Porting iPhone web app to WRT - a porting example

Recommended additional reading: Porting iPhone web app to WRT on Nokia devices, which provides a general description of porting iPhone web app to WRT on Nokia devices.

Introduction

This article describes the porting process of a simple web app from iPhone to Web Runtime (WRT) on Nokia devices such as the Nokia N97 device. It complements the article Porting iPhone web app to WRT on Nokia devices, which provides a general description of what an iPhone web app developer needs to know when switching to WRT development on Nokia devices based on S60 on Symbian OS. The purpose of this article is to describe the porting steps in practice. The example is for a horoscope web application (http://horoscope.internetdesigns.eu/) that will be ported from iPhone to WRT.

Porting a widget consists of the following steps:

- creating a new project
- copying remote content (HTML pages, style sheets, and image resources) to local static content
- enabling user interaction with JavaScript™ code in Nokia devices

In addition to a porting case, the article discusses co-development possibilities and lists device-specific extensions to widgets on both platforms.

Prerequisites

To get started, you will need a tool to create HTML pages and JavaScript files. A simple text editor will serve the purpose just fine, but in this document Aptana Studio with the Nokia WRT plug-in is used, because it contains more sophisticated features than a text editor. In addition, Aptana Studio can be used to create iPhone web apps.

Tip: Aptana has been superseded by the Nokia Web Tools

For developers who are new to WRT, it will be helpful to read an introduction to WRT technology:

http://www.developer.nokia.com/Develop/Web/

Porting steps

Create a new project

1. Start IDE
2. Select File > New > Project.
3. Select Nokia Web Runtime (WRT) > New Nokia Web Runtime Widget. Click Next.
4. Select Basic Widget Project.
5. Name the project 'horoscope' and enter the location where the project will be created. Click Next.
6. Name the widget 'horoscope' and use com.widget.horoscope as the widget identifier. A unique widget identifier is required for successful installation of the widget on the device. Click Next.
7. Name the main HTML file 'horoscope'. This file, horoscope.html, will be replaced in the following steps, but you'll avoid having to rename the file if you name it 'horoscope.html' here.
8. Name the CSS file 'style' and the JavaScript file 'horoscope'. Click Finish. The project has been created.

Transfer remote website content into local static content

Your first task is to get the HTML pages from the website into the WRT project. The Horoscope web app is a dynamic website running in PHP, so getting the HTML files is done most easily by saving the web pages into the root directory of the Horoscope project. There are three pages to save:

1. Save http://horoscope.internetdesigns.eu/ as horoscope.html. Overwrite the existing horoscope.html in the project with this one.
2. Save http://horoscope.internetdesigns.eu/zodiac.php as zodiac.html.
3. Save http://horoscope.internetdesigns.eu/horoscope.php as zodiac_result.html.

The next step is to get the style sheet for the web application. This can be done by browsing to http://horoscope.internetdesigns.eu/css/style.css and saving the file into your computer. As with the HTML files, save the Cascading Style Sheet (CSS) file into the root directory of the Horoscope project.

After this, refresh the project so that the HTML files and the CSS file appear in it. Right-click on the 'horoscope' project in the Project view and select Refresh. The following files should now be visible in the project:

- horoscope.html
- horoscope.js
- Info.plist

(C) Copyright Nokia 2013. All rights reserved.

- style.css
- zodiac_result.html
- zodiac.html
- (wrt_preview_frame.html)
- (wrt_preview_main.html)

After the web pages have been saved, the next task is to save the pictures from the web application as local copies. The reason for doing this is because the HTML pages have relative references to the pictures, and creating local copies saves you the trouble of updating all the references. Here is the list of pictures to be saved:

