# Learning MobileFirst hybrid client-side API

### **Overview**

To complete this tutorial, you must have previous experience with web development technologies such as HTML, CSS, JavaScript and DOM events & 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.

### MobileFirst application components

### **Application resources**

The following files are essential resources in a MobileFirst application:

- index.html: Main HTML filemain.js: Main JavaScript file
- messages.js: Messages file for storing application strings, primarily used for translation
- initOptions.js: Used for defining application initialization options (http://www-01.ibm.com/support/knowledgecenter/SSHS8R\_6.3.0/com.ibm.worklight.apiref.doc/html/refjavascriptclient/html/WL.Client.html%23init)
- wliq.is: An encapsulated version of ¡Query
- 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).

The WL namespace is 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 type of tasks.

Additional API methods are available in the IBM MobileFirst user documentation topic for WL.Client (http://www-01.ibm.com/support/knowledgecenter/SSHS8R\_6.3.0/com.ibm.worklight.apiref.doc/html/refjavascript-client/html/WL.Client.html? cp=SSHS8R\_6.3.0%2F9-0-0-1-7).

### 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)

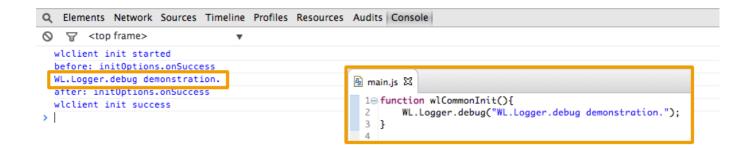
Write debug lines to a logger window (for example: Chrome's Dev Tools console):

• WL.Logger.debug

### WL.Logger

WL.Logger helps you troubleshoot errors in environments without debugging tools.

WL.Logger outputs to an environment console, such as Xcode console, Adobe AIR, Android LogCat, Chrome Dev Tools and the like.



## Sample application

Click to download

(http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/LearningMobileFirstHybridClientSideAPIProject.zip) the Studio project.