# Mobilising your Web Service

This document describes different approaches for enabling a web site for mobile use. It is not intended to be a concise guide but a working document with links to additional resource on the subject.

## Why mobilize a website

03 May 2009

Today, the users of the internet are more and more tapping into the world wide web from several different locations. The same user often uses a service or site both from a desktop computer over a broadband connection as well as from a smartphone. For these users it is important that the service offer a consistent user experience, regardless of the differences in screen sizes and input methods.

Another, ever growing part is those users whose only or principal method of using internet is with a mobile device. On top of being another media for browsing, a mobile device is a lot more. Most internet-capable phones have a camera and they are also used as Personal Digital Assistants, combining such functions as calendar and contacts databases together with messaging services. And most importantly, a mobile phone can instantly make a phone call, with a click of a button. Through connected widgets, it is also possible to utilize location based information on the device, which is a great help in providing the user with targeted, local services.

## Different mobile browsers and devices

The devices available today vary from very small to very big, from simple to advanced. This means that not only is the user base huge, it's also heterogeneous. The simplest device will only be able to show text based html pages on a small sreen, whereas the most advanced devices differ very little from desktop browsers, apart from the physical size.

All Nokia devices and their specifications can be found on the Nokia Developer website.

### Layouts and sizes

The most commonly used screen resolution in peoples' pockets is QVGA (240x320 pixels). More and more devices are using bigger resolutions, but there is a physical limit to how big a devices screen can be for it to be comfortably mobile, so even if the resolutions get bigger, resulting in crisper images, the physical size of the screen will not be growing over a certain pocketable size. Actually there is also a huge range of physical sizes in the common QVGA resolution. There's a nice article on the popularity of different display resolutions.

To add to the complexity, the devices can also be either portrait, landscape or both. This means that the device either has a fixed layout, like the Nokia E63, which is only in landscape mode or the Nokia 6212 classic, which is laid out as portrait. The other option is that the device can switch its layout between portrait and landscape. This can happen either through user action or automatically. User action here means something like sliding the screen in the Nokia N95, or opening the device in the Nokia E90 Communicator. The latter has even the additional challenge of changing from QVGA resolution to a communicator specific 800x352 pixels. Automatically means that the sensors inside the device will know which way it's held in the user's hand and will switch the display orientation accordingly, like in the Nokia 5800 XpressMusic or the Nokia N85

### Browser capabilities

There are also differences in browser capabilities. As said earlier, web-capable devices range from the very simple to the very advanced. For the website developer this means two things:

- Don't expect the mobile browser to have the same capabilities as the desktop browser.
- Don't assume that *all* mobile devices need a simple text-only html page.

This means that when designing a service or a web page for mobile users, it's always best to provide at least two options. The simple text version for feature phones and a more advanced model for the high end devices.

### Bandwidth and data plans

Not only is the amount of graphics and rich content limited by the device capabilities, but also by the network. And even when the device itself would be capable of displaying Flash and streaming media coupled together with dynamic layouts, transition effects and all sorts of eye candy - it's possible that the user does not have a fixed data plan or is currently roaming.


(C) Copyright Nokia 2014. All rights reserved.

In many regions fixed data charges are not even available and in these cases many users decide to turn off images and Flash content, except when browsing through a WLAN access point.

## Detecting the client browser

So in light of all the issues discussed above it is indeed a challenge to provide good usability and select the correct content to display to a user that's coming to your site armed with a mobile phone instead of a personal computer. The solution is, however, if not simple, at least available - browser and device detection.

### Is it mobile or not

The simplest method is to use the user agent string provided by each browser. It tells the web service which browser and in most cases also which device the user has. Unfortunately there is no standard way of telling "I'm a phone" in the user agent string so it takes a bit more work. The detection can be done with server side scripting, such as PHP or on the client side with JavaScript. Here's an example in JavaScript to detect all Nokia devices. Utilizing the fact that they have the word "nokia" in the user agent:

```
function isItNokia(){
    if (/(Series60|Nokia|SymbianOS)/i.test(navigator.userAgent)){
        return true;
    } else {
        return false;
    }
}
```

More examples on detecting devices with JavaScript can be found on this website: [1].

