Customizing with jQuery

jQuery is a popular JavaScript framework that makes it easy to manipulate the HTML Document Object Model (DOM) - which is to say you can modify the user interface. jQuery is already built-in to ViziApps so you don't need to load any library; just start using it.

Some of the examples on this page show you how to tap into the powerful and easy-to-use event processor to catch user actions; how to fetch data from a web API; and how to hide or show fields.

> You can reference any ViziApps objects in jQuery such as fields and pages, but you must prepend the ViziApps name with a hashmark, and then complete with the jQuery operator $(). For example:
>
> ```
> var vzButtonField = $( '#' + 'buttonAskUserField' );
> var buttonId = vzButtonField.attr( 'id' );
> ```

App startup and deferred code

The general format for capturing an event uses the **jQuery .on() operator** as follows: $(*jq_selector*).on(*event_name*, *event_function*);

Your HTML Header JavaScript code cannot make references to ViziApps field names until the app content has been loaded. Use the following example to defer startup code until your app is ready:

```
$(document).on('DOMContentLoaded', function() {
    setFieldValue( 'startTimeLabel', (new Date()).toLocaleString() );
});
```

Page open events

You may want to take some action when any page is first opened, which you can do with the following code. Note that all references to ViziApps objects should happen after the app is ready.

```
function onPageShowing( pageName, callback ) {
    $( '#' + pageName ).on( 'showing', callback );
}
function offPageShowing( pageName, callback ) {
    $( '#' + pageName ).off( 'showing', callback );
```

```
}
function loginPageHandler() {
    offPageShowing( 'loginPage', loginPageHandler );
    if ( !confirm( 'Are you sure you want to login?' ) ) {
        gotoPage( 'appHomePage' );
    }
}
$(document).on('DOMContentLoaded', function() {
    onPageShowing( 'loginPage', loginPageHandler );
});
```

Tap events for any field

While buttons and tables have no-coding events built-in to their properties, you can use jQuery to turn images, labels, and just about anything into an object that can be tapped. If you create an Image Field called tapThisImage and a Label Field called tapThisLabel, the following code will alert the user to the name of the field that was tapped.

```
function clickHandler( event ) {
    var tappedField = $(event.currentTarget);
    alert( 'Clicked: ' + tappedField.attr( 'id' ) );
}
function clickListen( fieldName ) {
    $( '#' + fieldName ).click( clickHandler );
}
$(document).on('DOMContentLoaded', function() {
    clickListen( 'tapThisLabel' );
    clickListen( 'tapThisImage' );
});
```

Simulate a user click

You can programatically “click” on any object. One important reason to do this is to trigger a database query from a button that is configured to “Get or send device data via a web data source”. This example shows a button that lets the user refresh data on that page, but you also want to show the data the first time the page is shown.

```
// refreshDataButton is configured to get or send device data
function firstRefreshData() {
    offPageShowing( 'showImportantDataPage', firstRefreshData );
    $( '#refreshDataButton' ).click();
}
function windowStartup() {
    onPageShowing( 'showImportantDataPage', firstRefreshData );
```

```
}
$(document).on('DOMContentLoaded', windowStartup );
```

Get data from websites (AJAX)

You can use the **jQuery .get()** function and the more complete **jQuery .ajax()** to make requests to websites and receive data. This example uses the **OpenWeatherMap** website to get the current weather conditions for a specific city. Try creating a button and calling this showTemperature() function from the button's tap event.

```
// put showTemperature() in HTML Header <script>
// forLocation is a city,state,country string
function showTemperature( forLocation ) {
    var openWeatherUrl = 'http://api.openweathermap.org/data/2.5/weather?q=';
    $.get( openWeatherUrl + forLocation )
    .done( function( response ) {
        var tempk = ( ( response || {} ).main || {} ).temp;
        if ( !tempk ) {
            alert( 'Could not get temperature for ' + forLocation );
            return;
        }
        var tempc = tempk - 273.15;
        var tempf = Math.round( tempc * 9 / 5 + 32 );
        tempc = Math.round( tempc );
        alert( 'Temperature is ' + tempc + 'C, ' + tempf + 'F in ' +
response.name );
    })
    .fail( function( response ) {
        alert( 'Could not get temperature: ' + response.statusText );
    });
}

// call showTemperature() from a button event:
showTemperature( 'boston,ma,usa' );
```

Making **AJAX** calls from the Web Preview mode or from Web Apps will often fail because of security restrictions called **Cross-Origin Resource Sharing** violations. Also some web services may fail with CORS errors, even with requests made from mobile devices.

In Case of CORS Errors from Mobile Devices

If your mobile app is making AJAX requests to a web service but receiving CORS errors, ViziApps provides the **v.httpRequest('url','method',params,win,fail)** method which should always succeed. However, this method will not work for Web Preview mode or from Web Apps, does not offer the full functionality of the **$.ajax()** jQuery method, and will not automatically parse XML or JSON responses. You can find the details in our **Function Reference**, and following is the previous code example redefined for **v.httpRequest()**.

```
// put showTemperature() in HTML Header <script>
// forLocation is a city,state,country string
function showTempDevice( forLocation ) {
    var params = {
        'q': forLocation
    };
    var openWeatherUrl = 'http://api.openweathermap.org/data/2.5/weather';
    v.httpRequest( openWeatherUrl, 'get', params,
    function( responseText ) {
        var response = JSON.parse( responseText );
        var tempk = ( ( response || {} ).main || {} ).temp;
        if ( !tempk ) {
            alert( 'Could not get temperature for ' + forLocation );
            return;
        }
        var tempc = tempk - 273.15;
        var tempf = Math.round( tempc * 9 / 5 + 32 );
        tempc = Math.round( tempc );
        alert( 'Temperature is ' + tempc + 'C, ' + tempf + 'F in ' +
response.name );
    },
    function( responseText ) {
        var response = JSON.parse( responseText );
        alert( 'Could not get temperature: ' + response.statusText );
    });
}

// call showTemperature() from a button event:
showTemperature( 'boston,ma,usa' );
```

Saving extra data in fields

You can extend the properties of any ViziApps field with the **jQuery .data()** function. You save your data, which could be any string or even a complex JavaScript Object, using a key name that you define. Say you have a button called showAlertButton that should show an alert message that may be changed for different circumstances. You save the special message right in the Button Field itself using this code which defines the key alertMessage:

```
$('#showAlertButton').data('alertMessage','You have a changed message now.');
```

And then in the button's tap event, put this JavaScript:

```
alert( $('#showAlertButton').data('alertMessage') );
```