean Gauthier

Nirali Jivrajani

Romain Edelmann

Dai Yiqiang

CONTENTS

1 Introduction 5
2 8.2 Usability of touch-based interfaces 5
8.2.1 Why? 6
8.2.2 Differences? 6
a. No cursor 6
b. Screen size 6
c. Layout 7
d. Interactive elements 8
e. Navigation 8
f. Feedback 9
g. Gestures 9
3 Performance 9
8.3.1 Optimization of Contents 10
a. Content optimization 10
b. Use of offline browsing 10
c. Optimization of images 10
8.3.2 Optimize scripts 12
a. Minimise your code 12
b. Using HTML5 13
4 Introduction to JQuery Mobile 13
8.4.1 JQuery Mobile crash course 13
a. Basic Page 13
b. Navigation 16
8.4.2 More Jquery Mobile... 18
a. ThemeRoller 18
b. Multiple pages in single HTML file 19
c. JQuery Library in JQuery Mobile 20
8.4.3 Summary 20
5 References: 20

1 INTRODUCTION

Smartphone penetration has grown exponentially in the last couple of years. And along with this the growth in mobile web users has also increased as more and more users use 3G and Wi-fi on their mobile phones. Based on the latest statistics for active mobile-broadband subscriptions worldwide, there ***1.2 billion*** mobile Web users worldwide, many of whom are mobile-only and do not use other means to access the Web. With the growing markets of China and India leading the smartphone penetration, this trend will only grow in the next few years.

In the last year, mobile searches have quadrupled according to Google. Despite the growing use of mobile web, 79 percent of large online advertisers still do not have a mobile optimized site[1]. "One in seven searches is now mobile." According to Google.

"Your customer is trying to engage you… it would be like not doing business with your customers on Thursdays." Jason Spero, Google (Feb 2011). To keep up with these changing trends it is very important to have a mobile-optimised website especially if you know that many of your users access your site through phones.

Due to the inherent differences between a smartphone and a desktop, a mobile optimized website offers its own challenges. The smaller screen size, touch-based interaction and the mobile nature of the users as well who're generally on the go mean a very different look and feel for your website. Retaining a mobile user of your site means exploiting the various opportunities a mobile device offers while simultaneously dealing with its various constraints to provide a great user experience.

In this chapter, we will discuss some ways to optimize this user experience in terms of the user interface and functionality. In addition, we will also provide a short introduction to JQuery Mobile and how it could be used to optimize your website for a mobile user.

2 8.2 USABILITY OF TOUCH-BASED INTERFACES

Usability is very often seen as something trivial, dictated by common sense . But, as you browse the mobile web, you will see that this is very often not the case. While the techniques covered in this section might seem natural to most people, only a few apply them correctly. This is especially the case for mobile and touch-based interfaces, as a few efforts must be made to ensure a good user experience on those platforms.

These are a few examples of websites that are not optimised for mobile platforms.

[1] http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats#mobilebehavior

Fig. 8.2.a

Fig. 8.2.b

Fig. 8.2.c

8.2.1 Why?

The user interactions in a touch-based environment are fundamentally different from the ones in a desktop environment. These differences must be taken into account when designing a mobile web site, otherwise the user experience of the website or application can be quite painful and you could lose viewers[2]. As we will see throughout this section, touch-based interfaces can also bring a lot of opportunities, not only to keep the same level of usability, but also improve it!

8.2.2 Differences?

a. No cursor

The most obvious difference between touch-based and conventional interfaces is the lack of cursor. This comes with many implications. First of all, the cursor is much more precise and small than a finger, and one must take that into consideration when designing a touch-based interface. It is widely observed that users make more mistakes in a touch-based environment compared with a conventional setup. We will see later how to overcome this problem using judicious interactive elements and feedback mechanism.

Using fingers instead of a mouse has its own advantages. To begin with, interactions with a touch-based interface are more intuitive and feel natural. Objects can be manipulated as if they were real. Users can swing a finger across the screen to browse through the pages of a virtual book as they would do with a real book. They can rotate, scale or move a picture across the screen with as little as two fingers in a very intuitive way, as we will later when we talk about gestures.

b. Screen size

Especially with Smartphones, as they must be able to fit in your pocket, the size of the screen is much smaller than what a desktop environment can offer. This means special thought has to be given to the arrangement of the content and images on the screen, picking just the right amount of content as well user friendly navigation unlike desktops which offer a lot more space and hence design options. However, on tablets, the screen size is much larger, giving opportunities for different layouts.

