This plugin is used for accessing the contacts database of the device. In this tutorial we will show you how to create, query and delete contacts.

Step 1 - Install Contacts Plugin

```
cordova plugin add cordova-plugin-contacts
```

Step 2 - Adding Buttons

The button will be used for calling the **createContact** function. We will place it in the **div class = "app"** in **index.html** file.

```
<button id = "createContact">ADD CONTACT</button>
<button id = "deleteContact">DELETE CONTACT</button>
```

Step 2 - Add Event Listeners

Open **index.js** and copy the following code snippet into the **onDeviceReady** function.

```
document.getElementById("createContact").addEventListener("click",
createContact);
document.getElementById("deleteContact").addEventListener("click",
deleteContact);
```

Step 3A - Callback Function (navigator.contacts.create)

Our first callback function will call the **navigator.contacts.create** method where we can specify the new contact data. This will create a contact and assign it to the **myContact** variable but it will not be stored on the device. To store it, we need to call the **save** method and create success and error callback functions.

```
function createContact() {
   var myContact = navigator.contacts.create({"displayName": "Test
User"});
   myContact.save(contactSuccess, contactError);

   function contactSuccess() {
      alert("Contact is saved!");
   }

   function contactError(message) {
      alert('Failed because: ' + message);
   }

}
```

Step 3B - Callback Function (navigator.contacts.find)

Our second callback function will query all contacts. We will use the **navigator.contacts.find** method. The options object has filter parameter which is used to specify the search filter. **multiple = true** is used since we want to return all contacts from device. The **field** key to search contacts by **displayName** since we used it when saving contact.

After the options are set, we are using **find** method to query contacts. The alert message will be triggered for every contact that is found.

```
function findContacts() {
   var options = new ContactFindOptions();
   options.filter = "";
   options.multiple = true;
   fields = ["displayName"];
   navigator.contacts.find(fields, contactfindSuccess,
contactfindError, options);

   function contactfindSuccess(contacts) {
      for (var i = 0; i < contacts.length; i++) {
         alert("Display Name = " + contacts[i].displayName);
```

```
        }
    }

    function contactfindError(message) {
        alert('Failed because: ' + message);
    }

}
```

When we press the **FIND CONTACT** button, one alert popup will be triggered since we have saved only one contact.

Step 3C - Callback function (delete)

In this step, we will use the find method again but this time we will set different options. The **options.filter** is set to search that **Test User** which has to be deleted. After the **contactfindSuccess** callback function has returned the contact we want, we will delete it by using the **remove** method that requires its own success and error callbacks.

```
function deleteContact() {
    var options = new ContactFindOptions();
    options.filter = "Test User";
    options.multiple = false;
    fields = ["displayName"];
    navigator.contacts.find(fields, contactfindSuccess,
contactfindError, options);

    function contactfindSuccess(contacts) {
        var contact = contacts[0];
        contact.remove(contactRemoveSuccess, contactRemoveError);

        function contactRemoveSuccess(contact) {
            alert("Contact Deleted");
        }

        function contactRemoveError(message) {
            alert('Failed because: ' + message);
        }
    }

    function contactfindError(message) {
        alert('Failed because: ' + message);
    }

}
```

Write code for index.html file:-

```
<!DOCTYPE html>


<html>
    <head>


        <meta http-equiv="Content-Security-Policy"
content="default-src 'self' data: gap: https://ssl.gstatic.com
'unsafe-eval'; style-src 'self' 'unsafe-inline'; media-src *; img-
src 'self' data: content:;">
        <meta name="format-detection" content="telephone=no">
        <meta name="msapplication-tap-highlight" content="no">
        <meta name="viewport" content="initial-scale=1,
width=device-width, viewport-fit=cover">
        <link rel="stylesheet" type="text/css"
href="css/index.css">
        <title>Hello World</title>
    </head>
    <body>
        <div class="app">
          <button id = "createContact">ADD CONTACT</button>
<button id = "findContact">FIND CONTACT</button>
<button id = "deleteContact">DELETE CONTACT</button>
            <h1>Apache Cordova</h1>
            <div id="deviceready" class="blink">
                <p class="event listening">Connecting to Device</p>
                <p class="event received">Device is Ready</p>
            </div>
        </div>
        <script type="text/javascript" src="cordova.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
```

```
    </body>
</html>
```

Write code for index.js file:-

```
var app = {
    // Application Constructor
    initialize: function() {
        document.addEventListener('deviceready',
this.onDeviceReady.bind(this), false);
    },

    // deviceready Event Handler
    //
    // Bind any cordova events here. Common events are:
    // 'pause', 'resume', etc.
    onDeviceReady: function() {
        this.receivedEvent('deviceready');
document.getElementById("createContact").addEventListener("click",
createContact);
document.getElementById("findContact").addEventListener("click",
findContact);
document.getElementById("deleteContact").addEventListener("click",
deleteContact);
    },

    // Update DOM on a Received Event
    receivedEvent: function(id) {
```

```
        var parentElement = document.getElementById(id);
        var listeningElement =
parentElement.querySelector('.listening');
        var receivedElement =
parentElement.querySelector('.received');

        listeningElement.setAttribute('style', 'display:none;');
        receivedElement.setAttribute('style', 'display:block;');

        console.log('Received Event: ' + id);
    }
};

function createContact() {
   var myContact = navigator.contacts.create({"displayName":
"Tushar"});
   myContact.save(contactSuccess, contactError);

   function contactSuccess() {
      alert("Contact is saved!");
   }

   function contactError(message) {
      alert('Failed because: ' + message);
   }

}

function findContacts() {
```

```
   var options = new ContactFindOptions();
   options.filter = "";
   options.multiple = true;
   fields = ["displayName"];
   navigator.contacts.find(fields, contactfindSuccess,
contactfindError, options);

   function contactfindSuccess(contacts) {
      for (var i = 0; i < contacts.length; i++) {
         alert("Display Name = " + contacts[i].displayName);
      }
   }

   function contactfindError(message) {
      alert('Failed because: ' + message);
   }

}

function deleteContact() {
   var options = new ContactFindOptions();
   options.filter = "Tushar";
   options.multiple = false;
   fields = ["displayName"];
   navigator.contacts.find(fields, contactfindSuccess,
contactfindError, options);

   function contactfindSuccess(contacts) {
      var contact = contacts[0];
      contact.remove(contactRemoveSuccess, contactRemoveError);
```

```
        function contactRemoveSuccess(contact) {
            alert("Contact Deleted");
        }

        function contactRemoveError(message) {
            alert('Failed because: ' + message);
        }
    }

    function contactfindError(message) {
        alert('Failed because: ' + message);
    }

}
app.initialize();
```