Learning MobileFirst hybrid client-side API

Overview

Prerequisite: To complete this tutorial, you must have previous experience with web development technologies such as HTML, CSS, JavaScript, and DOM events and manipulations. To learn these technologies, visit http://www.w3schools.com/ (http://www.w3schools.com/).

Although not required, a basic knowledge of jQuery and object-oriented JavaScript libraries is an advantage.

This tutorial covers the following topics.

- MobileFirst application components
- The WL namespace
- Sample application

MobileFirst application components

The following files are **essential application resources** in a MobileFirst application:

- **index.html:** The main HTML file
- **main.js:** The main JavaScript file
- **messages.js:** Messages file for storing application strings, primarily used for translation
- **initOptions.js:** Used for defining the initialization options of the application. See the documentation of the `WL.Client.init` method in the API reference part of the user documentation.
- **wljq.js:** An encapsulated version of jQuery
- **worklight.js:** The MobileFirst client API uses the WL namespace. This namespace provides bridging to native mobile platform APIs and other elements.

The `WL` namespace

The `WL` namespace is used to invoke MobileFirst APIs: `WL.Client`, `WL.App`, `WL.SimpleDialog` ...
The `WL` namespace exposes the API objects, methods, and constants (usually enums).
You make the `WL` namespace available in the application by having `worklight.js` automatically referenced in `index.html` when the application is generated in MobileFirst Studio.

WL.Client

With `WL.Client`, you can perform the following types of tasks.
Additional API methods are available in the IBM MobileFirst Platform Foundation user documentation topic for `WL.Client` (http://ibm.biz/knowctr#SSHS8R_7.0.0/com.ibm.worklight.apiref.doc/html/refjavascript-client/html/WL.Client.html).

Initialize and reload the application

- WL.Client.init(onSuccess, onFailure, timeout, ...)
- WL.Client.reloadApp()

Trigger login and logout

- WL.Client.login(realm, options)
- WL.Client.logout(realm, options)

Obtain general app information

- WL.Client.getEnvironment()
- WL.Environment.ADOBE_AIR
- ...

Retrieve and update data from corporate information systems

- WL.Client.invokeProcedure (invocationData, options)

Store and retrieve user preferences across sessions

- WL.Client.setUserPref(key, value, options)
- WL.Client.setUserPrefs({key1:value1, ...}, options)
- WL.Client.getUserPref(key)
- WL.Client.deleteUserPref(key, options)
- WL.Client.hasUserPref(key)

Specify environment-specific user interface behavior

- WL.App.openURL
- WL.App.getDeviceLanguage
- WL.App.getDeviceLocale
- WL.BusyIndicator
- WL.TabBar
- WL.SimpleDialog
- WL.OptionsMenu
- ...

Store custom log lines for auditing and reporting purposes in special database tables

- WL.Client.logActivity(activityType)

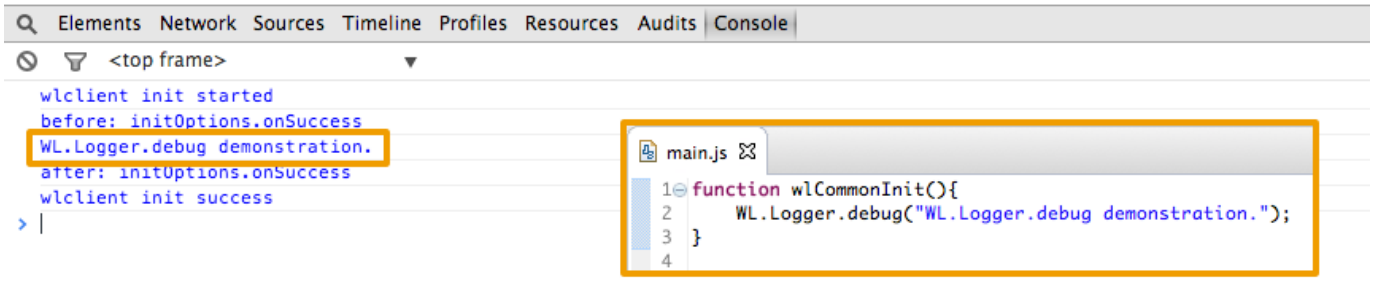
Note: This method is deprecated in V7.0. Use `WL.Logger` instead.

Write debug lines to a logger window (for example: Chrome's Dev Tools console)

- WL.Logger.debug

WL.Logger

`WL.Logger` helps you troubleshoot errors in environments that have no debugging tools.
`WL.Logger` outputs to an environment console, such as Xcode console, Adobe AIR, Android LogCat, Chrome Dev Tools, and similar tools.



Sample application

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/LearningMobileFirstHybridClientSideAPIProject.zip) the MobileFirst project.