- header.png (http://iphoneapps.internetdesigns.eu/images/header.png)
- home.png (http://iphoneapps.internetdesigns.eu/images/home.png)
- dayicon.png (http://horoscope.internetdesigns.eu/img/dayicon.png)
- weekicon.png (http://horoscope.internetdesigns.eu/img/weekicon.png)
- monthicon.png (http://horoscope.internetdesigns.eu/img/monthicon.png)
- aquarius.png (http://horoscope.internetdesigns.eu/img/aquarius.png)
- aries.png (http://horoscope.internetdesigns.eu/img/aries.png)
- cancer.png (http://horoscope.internetdesigns.eu/img/cancer.png)
- capricorn.png (http://horoscope.internetdesigns.eu/img/capricorn.png)
- gemini.png (http://horoscope.internetdesigns.eu/img/gemini.png)
- leo.png (http://horoscope.internetdesigns.eu/img/leo.png)
- libra.png (http://horoscope.internetdesigns.eu/img/libra.png)
- pisces.png (http://horoscope.internetdesigns.eu/img/pisces.png)
- sagittarius.png (http://horoscope.internetdesigns.eu/img/sagittarius.png)
- scorpio.png (http://horoscope.internetdesigns.eu/img/scorpio.png)
- taurus.png (http://horoscope.internetdesigns.eu/img/taurus.png)
- virgo.png (http://horoscope.internetdesigns.eu/img/virgo.png)

Create a new folder for the pictures:

1. Select File > New > Other.
2. Select General > Folder. Click Next.
3. Select the Horoscope project as the parent folder and name the folder 'img'. Click Finish. The folder has been created.

Save the pictures listed above into the img folder you just created.

Edit the Info.plist file

Open Info.plist in the IDEs editor by double-clicking on it in the Project view and make sure that horoscope.html is set as the main HTML file:

```
<key>MainHTML</key>
<string>horoscope.html</string>
```

Also make sure that the key AllowNetworkAccess is set to true:

```
<key>AllowNetworkAccess</key>
<true/>
```

AllowNetworkAccess enables the widget to access the network.

Edit HTML files and file paths to fit local context

The first modifications in the HTML pages will target the internal linking between the files in your project. First, in horoscope.html, change all occurrences of zodiac.php to zodiac.html. The result should look like this:

```
<tr valign="middle" align="center">
<td>Aries<br /><a href="zodiac.html?id=0" ><img src="img/aries.png" alt="Aries" border="0" /></a>
</td>
<td>Taurus<br /><a href="zodiac.html?id=1" ><img src="img/taurus.png" alt="Taurus" border="0" /></a>
</td>
<td>Gemini<br /><a href="zodiac.html?id=2" ><img src="img/gemini.png" alt="Gemini" border="0" /></a>
</td>
</tr>
```

Next, in the <head> tag of each HTML page (horoscope.html, zodiac.html, zodiac_result.html), change the JavaScript link to point to horoscope.js, from this:

```
script src="http://iphoneapps.internetdesigns.eu/javascript/functions.js" type="text/javascript"
```

to this:

```
<script src="horoscope.js" type="text/javascript"></script>
```


(C) Copyright Nokia 2013. All rights reserved.

You should have the horoscope.js in your project if you created the project with the New Project wizard (see Section 2.1, 'Create a new project').

Change the style-sheet link in every HTML file to point to style.css in the root folder of your project. This is the original line of code:

```
<link href="css/style.css" rel="stylesheet" type="text/css" />
```

and this should be the result:

```
<link href="style.css" rel="stylesheet" type="text/css" />
```

The style sheet itself contains many references to external images (for example, this line: li.menu:hover{background:url("http://iphoneapps.internetdesigns.eu/images/menutouched.png") repeat-x #369}), which causes the widget to download them from the internet. This doesn't necessarily matter, but if you prefer, you can comment these lines out. (Also, the widget looks just fine without them.)

With regard to the style sheet, you may have noticed some advanced CSS 3.0 parameters/values, such as text-overflow:ellipsis; or WebKit-related ones, such as -webkit-border-radius:8px;. These may or may not be understood by the WRT engine depending on the version in the user's device. However, this is not a problem; the ones that are not understood by the engine will be silently ignored by it.

