Debugging Web Apps

Quickview

- You can debug your web app using console methods in JavaScript
- If debugging in a custom WebView, you need to implement a callback method to handle debug messages

In this document

See also

- Remote Debugging on Android
- Debugging

If you are testing your web app with a device running Android 4.4 or higher, you can remotely debug your web pages in `WebView` with Chrome Developer Tools, while continuing to support older versions of Android. For more information, see Remote Debugging on Android.

If you don't have a device running Android 4.4 or higher, you can debug your JavaScript using the `console` JavaScript APIs and view the output messages to logcat. If you're familiar with debugging web pages with Firebug or Web Inspector, then you're probably familiar with using `console` (such as `console.log()`). Android's WebKit framework supports most of the same APIs, so you can receive logs from your web page when debugging in Android's Browser or in your own `WebView`. This document describes how to use the console APIs for debugging.

Using Console APIs in the Android Browser

When you call a `console` function (in the DOM's `window.console` object), the output appears in logcat. For example, if your web page executes the following JavaScript:

```
console.log("Hello World");
```

Then the logcat message looks something like this:

```
Console: Hello World http://www.example.com/hello.html :82
```

The format of the message might appear different depending on which version of Android you're using. On Android 2.1 and higher, console messages from the Android Browser are tagged with the name "browser". On Android 1.6 and lower, Android Browser messages are tagged with the name "WebCore".

Android's WebKit does not implement all of the console APIs available in other desktop browsers. You can, however, use the basic text logging functions:

- `console.log(String)`
- `console.info(String)`
- `console.warn(String)`
- `console.error(String)`

Logcat

Logcat is a tool that dumps a log of system messages. The messages include a stack trace when the device throws an error, as well as log messages written from your application and those written using JavaScript `console` APIs.

To run logcat and view messages, execute `adb logcat` from your Android SDK `tools/` directory, or, from DDMS, select **Device > Run logcat**.

See Debugging for more information about .

Other console functions don't raise errors, but might not behave the same as what you expect from other web browsers.

Using Console APIs in WebView

All the console APIs shown above are also supported when debugging in `WebView`. If you're targeting Android 2.1 (API level 7) and higher, you must provide a `WebChromeClient` that implements the `onConsoleMessage()` method in order for console messages to appear in logcat. Then, apply the `WebChromeClient` to your `WebView` with `setWebChromeClient()`.

For example, to support API level 7, this is how your code for `onConsoleMessage(String, int, String)` might look:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebChromeClient(new WebChromeClient() {
  public void onConsoleMessage(String message, int lineNumber, String sourceID) {
    Log.d("MyApplication", message + " -- From line "
                         + lineNumber + " of "
                         + sourceID);
  }
});
```

However, if your lowest supported version is API level 8 or higher, you should instead implement `onConsoleMessage(ConsoleMessage)`. For example:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebChromeClient(new WebChromeClient() {
  public boolean onConsoleMessage(ConsoleMessage cm) {
    Log.d("MyApplication", cm.message() + " -- From line "
                         + cm.lineNumber() + " of "
                         + cm.sourceId() );
    return true;
  }
});
```

The `ConsoleMessage` also includes a `MessageLevel` object to indicate the type of console message being delivered. You can query the message level with `messageLevel()` to determine the severity of the message, then use the appropriate `Log` method or take other appropriate actions.

Whether you're using `onConsoleMessage(String, int, String)` or `onConsoleMessage(ConsoleMessage)`, when you execute a console method in your web page, Android calls the appropriate `onConsoleMessage()` method so you can report the error. For example, with the example code above, a logcat message is printed that looks like this:

```
Hello World -- From line 82 of http://www.example.com/hello.html
```