How to localise a WRT widget

Scope

There are good instructions for localisation in the Web Developer's library this article is a step by step code example for localising a simple application. None of the content in the Library is duplicated here so if you have not already done so, **please read the localisation section of the library before continuing**

This widget uses a bit more straight forward method for using the localisation strings in the JavaScript code(variables instead of a function call), but the main principle is the same.

Step 1. The unlocalised code

We are using a simple Hello world example

Main HTML

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title></title>
        <script type="text/javascript" src="HelloWorld.js"></script>
    </head>
        <body onload="init()">
    </body>
</html>
```

JavaScript

A function to concatenate and display the Hello world text:

```
...
function helloButtonClicked(event) {
    var name = nameField.getText();
    if (name.length == "") {
        uiManager.showNotification(3000, "warning", "Please enter your name!");
    } else {
        uiManager.showNotification(3000, "info", "Hello " + name + "!");
    }
}
...
```

Manifest file

The name key in the info.plist file can be localised, but the change is automatic and the info.plist file itself needs not editing. You can leave the default language string in it.

```
...
<plist version="1.0">
    <dict>
        <key>DisplayName</key>
        <string>Hello World</string>
        ...
```


(C) Copyright Nokia 2014. All rights reserved.

Step 2. Creating localisation resource folders

Under the Widget's root folder, create a resource folder per language and name it like this: **`<ISO 639-1>.lproj`**

You can find the ISO 639-1 codes at the Registration authority's home page

In our example we provide localised content for English, Finnish and Danish. We need to create folders for *Finnish* `fi.lproj` and *Danish* `da.lproj` Note, that here is no need to create a folder for English, since we will use English as the default language.

The localised files will go into their respective folders. All having the same name as the the default files in the Widget's root folder.

Step 3. Making the code language aware

Main part of the language awareness is already done in the framework and the Widget knows how to retrieve certain files from a language specific location. Some source code changes are needed.

Main HTML

This is the best part! since the original html code is following good internationalisation customs and there's no string content in it,

```
only one addition is needed!
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title></title>
        <script type="text/javascript" src="locStrings.js" /> //<- Added
        <script type="text/javascript" src="HelloWorld.js" />
    </head>
            <body onload="init()">
    </body>
</html>
```

JavaScript

The JavaScript file, or Files need a bit more modification. This if of course not needed when you create your project as an internationalised code project from the very start, but we are assuming here that you had some good reasons for not doing it in the first place... What we do here is to replace all display texts with variable names. It is a good idea to use clearly distinguishable names to make localisation easier. Especially when the localisaer is a different person. In this case we are using the prefix **`qtn_`**, familiar from Symbian localisation, to identify a text string.

```
...
function helloButtonClicked(event) {
    var name = nameField.getText();
    if (name.length == "") {
        uiManager.showNotification(3000, "warning", qtn_note_warning_text);
    } else {
        uiManager.showNotification(3000, "info",
          qtn_note_hello_first_part + name + qtn_note_hello_last_part);
    }
}
...
```

Step 4. Creating the Localisation text file

Source language

(C) Copyright Nokia 2014. All rights reserved.

First, we need a source file, which is easy to localise. This means, that some commments are needed to explain where the text is used. This file is saved in the Widgets root folder as `locStrings.js` And this is also the file you can send to be localised.

```
//Default (English) texts. Don't modify this file.
//When localising, you must put keep the variable definition intact (var qtn_xxx =)
//You must also enclose the localised text in double quotes and close with a semicolon.
//If the language uses extended characters the localised file must be saved as UTF-8.
...
//Text to be concatenated as first_part + Name + last_part
//If the name is given as "Ruikku" the English example will be rendered as
//"Hello Ruikku!". Please remember to add spaces, if they are needed.
var qtn_note_hello_first_part = "Hello ";
var qtn_note_hello_last_part = "!"
...
```

Target language

The Finnish localisation texts are saved in **`fi.lproj/locStrings.js`** The comments are stripped, since they are not needed any more. Also please note, that the text is very different from the English. The Finnish version says "Hi <name>, howya?". Clearly the localiser has taken some privileges here...

```
//Finnish display texts for the Hello World Widget.
//Localised by Ruikku.
...
var qtn_note_hello_first_part = "Moi ";
var qtn_note_hello_last_part = ", mitä kuuluu?"
...
```

Afterthought

Think international!

Localisation is a very tricky business and if you have not kept internationalisation in your mind when coding your project, then it can be almost impossible to later add localisation to it. So instead of trying to add localisation at the end of your project, think about it all the time, when creating your code. If any parts of what you are doing at some point will affect how or what is displayed on the device's display, you need to internationalise it.

Also, not many of us are polyglots by nature and in real life current machine translation services do not provide legible text. So in many cases it is someone else localising the strings. This means, that it's good idea to make their work easier by providing some comments about the source text and intended use of the string, as well as some info about the space where the text will appear. Remember to allow extra characters some German words are twice the amount of letter of English words

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.