[2] Ergonomie des interfaces, 5th edition, J.F. Nogier, T. Bouillot, J. Leclerc, Édition Dunod, 2011

There are numerous techniques you could use to deal with this:

c. Layout

The same layout of a page on a desktop website cannot be directly ported to mobile device due to the differences seen so far. Below in Fig 8.2.2.a, you can see the news agency Reuters' website on desktop.

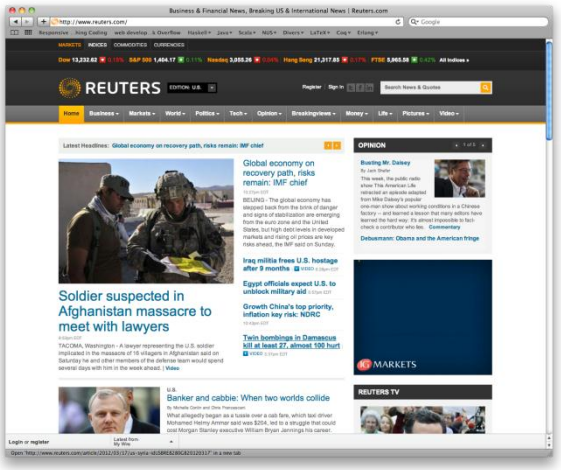


Fig. 8.2.2.a

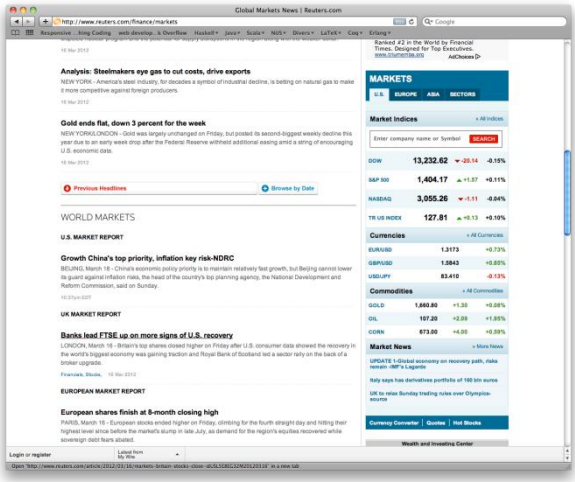


Fig. 8.2.2.b

On a Smartphone, the content is arranged on a single column to facilitate reading and selection. Below, the same website is shown accessed using a mobile. For instance in Fig. 8.2.2.d you could click on the last article for more information.

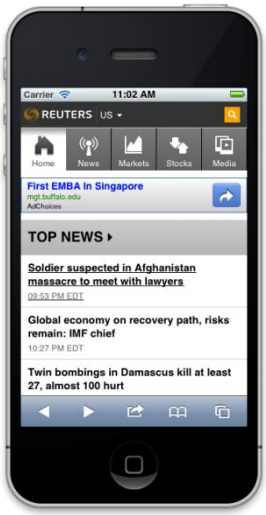


Fig. 8.2.2.c

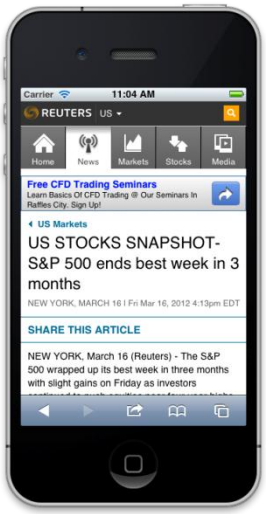


Fig. 8.2.2.d

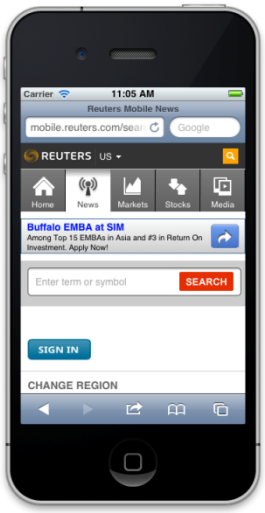


Fig. 8.2.2.e

On tablets, as the space available is more important, multi-columns layouts can be used easily, and sometimes, the desktop version is used when visiting with a tablet, as with Reuters for

instance. Regarding the layout of pages, this approach is not problematic, but the interactions are very different. This leads us to the interactive elements.

d. Interactive elements

Compared to a usual website, a touch-based website does make extensive use of clickable zones instead of small links to compensate for the imprecision of finger interactions. Instead of using a link at the end of a paragraph to give access to the complete text, one can turn the entire paragraph into an interactive element, as seen on various mobile versions of newspapers for instance (as on figures 8.2.2)[3].