Returning to the HTML files, change the menu list element in the zodiac.html so that it points to the zodiac_result.html instead of horoscope.php:

```
<ul class="pageitem">

<li class="textbox"><p>Which horoscope would you like to see?</p>
</li>

<li class="menu"><a href="zodiac_result.html?h=d&id="><img alt="changelog" src="img/dayicon.png"
/><span class="name">daily horoscope</span><span class="arrow"></span></a></li>

<li class="menu"><a href="zodiac_result.html?h=w&id="><img alt="changelog" src="img/weekicon.png"
/><span class="name">weekly horoscope</span><span class="arrow"></span></a></li>

<li class="menu"><a href="zodiac_result.html?h=m&id="><img alt="changelog" src="img/monthicon.png"
/><span class="name">monthly horoscope</span><span class="arrow"></span></a></li>

</ul>
```

Lastly, remove the content from zodiac_result.html:

```
<img src="img/" alt="" border="0" align="left" hspace="10"/><br /><span class="graytitle">
<br /></span><br /><br /><br />

<ul class="pageitem">

<li class="textbox"><img alt="daily" src="img/dayicon.png" /> <span class="name">daily horoscope
</span></li>

<li class="textbox"><strong></strong><br /><br /></li>

</ul>
```

The fewer elements on the screen the better, because they are visible while the result page is loading.

Remove references to external resources

You can now clean up the HTML pages a bit by removing references to the resources in the web. First, remove the Google-related JavaScript at the end of each HTML page:

```
<script type="text/javascript">

var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");

document.write(unescape("%3Cscript src='" + gaJsHost
+ "google-analytics.com/ga.js' type='text/javascript'%3E%3C/script%3E"));

</script>

<script type="text/javascript">

try {

var pageTracker = _gat._getTracker("UA-5622831-6");

pageTracker._trackPageview();

} catch(err) {}</script>
```

Also remove advertisements that lead to the affiliate website in zodiac.html and zodiac_result.html:

```
<a href="http://iphoneapps.internetdesigns.eu/ringostation.htm">
<img src="http://affiliates.ringostation.com/images/iphone_320_50_22.png" border="0" />
</a>
```

However, keep the reference to the original creator of the web app in the footer (to give credit where credit is due).

```
<div id="footer">

<a class="noeffect" href="http://iphoneapps.internetdesigns.eu">Designed by InternetDesigns</a>
```


(C) Copyright Nokia 2013. All rights reserved.

```
</div>
```

Test the first working project

At this point, you have a working set of static HTML pages with internal linking and without unnecessary references to the resources in the web.

Test the HTML pages with the IDEs own editor. To do this, click on the Nokia Web Runtime (WRT) tab at the bottom of the editor view. The result should look something like this:

Don't worry about functionality yet; we'll return to that shortly.

Make the application scalable to different screen sizes

iPhone has a specific screen size that is used among all models. This is not the case with the plethora of Nokia devices. For Nokia touch devices, the current resolution is 360 x 640 pixels, and it can be either portrait or landscape, depending on which way the user is holding the device (for example, the Nokia N97 and Nokia 5800 XpressMusic devices). The common screen size for non-touch devices is QVGA (240 x 320 pixels) in a variety of physical sizes. Depending on the device, it can be portrait (for example, the Nokia 6730 device), landscape (the Nokia E71 and similar devices), or both (for example, the Nokia N95, Nokia N96, and Nokia 6760 slide devices). Furthermore, there are special cases such as the Nokia E90 Communicator, which has a small cover display of 240 x 320 pixels and a large internal display of 800 x 352 pixels.

The same WRT widget can be installed into all of these devices, so you should make sure that your WRT widget always fits nicely in the user's screen.

Take a look at the <table> tag in horoscope.html. It has a fixed width that matches the iPhone's screen width. This is what you would like to change, but since your widget will have to deal with many different screen sizes, you should handle that dynamically through some JavaScript code.