This is of course a very lightweight approach and should be only used while waiting for a more robust and extensive device detection to be enabled.

### What kind of mobile

The next step is to figure out what kind of device the visitor has. For this there are several solutions, which are described in detail in the document Detecting Mobile Devices on Web Services, therefore there is no need to go into details here.

## Selecting a method for providing the mobile content

Now that it's known that a mobile-specific content is in order and we possibly even know which particular device it is, it's time to decide how to provide the adjusted content to the mobile browser. For simplicity the different approaches are described here as three different methods. In real life the way to provide the content depends of course on the technologies used and will be a mix of


(C) Copyright Nokia 2014. All rights reserved.

these methods or something completely different. The following methods should, however, give a general idea.

## Method 1. Designing a simple, scaling layout

One of the easiest ways, and one that suits a very simple web page, is to use relative positioning and scaling on the page. For example, not to use absolute pixel widths for the content, but define relative widths and let the height scale according to the content. In the following example colors are used for emphasis, the font heights are defined as millimiters and each element has been given a relative width. This means that the page will scale to fill the whole screen and it's up to the device or browser vendor to define whether a millimeter is really a millimeter. For example in Nokia S60 5th edition devices 10mm will be interpreted as 5mm.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <title>Dynamic scaling</title>
        <style type="text/css">
            * {
                font-size:10mm;
            }
            body {
                font-family:sans-serif;
                text-align:center;
                background-color:gray;
            }
            h1, h2, p, a {
                width:80%;
                display:block;
                font-weight:bold;
                padding:10px;
                margin:0 auto;
                color:white;
                font-size:70%;
            }
            h1, p {
                margin-bottom:10px;
            }
            h1 {
                background-color:red;
```


(C) Copyright Nokia 2014. All rights reserved.

```
                font-size:100%;
            }
            h2 {
                background-color:blue;
            }
            p {
                background-color:white;
                color:black;
                font-size:40%;
                font-weight:normal;
            }
            a {
                width:50%;
                background-color:green;
            }
        </style>
    </head>
    <body>
        <h1 name="top">Breaking news!</h1>
        <h2>Would you believe</h2>
        <p>
            Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
            Sed posuere interdum sem.
        </p>
        <h2>What about this!</h2>
        <p>
            Quisque ligula eros ullamcorper quis, lacinia quis
            facilisis sed sapien. Mauris varius diam vitae
            arcu. Sed arcu lectus auctor vitae, consectetuer
            et venenatis eget velit. Sed augue orci, lacinia eu
            tincidunt et eleifend nec lacus. Donec ultricies
            nisl ut felis, suspendisse potenti. Lorem ipsum
            ligula ut hendrerit mollis, ipsum erat
            vehicula risus, eu suscipit sem libero nec erat.
            Aliquam erat volutpat. Sed congue augue vitae
            neque. Nulla consectetuer porttitor pede.
        </p>
        <a href="#top">Back to top</a>
    </body>
</html>
```

The reason why the CSS data is embedded in the HTML is simply download time and user experience. A separate style sheet would load *after* the html and over a slow connection this means that the page is first first shown without any styling, which can be rather confusing to the user. This example is of course very basic, but can be extended with images and CSS3 effects like border-radius, drop-shadows etc. These will be transparent to any non-supporting device and will be ignored.

This kind of approach has the advantage of not needing any device or browser detection, but might be a bit dull on the desktop side. However, going "vanilla" this way can be a good option to use for a generic mobile page. And if the page is created server-side with e.g. PHP, there are more ways to make sure the content fits the given real estate.

## Method 2. Switching between mobile and Desktop layout


(C) Copyright Nokia 2014. All rights reserved.

If more differentiation is needed between the mobile and desktop versions of the site, but the content will stay more or less the same, it's a good idea to use different styles for both.

There are several ways for doing this, some of them are described in the Mobilising document.

## Selecting different CSS on the fly

There is an option to switch the used stylesheet dynamically. One way for doing this is using two separate css files and direct to a mobile version with Javascript after detecting a mobile browser is used. Here's an example:

First the two stylesheets are defined in the **head** section of the html file. Both are given a **title** attribute, which is used to tell them apart in code.