The size of interactive elements is also important, for instance, the more frequently used an element is, the bigger the element must be. For very quick access button, a size of around 25mm in diameter is advised, and around 19mm for less frequently used.

The size of rarely used elements or elements that triggers actions that are not easily cancelled can be made much smaller, around 10mm, to force users to focus on the action. Moreover, confirmation of such an action should be asked, as it still could be an involuntary action.

e. Navigation

Navigation on mobile devices is fundamentally different from usual website navigation. The lack of the space and imprecision of selection would make traditional navigation impossible on those platforms. To overcome this problem, a new way of navigating the website must be found. We will see next the two main methods to have an efficient navigation on mobile.

Navigation list

Navigation on mobile devices is usually made using a single list of links that can be scrolled if too long to fit the screen. Links can lead to sub-lists of elements and so on, until a page is selected.

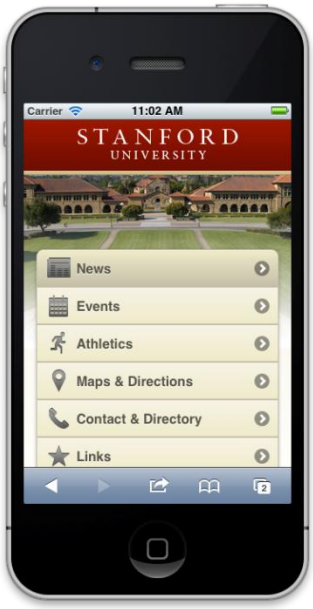


Fig. 8.2.2.e

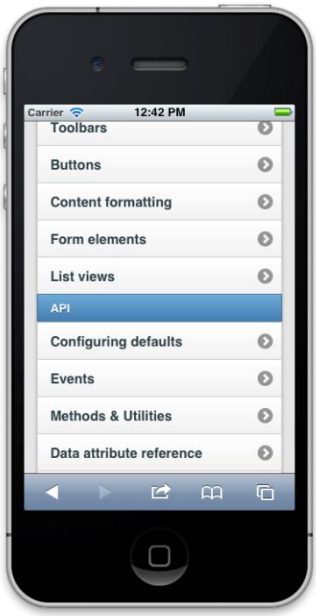


Fig. 8.2.2.f

[3] Ergonomie web, 2nd edition, A. Boucher, Groupe Eyrolles, 2009

Although this scheme is very popular, it is far from being the only one, as we will see next.

Navigation bar

Another scheme for navigation is to use a navigation bar, either at the top or bottom of the screen, as seen for example on the Reuters website. The problem with this method is that the number of item is limited by the screen size. Using an unambiguous icon and/or a short label per item, the maximal number of elements is limited to around five.

To get over this problem, one of these buttons can be used as a link to a complete menu, including links to less frequently used elements, while keeping the regularly used items on the navigation bar. The complete menu is very often implemented using the previous technique of having a single list of items.

Another solution to the limited number of elements in a navigation bar, is to make is horizontally scrollable, showing first the frequently used items and then the occasional ones. If using this technique, the fact that the navigation bar is scrollable must be made evident.

f. Feedback

As we have seen throughout this section, everything must be made to turn the user experience of a touch-based or mobile device into a nice experience. This implies also giving the user feedback for their actions so that they know that they have performed something, intended or not.

Visual feedback is very often used. This can be that the background of an element changes when clicked, or that a button is displayed as pushed for instance. Extra care must be taken when feedback is shown to the screen, so that it is not displayed under the finger of the user, where they cannot see it!

Mobile devices can also offer other kind of feedback, either using sound or vibrations. While they can be very effective, they must be sparsely used to keep their impact and not annoy your users.

g. Gestures

Touch-based platforms that support multi-touch offer the user the opportunity to use gestures to interact with the interface. These gestures can potentially multiply the interactivity and usability of your application or website.