First, give an ID to your table 'mainTable'):

```
<table id="mainTable" border="0" cellpadding="0" cellspacing="0"
width="320" class="zodiac"  align="center">
```

Then, in horoscope.js, add this code:

```
window.onload = init;

window.onresize = windowResized;

function init() {

// Call windowResized() manually because it doesn't get called automatically

// for the first time

windowResized();

}

function windowResized() {

document.getElementById("mainTable").width = window.innerWidth;

}
```

init() is called when the HTML page is loaded, and windowResized() is called whenever the screen size changes. windowResized() must also be called from the init() function because it does not get called automatically for the first time (in other words, the resize event is not raised when the widget is launched). windowResized() gets the inner width of the window (the width of the content area) and sets the width of the table to match it.

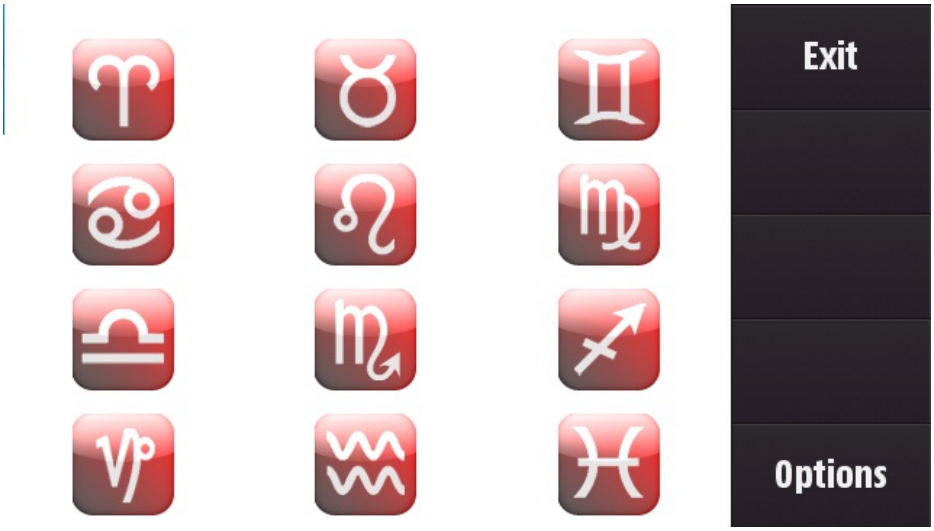
Of course there may also be cases where the content doesn't fit the screen vertically. However, the height of the whole HTML page is not easy to restrict. After all, HTML is designed for computer displays where it can flow vertically and be scrolled. Therefore, vertical scaling of content is omitted from this document.

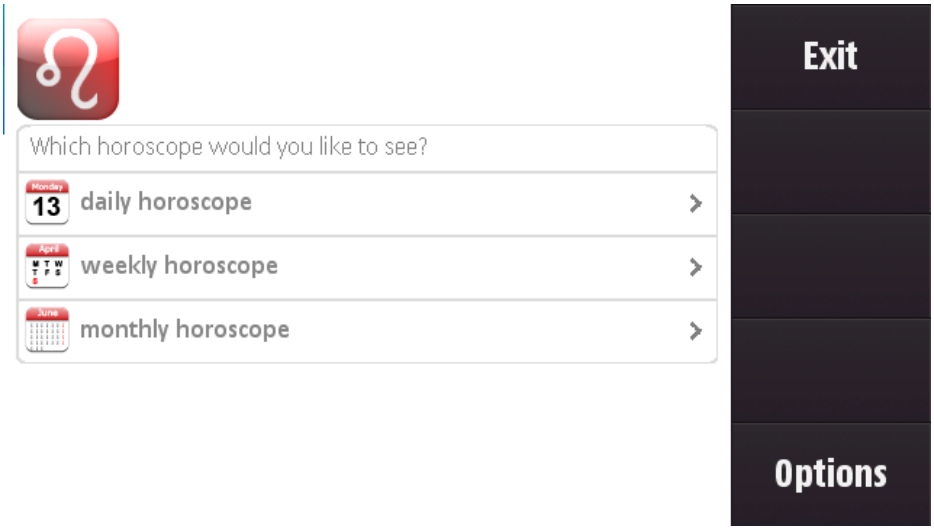
(C) Copyright Nokia 2013. All rights reserved.

