# Using Device Features with JavaScript

The following descriptions with examples show how to access various capabilities of a mobile device from JavaScript code.

## Popup user interactions

Use these functions to alert() the user, ask them to confirm() a question, or prompt() a user to enter some text. These are more fully described in: **JavaScript Popup Boxes**.

```
alert( "Your username or password is incorrect!" );

var response = confirm( "Are you ready to proceed?" );
if ( response ) alert( 'User is ready!' );

var username = prompt( "Who are you, really?", "Dr. Moriarity" );
// "Dr. Moriarity" is the default text in the prompt which the user can change
if ( username ) alert( 'This user is: ' + username );
```

### Improved popup interactions

The JavaScript functions for alert, confirm, and prompt will post your app's filename as the dialog title, which may not be acceptable for some apps. In this case, you can use the **Apache Cordova notification functions**. These functions, which are built-in to ViziApps, will allow you to change the title of the dialog box and even the button names, but they will not work when you use your app in Web Preview or for Web Apps. Here's an example of how to combine these functions so that they work across all platforms:

```
function popupConfirm( message, title, callback )
{
    if ( !navigator.notification ) {
        var response = confirm( message );
        if ( callback ) callback( response );
        return;
    }
    function handler( response ) {
        if ( callback ) callback( 1 == response );
    }
    navigator.notification.confirm( message, handler, title, null );
}
```

## Open a browser to any website

You can display any website to your users without leaving your ViziApps app by opening a popup web browser with a specific URL. Here is how to do it.

```
function openWebsite( url ) {
    return window.open( url, "_blank",
"location=yes,toolbar=yes,scrollbars=yes" );
}
var newWnd = openWebsite( 'http://www.wikipedia.org' );
newWnd.addEventListener( 'exit', function(ev)
{
    // this event will happen in the phones, not in Web Preview
    alert( 'Window Closed!' );
});
```

## Storing data on the device

With JavaScript you can store data on the device, up to 5MB, using the HTML5 **Local Storage** feature. This storage isn't a database, but a very simple key:value store. When you want to save some data value, you save it with a unique “key” that you make up, like a variable name, which you can use at a later time to retrieve that data. For example, if a user types in their username into a login page, you may want the app to remember the username so that the next time the app runs, the user doesn't have to retype their username.

```
// Save a string value
localStorage.setItem( "usernameKey", getFieldValue( "usernameField" ) );
// Restore the string value
setFieldValue( "usernameField", localStorage.getItem( "usernameKey" ) || 'new
user' );
```

#### Saving and Restoring Array Values from Tables

Sometimes the value that you want to save isn't a simple string, but a complex JavaScript object or array, such as the contents of a table. This is accomplished by converting the complex object to a string using JSON. If you have a table with fields defined as tblFullName, tblAddress, and tblUserKey, you could save and restore the table in this way:

```
// Save the table values
var saveTable = {
    name: getFieldArray( "tblFullName" ),
    addr: getFieldArray( "tblAddress" ),
    keys: getFieldArray( "tblUserKey" ),
};
localStorage.setItem( "savedUserTable", JSON.stringify( saveTable ) );

// Restore the table values
var restoreTable = JSON.parse( localStorage.getItem( "savedUserTable" ) );
setFieldArray( "tblFullName", restoreTable.name ),
setFieldArray( "tblAddress", restoreTable.addr ),
setFieldArray( "tblUserKey", restoreTable.keys ),
```

## Device Information

Information about a device's hardware or software is available from the **Device** object as in this example. The device.platform value can be either “Android” or “iOS”. The device.model value will indicate which iPhone or iPad model represents this device, or the specific Android manufacture's model name.

***This feature is not available for Web Apps and App Preview mode.***

```
function getDeviceInfo()
{
    var info = {};
    if ( !device ) return info;
    info.model = device.model;
    info.cordova = device.cordova;
    info.platform = device.platform;
    info.uuid = device.uuid;
    info.version = device.version;
    return info;
}
alert( 'Device information is: ' + JSON.stringify( getDeviceInfo(), null, '   '
) );
```

## Beep and Buzz

You can cause your device to emit a **beep** sound or to **vibrate** , as in this example.

***This feature is not available for Web Apps and App Preview mode.***

```
function makeBeep( howManyTimes ) {
    if ( !navigator.notification ) return;
    navigator.notification.beep( howManyTimes );
}
function makeBuzz( forMilliseconds ) {
    if ( !navigator.notification ) return;
    navigator.notification.vibrate( forMilliseconds );
}
makeBeep(2); // two beeps
makeBuzz(125); // vibrate for 1/8 second
```

## Accelerometer

The accelerometer is a motion sensor that detects the change (delta) in movement relative to the current position. The accelerometer can detect 3D movement along the x, y, and z axis. The **accelerometer.watchAcceleration()** method retrieves the device's current Acceleration at a regular interval. This example shows how to monitor acceleration.

***This feature is not available for Web Apps and App Preview mode.***

```
function onSuccess(acceleration) {
    alert('Acceleration X: ' + acceleration.x + '\n' +
          'Acceleration Y: ' + acceleration.y + '\n' +
          'Acceleration Z: ' + acceleration.z + '\n' +
          'Timestamp: '      + acceleration.timestamp + '\n');
}
function onError() {
    alert('onError!');
}
function watchAcceleromoter()
{
    if ( !navigator.accelerometer ) return;
    var options = { frequency: 3000 };  // Update every 3 seconds
    var watchID = navigator.accelerometer.watchAcceleration(onSuccess,
onError, options);
}
watchAcceleromoter();
```

## Network Connections

You can get the current status of the device's cellular or WiFi network connection by requesting the type from **navigator.connection** . Here is an example.

***This feature is not available for Web Apps and App Preview mode.***

```
function checkConnection() {
    if ( !navigator.connection ) return;
    var networkState = navigator.connection.type;
    var states = {};
    states[Connection.UNKNOWN]  = 'Unknown connection';
    states[Connection.ETHERNET] = 'Ethernet connection';
    states[Connection.WIFI]     = 'WiFi connection';
    states[Connection.CELL_2G]  = 'Cell 2G connection';
    states[Connection.CELL_3G]  = 'Cell 3G connection';
    states[Connection.CELL_4G]  = 'Cell 4G connection';
    states[Connection.CELL]     = 'Cell generic connection';
    states[Connection.NONE]     = 'No network connection';
    alert('Connection type: ' + states[networkState]);
}
checkConnection();
```