However, gestures constitute a new language that users must learn. Some of those, mainly proposed by Apple, are considered standards. While other gestures might be useful in certain contexts, it is better to stick to the one most people know. Among those well known gestures are , “touch and hold” for contextual actions, “pinching” for zooming in and out, “dragging”, “swiping” and “rotating”.

You should always propose other solutions for people not familiar with gestures, while offering power users the ability to do things very quickly using them.

3 PERFORMANCE

The inherent difference between mobile browsers and desktop browsers render web pages looking different. Considering that mobile devices have restricted resources, various techniques could be applied to produce a user friendly design which is simple to use, fast and easy. Some of

these practices are best practices in normal web designs; however they are particularly useful to mobile website.

8.3.1 Optimization of Contents

a. Content optimization

Limited space on a mobile browser means the first step towards optimization would be content selection to ensure only the most important or relevant content is displayed. All categories of content which is unimportant for mobile users should be removed. Any additional information that is complex or not needed when they are solving a pressing problem should not be present on your site or should be summarized. This is to make sure your content is concise and straightforward. Minimizing the content allows the mobile website to load faster and be more user-oriented. Direct or advise them to use the desktop version if users need more information.

Desktop websites may have a lot advertisement banners, flash and big java applets, however these features are not mobile website friendly (e.g. there are a number of mobile browsers does not support flash), so their usage should be limited or even eliminated if possible.

While we are trying to give most relevant information to mobile users, we can also use server adaption to further improve the user experience. Our server side scripts will first gather the HTTP headers such as User-Agent header, Accept header and x-wap-profile header to determine the devices users are using to surf the mobile website, after that server scripts will render best content to return in response to the device's request. For example, Google is implementing this technique in its search page by displaying different optimized contents on iPhone, Blackberry and Nokia etc.

b. Use of offline browsing

Due to the resource constraints, we need to ensure that the mobile websites still function when the internet connectivity is a major issue. HTML5 comes with a handy ***"Cache Manifest"*** for creating offline web applications. The Cache Manifest is a text file that lists out all of the application resources that need to be cached in order for the given application to work without an internet connection. The files listed in the cache manifest get stored in the "Application Cache". The Application cache is like the browser's standard disk cache but much more robust. Once the browsers see the cache manifest file and get the user's approval to create an offline cache, the browser then proceeds to progressively download all cachable items in the background. As such, you can hit your homepage (for example) and cache the entire application. Once the application resources are cached, navigating around the application essential becomes a local operation, not a remote one. Cached resources are stored in the application cache, and the browser will never request it from the web until the cached item is de-cached or the cache manifest is invalidated.