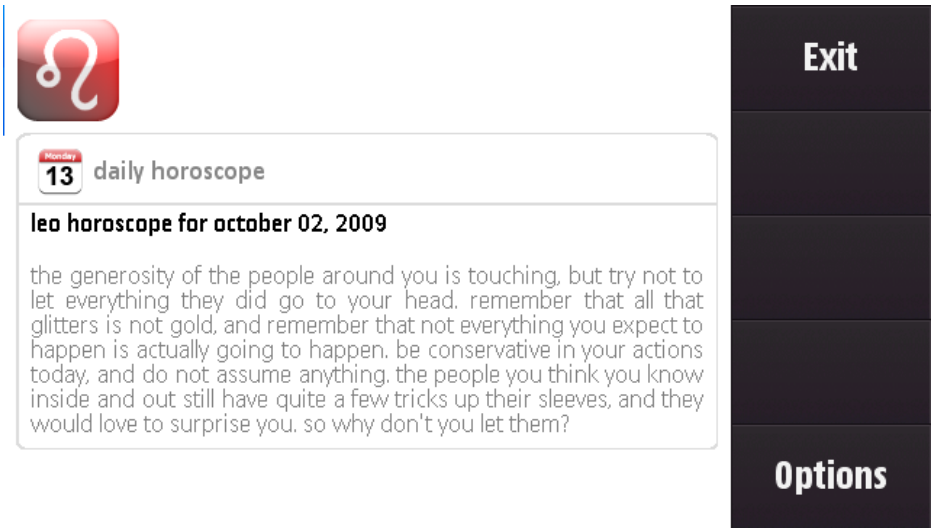
There is one other thing that can be done to improve the layout of the widget — remove the iPhone-specific title bar from every HTML file:

```
<div id="topbar">
<div id="title">Horoscope</div>
</div>
```

Removing the title bar allows the content to fit nicely in landscape orientation in nHD resolution (640 x 360 pixels):







NOTE: If there are any commands in the title bar, they should be moved to the Symbian S60 Options menu. Creation of the Options menu will be

(C) Copyright Nokia 2013. All rights reserved.

discussed later.

Add horoscope-related functionality with JavaScript

Horoscope is originally a PHP-based website, which means that all work is done on the server side in PHP code. This is fine for a remote website, but not for your WRT widget. The goal now is to make zodiac.html and zodiac_result.html work with some JavaScript so that they will act as original PHP code.

In this section, nothing is WRT specific. This is all about JavaScript code.

* zodiac.html:

This HTML page shows the sign that was selected in horoscope.html. Currently, the image source (the src attribute) is empty because no sign has been selected:

```
<img src="img/" alt="" border="0" align="left" hspace="10"/>
<br /><span class="graytitle">
<br /></span><br /><br /><br />
```

To be able to fill in the required image source through JavaScript, add an ID attribute to the img tag. This way you can refer to the img element from JavaScript code. Call the element 'image':

```
<img id="image" src="img/" alt="" border="0" align="left" hspace="10"/>
<br /><span class="graytitle"><br /></span><br /><br /><br />
```

Next, parse the ID value from the URL parameters. For this, write a helper function that stores all URL parameters into an array. Open horoscope.js and write the following function in it:

```
function getUrlVars() {
var vars = [];
var hash;
var hashes = location.search.substr(1).split("&");
for (var i = 0; i < hashes.length; i++) {
hash = hashes[i].split("=");
vars.push(hash[0]);
vars[hash[0]] = hash[1];
}
return vars;
}
```

Now you are ready to replace the empty src attribute of the image element with the location of the zodiac image in question:

```
function setZodiacImage() {
var hash = getUrlVars();
var image = "";
switch (hash['id']) {
case '0':
image = "img/aries.png";
break;
case '1':
image = "img/taurus.png";
break;
// Rest of the cases omitted for brevity.
}
document.getElementById("image").src = image;
}
```

