

Hybrid applications client-side API basics

fork and edit tutorial (<https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.1/client-side-development-basics/hybrid-applications-client-side-api-basics.html>) | report issue (<https://github.ibm.com/MFPSamples/DevCenter/issues/new>)

- Download MobileFirst project (<https://github.com/MobileFirst-Platform-Developer-Center/ClientSideAPI/tree/release71>)

Overview

This tutorial lists basic client-side API methods used to develop hybrid applications. The accompanying sample application demonstrates using a small subset of the methods.

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:

- The WL namespace
- Sample application

The WL namespace

The `WL` namespace is used to invoke MobileFirst APIs: `WL.Client`, `WL.App`, `WL.SimpleDialog` and so on.

The `WL` namespace exposes API objects, methods, and constants (usually enums).

The `WL` namespace is made available in the application by referencing `worklight.js` in `index.html`. This is done automatically by MobileFirst Studio or CLI when the application is generated.

WL.Client

With `WL.Client` you can perform the following type of tasks.

Additional API methods are available in the API reference for `WL.Client` in the user documentation

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)` - for environment that do not support REST API
- `WLResourceRequest (request URL, request type)`

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.

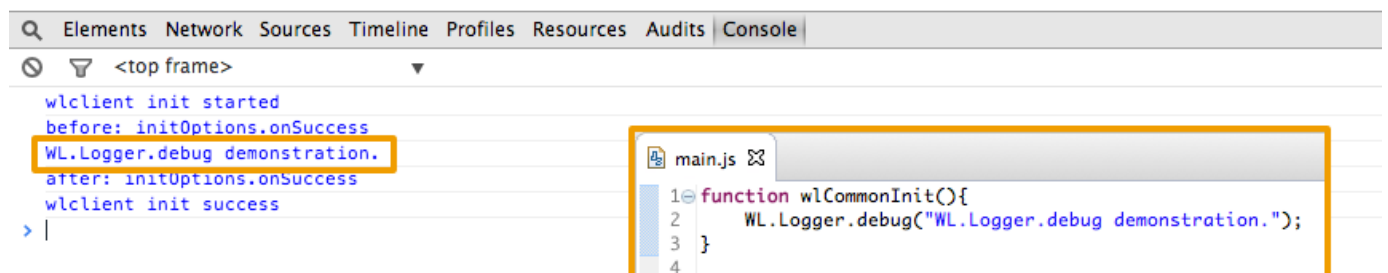
Write debug lines to a logger window (for example: Chrome's Dev Tools console)

- `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 (<https://github.com/MobileFirst-Platform-Developer-Center/ClientSideAPI/tree/release71>) the MobileFirst project.