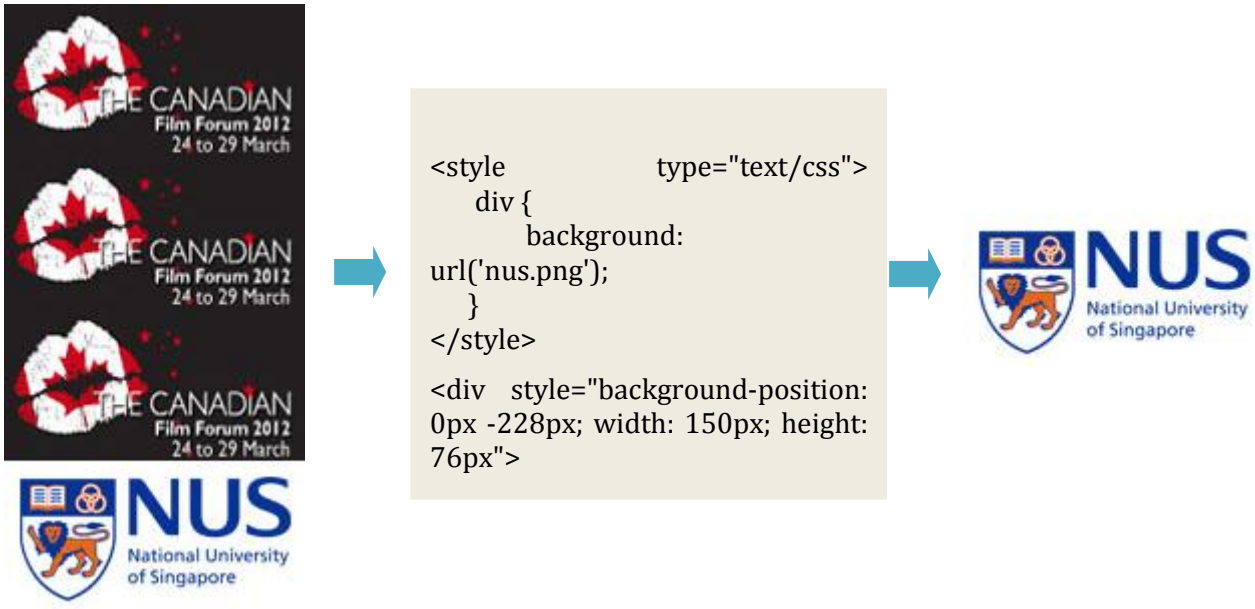
c. Optimization of images

Images are key components of a website. Reducing the image sizes and loading time will greatly improve the mobile websites. We have several ways to achieve this purpose.

CSS Sprites

Tenni Theurer from Yahoo! Exceptional Performance team mentioned that "popular web sites spend between 5% and 38% of the time downloading the HTML document. The other 62% to 95% of the time is spent making HTTP requests to fetch all the components in that HTML document such as images, scripts and stylesheets. The impact of having many components in the page is exacerbated by the fact that browsers download only two or four components in parallel per hostname, depending on the HTTP version of the response and the user's browser. Our experience shows that reducing the number of HTTP requests has the biggest impact on reducing response time and is often the easiest performance improvement to make." Thus a website with a lot request on images is a big problem for optimization.

CSS sprites allow you to create a single file that contains all the images laid out in a grid, meaning only a single image and only a single server call, with roughly the same file size because the empty space is compressed[4]. In that file, one can place all individual images that make up your interface and separated by enough space that they don't start overlapping. You can then set the background position (using negative values to move the background up) and include enough space around each image so that it only appears in the background of the element, effectively masking the rest of the sprite images. However an easier method to use sprites is also available by using tools like CSS Sprites generator offered by csssprites.com to create one as shown in Fig. 8.3.1.a.



CSS Sprites Images **CSS Code** **Result**

Fig. 8.3.1.a

[4] http://csssprites.com/

Use of Data URI

Instead of linking to an external image file when using an *<img>* element in HTML, or declaring a background-image in CSS, we can simply embed the image data directly into the document with data URIs. Data URIs are base 64 data lines of codes[5].

For instance,

```
<img src="
AAAFCAYAAACNbyblAAAAHElEQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAAO
```

An advantage of doing this is that HTTP requests and header traffic are not required for embedded data, so data URIs consume less bandwidth whenever the overhead of encoding the inline content as a data URI is smaller than the HTTP overhead. For example, the required base64 encoding for an image 400 bytes long would be 600 bytes, so if an HTTP request required more than 200 bytes of overhead, the data URI would be more efficient. In addition, for transferring many small files (less than a few kilobytes each), this can be much faster. TCP transfers tend to start slowly. If each file requires a new TCP connection, the transfer speed is limited by the round-trip time rather than the available bandwidth.

Use of image compressing tools

Besides using CSS Sprites and Data URIs, we can simply compress the image sizes. There are a number of image compression tools which allow you to do so. For example, Google WebP which is able to produce lossless images 28% smaller in size compared to PNGs and 25-34% smaller in size compared to JPEG images at equivalent SSIM index. Similarly Smush.it is an online service from Yahoo which helps you to compress your image. The main features of Smush.it include stripping meta data from JPEGs, optimizing JPEG compression, converting certain GIFs to indexed PNGs and stripping the unused colors from indexed images. All these features will significantly reduce the image sizes and make the images load faster.

8.3.2 Optimize scripts

a. Minimise your code

The lesser code, the smaller the amount of data transferred and faster the speed with which your web page will be loaded.

HTML is different from XHMTL. It allows the developer to omit certain tags. For example, if you are writing ***
List item
, you could just write *** List item instead, or instead of a paragraph that you would usually close with *** ***, you could simple omit it to write *** My paragraph***. Omitting optional tags reduce your html file size and make your codes leaner. Google developers implemented this technique on their web pages such as privacy centers and successfully saved 20% of the original file size. For a complete list of optional tags which you can omit you could refer to the various HTML DTD websites.