The setZodiacImage() function above can be called after the image element has been declared. One possible place is right after the content div:

```
<div id="content">
<img id="image" src="img/" alt="" border="0" align="left" hspace="10"/>
<br /><span class="graytitle"><br /></span><br />
<br /><br />
<ul class="pageitem">
<li class="textbox"><p>Which horoscope would you like to see?</p></li>
<li class="menu"><a href="zodiac_result.html?h=d&id=">
<img alt="changelog" src="img/dayicon.png" />
<span class="name">daily horoscope</span>
<span class="arrow"></span></a></li>
<li class="menu"><a href="zodiac_result.html?h=w&id=">
<img alt="changelog" src="img/weekicon.png" />
```


(C) Copyright Nokia 2013. All rights reserved.

```
<span class="name">weekly horoscope</span><span class="arrow"></span></a></li>

<li class="menu"><a href="zodiac_result.html?h=m&id=">
<img alt="changelog" src="img/monthicon.png" />
<span class="name">monthly horoscope</span><span class="arrow"></span></a></li>

</ul>

</div>

<!-- google_afm -->

<script language="javascript">;

setZodiacImage();

</script>
```

Clicking on the 'menu' list item elements in zodiac.html should also provide zodiac_result.html with correct IDs. Currently ID parameters are empty (**'href="zodiac_result.html?h=d&id="'**). '*First, however, give each menu anchor an ID so that the anchors can be referred to*' from the JavaScript code. Call them 'day', 'week', and 'month':

```
<li class="menu"><a id="day" href="zodiac_result.html?h=d&id=">
<img alt="changelog" src="img/dayicon.png" />
<span class="name">daily horoscope</span>
<span class="arrow"></span></a></li>

<li class="menu"><a id="week" href="zodiac_result.html?h=w&id=">
<img alt="changelog" src="img/weekicon.png" />
<span class="name">weekly horoscope
</span><span class="arrow">
</span></a></li>

<li class="menu"><a id="month" href="zodiac_result.html?h=m&id=">
<img alt="changelog" src="img/monthicon.png" />
<span class="name">monthly horoscope</span>
<span class="arrow"></span></a></li>
```

Now write a function that appends the right ID to the href attributes of every anchor element:

```
function setZodiacId() {

var vars = getUrlVars();

document.getElementById("day").href = document.getElementById("day").href + vars['id'];

document.getElementById("week").href = document.getElementById("week").href + vars['id'];

document.getElementById("month").href = document.getElementById("month").href + vars['id'];

}
```

Here, again, the getUrlVars() function (declared previously) parses the parameters from the URL and returns them in an array.

Call this function right after you have called the setZodiacImage() function:

```
<script language="javascript">;

setZodiacImage();

setZodiacId();

</script>

* zodiac_result.html:
```

This HTML page shows the requested horoscope. It is the only thing that needs to be picked up from the internet. Since you do not have the original PHP code, you will get the content from the original website using Ajax, and place the content between tags.

The JavaScript function getHoroscope() makes a request to the horoscope website, and, after receiving data, calls another function, named gotResult():

```
function getHoroscope() {

var vars = getUrlVars();

if (window.XMLHttpRequest) {

xhr_object = new XMLHttpRequest();

} else {

alert("Ajax not supported.");

return;

}

var url = "http://horoscope.internetdesigns.eu/horoscope.php?h="+vars['h']+"&id="+vars['id'];

xhr_object.open("GET", url, true);

xhr_object.onreadystatechange = gotResult;

xhr_object.send(null);

}

function gotResult() {

if (xhr_object.readyState == 4) {

if (xhr_object.status == 200) {

displayHoroscopeResult();
```


(C) Copyright Nokia 2013. All rights reserved.

```
}
else {
alert("Bad HTTP status");
}
}
}
```

Finally, displayHoroscopeResult() is called, and it parses the web page and writes the selected data into zodiac_result.html:

```
function displayHoroscopeResult() {
var content = xhr_object.responseText.toLowerCase();
var div_start = 0, div_stop = 0;
var text;
div_start = content.indexOf("<div id=\"content\"");
div_start = content.indexOf(">", div_start);
div_stop = content.indexOf("</div", div_start);
text = content.slice(div_start+1, div_stop-1);
document.getElementById("content").innerHTML = text;
}
```

Create Options menu

Remember that iPhone is a touch-only device, whereas Nokia devices range from full QWERTY keyboard and conventional ITU-T mobile phone keys to touchscreen only. And these devices can, of course, have any or all input options in the same device.

