How to debug Symbian Web Runtime apps

This example demonstrates how to debug Symbian Web Runtime apps. The JavaScript debug helper library included in this snippet can also be used with ordinary HTML pages.

Overview

Sometimes you might want to test and debug web runtime apps on the phone. This can be achieved by using a **debug.js** file and adding a corresponding **debug.css** file in your app. These files are available as zipped here: Media:how to debug widgets debug helper.zip. After including a **debug.js**, you have a JavaScript function `debug`, which takes one string (or object) and prints it into a page and also into the console log (if the application is exists).

By default, the debug messages are printed into inside a created div element with an id "debug". If there are no elements having an id "debug", one is created and appended into a body element. The debug div element will have a class "debug". The debug div element will have a h3 element containing a title, and each of the following debug messages are put inside a p element. The **debug.css** looks as follows:

```
.debug {
  font-size: x-small;
  border: thin black solid;
 margin: 0;
}

.debug p {
  font-size: x-small;
  margin: 0;
  padding: 0;
}

.debug h3 {
  font-size: x-small;
  margin: 0;
  padding: 0;
}
```

To use the debug helper javascript library, add the following lines into the head part of the html page:

```
<link rel="StyleSheet" href="style/debug.css" type="text/css" />
<script type="text/javascript" src="script/debug.js" charset="utf-8"></script>
```

The debug.js should be the first javascript file included.

To demonstrate the debug class, we added also a one debug("Here we are!") call inside the executeSnippet function.

```
// Executes the snippet.
// As an example, the current implementation queries a string from the user
// and prints it on the screen.
function executeSnippet() {
    text = queryText();
    if (text == null) {
        return;
    }
    debug("Here we are!");
    printString(text);
    displayNote("You entered: " + text);
```


(C) Copyright Nokia 2014. All rights reserved.

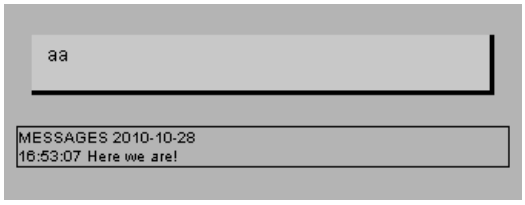
```
}
```

To make the plain HTML page usable with a normal web browser, we also added a following `if` block into the createMenu function, so that in normal browsers, the menus are not created.

```
// Creates the main menu
function createMenu() {
    if (!window.widget) {
      document.getElementById("bodyContent").innerHTML = "<button
onclick='executeSnippet();'>Click me!</button>";
      return;
    }
    executeMenuItem = new MenuItem("Execute snippet", CMD_EXECUTE);
    executeMenuItem.onSelect = onMenuItemSelected;
    window.menu.append(executeMenuItem);
}
```

Before, the widget looked as follows (after executing the widget):

After starting to use the debug helper javascript library, the widget looks as follows



The debug helper javascript library can be easily customized. For example, if the HTML page is divided into multiple subpages, which are shown by the javascript, you can output the message sent to the debug function into each subpage. For example, adding output into a subpage consisting of a div with id "viewX", use the following javascript code:

```
window.debugHelperFactory.addDebugElement("anotherDebugElement",{parentElement:
"viewX"});
```

There are the following initial parameters used inside the debug.js file when initializing it:

```
* canUseOriginalConsole: true // can use console.log?
* assertConsoleExists: true // assert that there will be a window.console object with a
log method
* createDefaultDebugHelper: true // create a window.mainDebugHelper with
* defaultDebugElementId: "debug" // if a default debug helper is created, set its id to
this
```

If you want to create a new instance of DebugHelper class, use the window.debugHelperFactory.make function, and provide it an object with the needed "configuration parameters" as properties. The configuration parameters are listed as follows:

Used by DebugHelper:

```
* useOriginalConsole: true // use console.log to log messages?
* alertMessages: false // use alert function to alert messages?
```


(C) Copyright Nokia 2014. All rights reserved.

```
* setAsMainDebugHelper: false // sets the created debugHelper as window.mainDebugHelper
element
```

Used by FakeConsole:

```
* debugElementId // optional, no default value, debug element id (if not found, the
element will be created)
* debugElement // optional, no default value, equals to the debug element
* showMessagesOnScreen: true // do we want to add messages into the
* containerElementClass: "debug" // class for the div element
* plainStub: false // used to make stub fake consoles, which do not print or create
anything
* showMessageId: false // in each debug log entry, show also the id of the debug message
* showTimeStamp: true // in each debug log entry, show also the timestamp of the debug
message
* title: "MESSAGES" // title
* createTitle: true // show the title?
* addDateToTitle: true // do we want to add a date to the title
* parentElement: "body" // parent element or parent element id (if not found, the body
element is used)
```

The created DebugHelper instances have the following, quite self-explanatory, methods: debug, log, disable (for temporarily disabling the output), enable, and reset. For example, to make a div element with id "alreadyExistingElement" to have a text "foobar" inside a p element, use the following lines of code:

```
var params = {debugElementId: "alreadyExistingElement",
              createTitle: false,
              showTimeStamp: false};
var newDebugHelper = window.debugHelperFactory.make(params);
newDebugHelper.reset(); // must be called after the XML has been loaded
newDebugHelper.debug("foobar"); // does not need to be called after the XML has been
loaded
```

Use the function `window.debugHelperFactory.preventDefault()` to prevent the creation of the default debug helper or to delete the created elements.

Postconditions

Now, when clicking the Execute menu item and entering some text, there will be a debug messages shown in a bordered box.

Supplementary material

This code snippet is part of the stub concept, which means that it has been patched on top of a template application in order to be more useful for developers. The version of the WRT stub application used as a template in this snippet is v1.2.

- The patched, executable application that can be used to test the features described in this snippet is available for download at Media:how to debug widgets example stub.zip.
- For unpatched stub applications, see Example app stubs with logging framework.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.