We know the slowest part of any mobile website is the network. Every http requests take an unpredictable amount of time. Thus we need to reduce the amount of files. We can combine

[5] http://css-tricks.com/data-uris/

javascripts into one single .js file and likewise for stylesheets. We can also configure our servers to use gzip to compress our files when our servers send the files.

b. Using HTML5

Mobile frameworks like jQuery Mobile and Sencha work best with HTML5. Besides these, HTML5 has many aspects which are most suitable with mobile websites. For example, HTML5 comes with local or session storage rather than relying on cookies. You can take advantage of these by storing some of your users' data instead of repeatedly requesting them from server. HTML5 also provide native in-browser validation of certain generic form types. You can test whether an element is a number in certain range, or conforms to a certain regex, or whether a required field is completed, etc. As these validations are down on client side, request to servers are reduced.

4 INTRODUCTION TO JQUERY MOBILE

JQueryMobile(JQM) is a free and open source JavaScript library which facilitates the client-side scripting of HTML and is specially optimized for touch-based devices. Its simple design allows easier coding of dynamic web pages as well as web and AJAX applications[6].

The JQM works as following:

- The basic HTML file, as coded by the web developer is downloaded from the web server.
- This implies that the JQM script and JQM CSS are downloaded to the client's device as well.
- The JQM script is executed on the client-side, the HTML file processed, allowing the JQM CSS to display the page in JQM's layout.

8.4.1 JQuery Mobile crash course

JQM's syntax is quite straightforward, since it essentially uses basic HTML, for the layout part at least.

The following example explains how to build the most important elements of a website besides content: navigation. The key observation is that the screen of smart phones is much smaller than the screen of a desktop computer, and the page layout has to be adapted to this.

a. Basic Page

Let's start with a basic JQM page (index.html). It should be noted that almost all the basic layout is done using the data-role attribute, which can take values such as e.g. page, header, content, navbar or button:

[6] http://jquerymobile.com/

```
<!DOCTYPE html>
<html>
<head>
<title>NUS - Home</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>
```

Fig. 8.4.1.a

In this code snippet we import JQM's CSS and JavaScript files. These files will allow us to use the HTML-like attributes of JQM inside HTML tags, and yet allow JQM take care of the layout. All the page content must be placed inside the tag. Simply use the same syntax and tags, as for any other HTML website.

Setting data-add-back-btn="true" allows us to add a back button to the site without relying on the mobile browser implementation for showing a back-button. The back-button is added inside the header, provided the user has already navigated through the website.

Page headers are added using :

```
<div data-role="page" data-add-back-btn="true">
        <div data-role="header" data-position="fixed">
                <h1>NUS - Home</h1>
        </div><!-- /header -->
</div><!-- /page -->
```

Fig. 8.4.1.b

In order to fill up the index page with specially formatted content, we introduce JQM's data-role="listview" attribute, which transforms HTML lists into touch-optimized lists. The size of the clickable area is increased, one list-item taking the whole screen width, thus making the user interaction less error-prone. As seen before, various attributes can be added to a data-role marked tag, in order to customize the layout:

```
<div data-role="content">
  <img src="images/b_campuslife_feb.jpg" alt="Campuslife"/>
  <ul data-role="listview" data-inset="true">
 <li>
    <a href="navigation.html">
   <img src="images/chemistry.gif" alt=""/>
   <h3>NUS scientists unravel 30-year-old chemical mystery</h3>
   <p>15 March 2012</p>
  </a>
</li>
                    ...
          </ul>
</div><!-- /content -->
```

Fig 8.4.1.c

b. Navigation

Now the navigation has to be taken care of. There is no way horizontal navigation menus can be used, unless they exactly fit to the screen's width: smart phone users seem to have a strong preference for vertical scrolling rather than for horizontal scrolling. So there remains only one solution, i.e. to use a vertical navigation menu. This menu should be placed at the bottom rather than the top of the page, because the user should be able to access the page content immediately.