When using WRT's widget.menu object, the WRT engine is aware of the device's input methods, and shows the menu in the correct format. Therefore, using menu-controlled WRT is usually the way to proceed. In this case, the user probably wants to have a way to go back easily from zodiac_result.html to zodiac.html to select another scale of dates.

You will want to add a menu entry in your WRT to be able to go back:

- From zodiac.html to horoscope.html;

- From zodiac_result.html to zodiac.html.

You'll also want to be able to go back to the main HTML page from the zodiac.html and zodiac_result.html pages.

Add the menu entries using the widget menu object, which is specific to WRT technology. Creating menu entries is straightforward:

- Create a menu item;

- Link a callback to this menu item;

- Append the menu item to the menu;

- Handle menu item selection in the callback.

For instance:

```
var CMD_BACK = 1;
backMenuItem = new MenuItem("Back", CMD_BACK);
backMenuItem.onSelect = onMenuItemSelected;
window.menu.append(backMenuItem);
```

CMD_BACK is a global variable that owns a number (here: 1). Each number is used to identify the menu item in the callback, which should follow this prototype:

```
function nameOfCallBack(<parameter>)
```

In your case:

```
function onMenuItemSelected(command) {
switch (command) {
case CMD_BACK:
// Branch depending on where you are
if (location.href.indexOf("zodiac.html") != -1) {
// If you are on the zodiac.html page, go back to the main page
location = "horoscope.html";
```


(C) Copyright Nokia 2013. All rights reserved.

```
} else if (location.href.indexOf("zodiac_result.html") != -1) {
// If you are on the zodiac_result.html page, go back to the
// zodiac.html page. Take care that the zodiac ID is preserved.
var vars = getUrlVars();
var id = vars['id'];
location = "zodiac.html?id=" + id;
}
break;
default:
break;
}
}
```

Have a look at the addMenus() function in the horoscope.js file. It is called by init(), which is called by the onload JavaScript event.

NOTE: It is also possible to implement the Back command on the right softkey. This provides a typical way of switching between different views and also prevents accidental closing of the application. For further details, see a related Nokia Developer code snippet: CS001402 - Controlling the softkey in WRT.

Conclusions

First conclusion

Porting the web app to WRT is easy. Moreover, you don't need any specific knowledge or technique to do it. In other words, an experienced iPhone web app developer won't have any trouble switching to WRT. The only issue is if the web app uses CSS 3.0 or iPhone-specific JavaScript events (see Chapter 4, Replacement technologies Porting iPhone web app to WRT on Nokia devices.

Second conclusion

Creating menus on WRT is quite simple. Though this was a small example of what you can do with WRT JavaScript APIs, all other APIs are quite easy to use, too. Again, switching from iPhone web app development to WRT should not be hard — knowing JavaScript enables you to be ready to develop for WRT; you simply need the API documentation provided in the Nokia Developer Web Developer's Library.

iPhone web app/WRT widget co-development tips

You've seen how to port an iPhone web app to a WRT widget. This works well as a one-shot activity, but is tedious to maintain if the original web app is updated. Thus, when creating a new web app/widget it is worthwhile to avoid using any WRT- or iPhone-specific components, or to make sure that both platforms are supported.

A small WRT widget acts as a launcher

If you want to keep both platforms supported, here's one solution. Provide a small launcher WRT widget, which will load the iPhone- and Nokia device -enabled remote web app.

The launcher widget will:

- Set up the menu entries (back item, etc.);

- Load the remote web page from the internet.