```
<head>
...
        <link rel="stylesheet" type="text/css" href="desktop.css" id="desktop" />
        <link rel="stylesheet" type="text/css" href="mobile.css" id="mobile" />
...
</head>
```

Then some way for detecting the device is used, as explained earlier, and a JavaScript function switches the stylesheets based on the **title** attribute.

```
function switchCSS(media){
    var obj = media == 'desktop'?'mobile':'desktop';
    document.getElementById(obj).disabled = true;
}
```

In the mobile version of the page some of the not-so-mobile-friendly content of the site can be switched or left out altogether. Also it's a good idea to make sure the font is big enough.

If the device detection algorithm is intelligent enough and uses services such as WURFL or Handset Detection the site will know if the device is touch enabled. In this case it's also a good policy to make sure that links and other tappable elements are big enough and far enough apart for enabling finger use.

## Hiding does not prevent downloading

One thing to keep in mind is, that if the page has large images, flash and other elements that are simply hidden in the mobile layout they will still be loaded to the device. For this reason it's in many cases better to rely on server side scripting to first select the content based on the detected device and then provide the appropriate content based on a suitable template.

# Method 3. Redirecting after detection


(C) Copyright Nokia 2014. All rights reserved.

Sometimes it's good to maintain separate sites for mobile and desktop users. This can be useful, for example in cases where the mobile site provides very different services comparing to the desktop version. It's also possible, that the owner of the site wants to use different domain names, such as **.com** or **.org** for desktop browsers and **.mobi** for the mobile.

Regular users can be taught to bookmark the relevant site and the device detection algorithms are used only to make sure first time users don't get lost. When directing to a mobile site it is good policy to ask the user if they really wish to do this and maybe even use cookies to remember the decision. The fact is, that especially in cases where the mobile site is very light and stripped, some users prefer to use the desktop site even on mobile.

### A general mobile site

Even when redirecting or teaching the users to go to a **.mobi** or *m.site.com* etc. It is still a good idea in many cases to detect which device and browser is used. As explained earlier, the so many different types of devices as well as users, that it's very hard to serve everyone with just one site. A text based site is no good or some smartphone users with a flat rate deal with their operator and a 2G feature phone user with no data plan will be very surprised with their next phone bill if the site is too heavy.

Since the ultimate decision relies on the users themselves it's good to provide them with options and not make too many assumptions. Although one has to keep in mind that users generally do not wish to be faced with *too* many choices either. A good solution is to make a silent and accurate device detection and only provide choices in unsure cases. Perhaps having a link to the lighter or heavier versions at the top of the site. Sometimes even a laptop user might prefer the mobile version, if the are connected through a phone roaming in EDGE network...

### RSS

RSS readers are becoming standard issue in smartphones. It's therefore a good idea to provide a link to your feeds also for the mobile user. And sometimes the easiest way to provide a mobile version of the site is to direct the mobile users to the RSS feeds directly. All modern mobile RSS readers will download the content periodically and save the user costs as well as energy in increased battery life if constant online use is not needed.

### Device specific sites

The ultimate device detection combined with smart content selection will be something perhaps an online game store would use. When you first find out exactly the make and model of the device, you can offer the user pre-selected software that will definitely work in their device. This can also be a great booster for sales when the user does not get frustrated wading through tons of non-applicable apps.

## Selecting the content to display for mobile users

As discussed earlier, there are several ways of providing content suitable for mobile browsers. Whether you decide to go light and strip all images out or rely heavily on device detection and target each device specifically, you still have to decide the actual content and form of the presentation. One way for doing it is ordering a market research to be done by a professional consulting company. Or you can simply ask your users what they would like to see on the site. If you are thinking of conducting a user questionnaire for mobile users, please keep in mind that very few devices have input methods suitable for lengthy replies, so keep it simple.

After figuring out, either through your own business intelligence or hired help, what the users want it's a lot easier to provide content that will keep the users coming back to your service.