There is one more constraint: the menu should be accessible immediately, from everywhere. So it is natural to use a fixed footer that contains the link to the menu. Using the header for this purpose would be a suboptimal solution; due to the way users hold a smart phone, which implies that the bottom of its screen is much more accessible. Note that the links will be displayed as buttons, because of the data-role="button" attribute. In order to have JQM doing the formatting, the menu links have to be placed inside a tag.

```
<div data-role="footer" data-position="fixed">
  <div data-role="navbar" data-iconpos="left">
    <ul>
      <li><a href="index.html" data-role="button" data-icon="home">Home</a></li>
      <li><a href="navigation.html" data-role="button" data-icon="search">Menu</a></li>
    </ul>
  </div>
</div><!-- footer -->
```

Fig. 8.4.1.d

With the already acquired knowledge, it is very straightforward to build a navigation menu (navigation.html). The only tricky part is that the sub-menus should be displayed on another page, while keeping them in the same HTML file, for convenience. This is done by nesting lists: one more level of lists is added inside an <li> tag:

```
<div data-role="content">
        <ul data-role="listview">
                <li>About NUS
                        <ul>
                                <li><a href="president.html">President's Welcome</a></li>
                                ...
                        </ul>
                </li>
                ...
        </ul>
</div><!-- /content -->
```

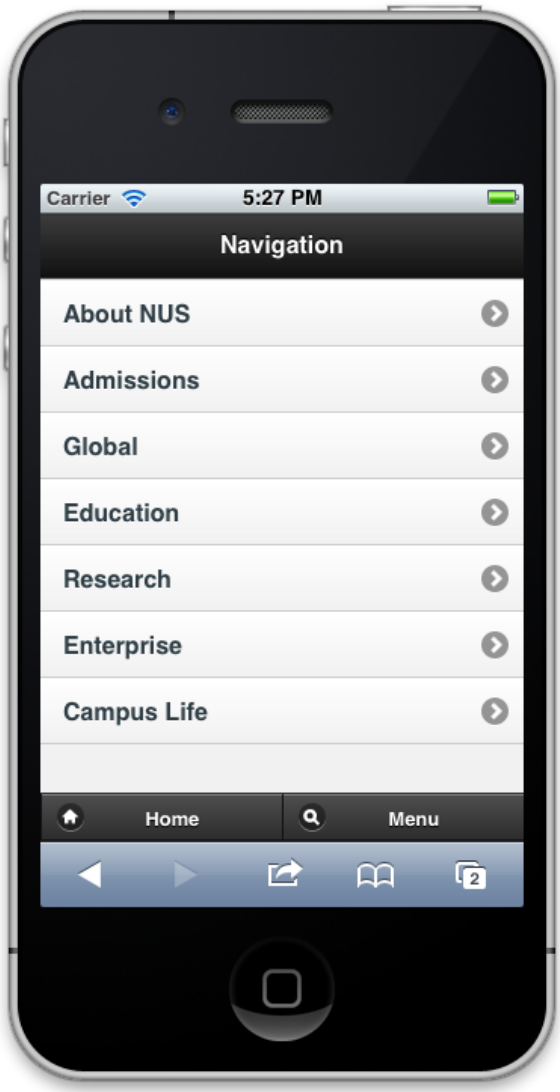


Fig. 8.4.1.e

8.4.2 More Jquery Mobile...

a. ThemeRoller

ThemeRoller is a useful tool that allows us to create custom themes for JQM. Three themes can be created simultaneously, by dragging and dropping colors from a color palette onto the wished items. Not only colors, but also text fonts, button sizes and styles and many other UI elements can be easily modified in this way.

The interface themes can be modified using the data-theme attribute, which can be applied to any tag, taking the values a, b, c, d or e:

```
<div data-role="..." data-theme="b">
```