You will only need the horoscope.html page in your widget, with the following content:

```
<!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>
<title>Horoscope</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
</body>
<script language="JavaScript">
var CMD_BACK = 1;
function onSelectedItem(command) {
```


(C) Copyright Nokia 2013. All rights reserved.

```
switch (command) {
case CMD_BACK:
history.back();
break;
}
}
if (window.widget) {
var timerid = setInterval(function() {
if (history.length != 0) {
var backMenuItem = new MenuItem("Back", CMD_BACK);
backMenuItem.onSelect = onSelectedItem;
window.menu.append(backMenuItem);
clearInterval(timerid);
}
}, 1500);
widget.setNavigationEnabled(true);
window.menu.showSoftkeys();
}
this.location.href="http://horoscope.internetdesigns.eu/";
</script>
</html>
```

In this case, the only maintenance needed in the WRT widget involves the menu: adding/removing/modifying items. The work will be centralised on http://horoscope.internetdesigns.eu/, the original web app.

Device includes the simple WRT files and the updateable code resides on the server side. Code updates have an effect on both WRT and iPhone pages.

Optimising the UI

Taking both iPhone and Nokia devices into account is a pretty straightforward matter with this web application. The only required modification is to adapt the table on the first page (horoscope.html) to the screen size of the device, which you already did in Section 2.6, 'Make the applications scalable to different screen sizes'. That is all that is needed, because JavaScript works on both platforms. Now, maintaining both the web app and the WRT widget will only require maintaining the code on http://horoscope.internetdesigns.eu/.

When porting the application, we did not fully optimise the user interface for Nokia devices (for example for S60 5th Edition devices). Optimisation is naturally recommended to achieve the best look and feel on target devices. This porting case was intended to demonstrate basic porting steps; fine-tuning of the application to fully meet the Nokia UI guidelines and user experience recommendations will be covered in upcoming articles.

Using device-specific JavaScript APIs in WRT

If the web app uses specific third-party JavaScript APIs that let the developer access device-specific (in this case, iPhone-specific) features, porting to WRT may not be as simple as demonstrated here. One of these APIs is PhoneGAP. This API provides functions for accessing location, contact and accelerometer data, and making use of the device's vibration and sound functionality. Unfortunately, even though PhoneGap is supposed to be a cross-platform API, it will not run on Nokia devices. However, if you have used such a third-party API, it is still possible to make your code compatible with Nokia devices by using Platform Services. Platform Services will allow WRT widgets to do the following:

- Access and launch applications on a device (AppManager Service API)
- Access and manage calendar information (Calendar Service API)
- Access and manage information about contacts (Contacts Service API)

(C) Copyright Nokia 2013. All rights reserved.

- Access and manage information about landmarks (Landmarks Service API)
- Access device logging events (Logging Service API)
- Access device location information and perform location-based calculations (Location Service API)
- Access information about media files stored on a device (Media Management Service API)
- Send, retrieve, and manage messages such as SMS and MMS (Messaging Service API)
- Access data from the physical sensors of a device (Sensor Service API)
- Access and modify system information on a device (SystemInfo Service API of WRT 1.1)
- Access system information and control certain device features (SystemInfo Service API of WRT 1.0)

Platform Services are supported from S60 5th Edition onwards and are available as updates to some S60 3rd Edition devices as well.

Developers can use device detection mechanisms familiar from the mobile web to figure out if the device the application is running in supports extra functionalities. More information can be found at: Mobilising your Web Service#Detecting the client browser.

Final conclusion

It is fairly straightforward to port an iPhone web app that uses basic HTML, CSS, and JavaScript to WRT. Developers will need to rework the web app a bit more to provide similar functionality in a WRT widget *only* if device-specific APIs were originally used. Fortunately, the technologies supported in the platforms are very similar — behind the scenes it is still HTML/CSS/JavaScript. This means that iPhone developers will not face an unknown world when switching to WRT widgets. On top of that, the potential user base for their applications will expand tremendously.

To test the ported application on a device, download the unzip the Horoscope.zip file and install the widget (horoscope.wgz) to your device.

Media:Horoscope.zip

Copyright © 2009 Nokia Corporation. All rights reserved. Nokia and Nokia Developer are trademarks or registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned may be trademarks or trade names of their respective owners.

 (C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.