### Know your users

One possible use for device detection is to silently check each browser visiting your site. Many content management systems provide such functions already and it's not hard to do it with PHP yourself. You just need to be able to save the data on the server side. When you analyse the data you can see how many of your users are using mobile devices, which devices those are and which parts of your site the mobile users are going for the most.

This data is first useful in figuring out the initial content, but later on, when you start advertising a mobile optimised web service it's even more important to be able to know which way to further develop your site.

### Omitting or adding?

The common misconception concerning mobile web sites is that they are *just* WAP pages that are a subset of the *real* site. This


(C) Copyright Nokia 2014. All rights reserved.

can of course be true, but in that case you are losing the opportunity to provide users on the move with services that fit their connected lifestyle. Already a *simple* WAP-sites could provide specific services, like safe operator billing and separate delivery DRM protection. When you know the capabilities of the specific device browsing your site it opens up a lot of potential business. Premium SMS payments is one of them, the ability to send special vouchers directly to the phone, or give a click able link to a phone number are some of more obvious choices. Organising competitions and getting user generated content becomes a lot more interesting when your users have GPS enabled devices with cameras in their pockets!

### Designing with a mobile focus

When you are designing a new service or site you should design your desktop site and mobile site together. Not only is it easier to design the functionality first than to try to *glue it on* afterward, but you can also have the added benefit of reusing your desktop components as such on your mobile site. Let's say you are planning on having a side navigation widget on all your pages. Why not design it in such a way that it is exactly 240x320 pixels and the font is big enough or scales easily for the mobile context? That way you side navigation could be your mobile landing page without any modifications needed. The same goes for any other element - think mobile and you save time and money. If your sites components can be shown one after another instead of all at once, and they fit snugly on a mobile phone display, you have a mobile version of your site already made.

Another thing to keep in mind is the added value the mobile audience and devices bring, referred to in the previous section. With a little thought you can design your service, from the scratch, in such a way that it can utilise the mobility of the users.

## Using transcoders

Another way of getting a mobilized version of you site is funneling your content through a transcoder. A transcocer is a service that downscale and modifies the stuff shoved in it's virtual funnel so that it better fits a mobile device. Transcoders can also use device detection to help decide the correct formats. Advanced versions can even downgrade video on the fly.

The benefit of using transcoders is that the content will become smaller in kilobytes, thus loading faster to the mobile device. You can create a transcoder yourself or use a commercially available proxy server. The main point is, that the content gets reduced in size *before* it's downloaded to the mobile device.

Some examples of commercial transcoder services are Skweezer, Teashark and Opera Mini.

## What if I'm using a content management system

A CMS is an increasingly popular way of maintaining a web site. Especially for large organizations. Some of the most popular Content Management Systems have had a mobile plugin developed for them. The idea behind such a plugin is the same than what has been described here. Below is a collections of links to the currently available plugins and solutions:

### Joomla

http://sourceforge.net/projects/joomlamobileplu/files/

### Drupal

http://www.mobiledrupal.com

### Wordpress

http://alexking.org/blog/2009/03/29/wordpress-mobile-edition-302

## Examples

Please feel free to link code snippets and example sites (good or bad) below.

Source: File:Mobileweb.zip

### Mobilised web sites

About.com has a top ten list of well designed mobile web sites: http://webtrends.about.com/od/mobileweb20/tp/Top-Mobile-Websites-.htm
Also MobiThinking.com has a list of Best and Worst of Mobile sites, besides which there are number of other Best Practices white papers and a catalogue of Showcases.

### Code snippets


(C) Copyright Nokia 2014. All rights reserved.

### W3C Schools

http://www.w3schools.com/

### JavaScript

http://www.hand-interactive.com/resources/detect-mobile-javascript.htm

### ASP.NET

http://www.asp.net/mobile

## Read more

- http://www.developer.nokia.com/Develop/Web/
- http://en.wikipedia.org/wiki/Mobile_browser
- http://www.w3.org/Mobile/

--Risalmin 11:23, 21 April 2009 (EEST)


(C) Copyright Nokia 2014. All rights reserved.