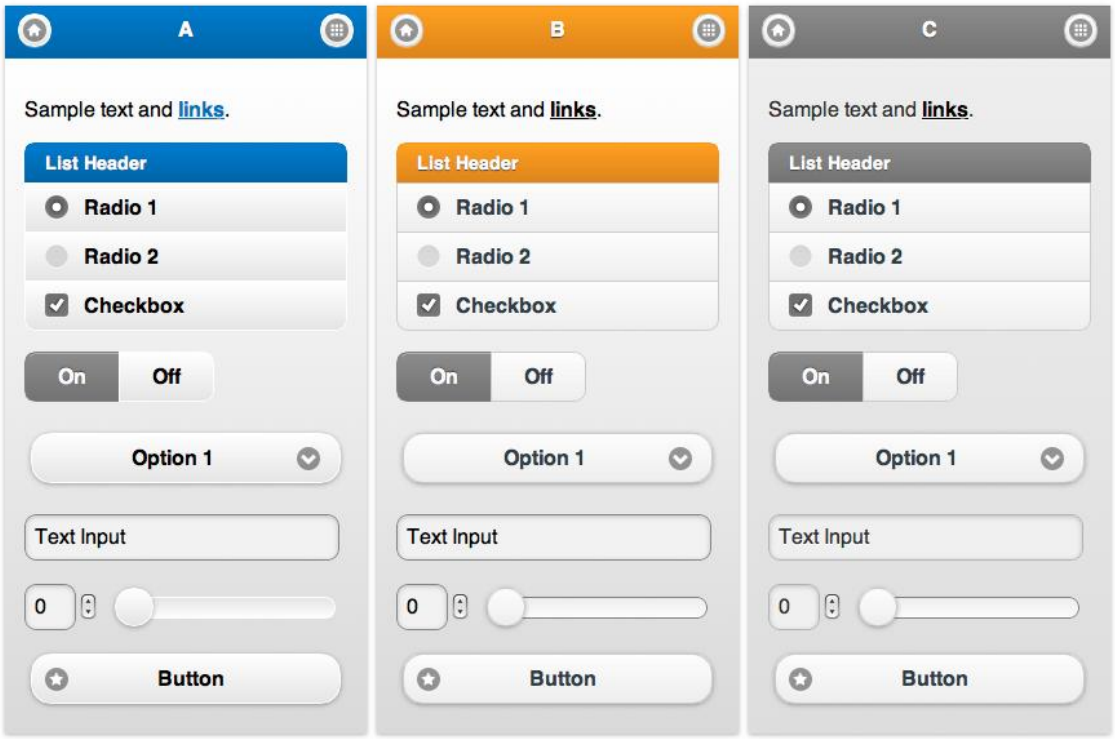


Fig. 8.4.2.a

Once the customized themes are set, the user simply downloads the modified CSS from http://jquerymobile.com/themeroller and includes it into his HTML pages. Note that the basic CSS file still has to be included.

b. Multiple pages in single HTML file

JQM furthermore allows us to put the content for several pages into one page. The next example shows how this is done. Note that in the standard behavior, only one page is shown at a time.

```
<!-- Start of first page: #page_one -->
<div data-role="page" id="page_one">

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

        <div data-role="content" >
        <p><a  href="#page_two"  data-role="button"  data-rel="dialog"  data-transition="pop">Page
                        Two</a></p>
        </div><!-- /content -->
</div><!-- /page_one -->

<!-- Start of second page: #page_two -->
<div data-role="page" id="page_two">
        <div data-role="content" >
```

```
                <p><a href="#page_one" data-role="button" data-rel="back">Back to Page One</a></p>
        </div><!-- /content -->
</div><!-- /page two -->
```

c. JQuery Library in JQuery Mobile

Since JQM is built on top of JQuery, the different functions of this library are usable as well. Knowing this, it is possible to add a button that scrolls the page back to top. The script can be inserted e.g. in the <head> tag, and the button in the footer:

```
<script type="text/javascript">
        $(function() {
                $('#go_top').click(
                function (e) {
                        $('html, body').animate({scrollTop: '0px'}, 800);
                });
        });
</script>
```

```
<li><a id="go_top" class="go_top" data-role="button" data-icon="arrow-u">Back to Top</a></li>
```

8.4.3 Summary

These examples perfectly illustrate JQuery's mantra "Write less, do more." The basic content formatting is very simple and intuitive, ultimately summarized by the data-role attribute that can be applied to diverse HTML tags. More information can be found on JQM's website http://jquerymobile.com/.

The interested reader can also check out http://jquery.com/ for further information about JQuery, on which JQM is built, and http://jqueryui.com/ which offers useful elements for building a web UI.

5 REFERENCES:

1. Ergonomie des interfaces, 5th edition, J.F. Nogier, T. Bouillot, J. Leclerc, Édition Dunod, 2011
2. Ergonomie web, 2nd edition, A. Boucher, Groupe Eyrolles, 2009
3. Mobithinking: http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats#mobilebehavior
4. CSS Sprites:
5. Data URIs: http://css-tricks.com/data-uris/
6. jQuery Mobile: http://jquerymobile.com/
7. jQueryUI: http://jqueryui.com/
8. jQuery Mobile Themeroller: http://jquerymobile.com/themeroller