Aura Component Controller.js – Detailed Explanation ----------

This JavaScript controller is for an Aura component with a lightning:datatable that loads Contacts, allows inline editing, and lets the user save or cancel changes.

1) doInit – Set up the table and fetch data ---------- - Define columns for the datatable: First Name, Last Name, Phone, Email (editable), and Account Name (read-only). - Calls the Apex method getContacts to retrieve a list of contacts from the server. - Processes the returned contacts to flatten Account.Name into a simple AccountName field for display. - Sets v.contacts with the processed list. - Uses $A.enqueueAction(action) to send the request.

2) handleSave – Persist inline edits ---------- - Reads pending edits from the datatable's draftValues parameter (only includes Id + changed fields). - Calls the Apex method saveContacts with the updated data. - On success: * Clears v.draftValues to remove edit markers. * Calls doInit again to reload the data from the server.

3) handleCancel – Discard changes ---------- - Clears v.draftValues to remove edits. - Calls doInit to reload fresh data from the server.

Apex Controller (Implied) ---------- Example: See code snippet below.

Key Points ---------- - Flatten related fields for display (Account.Name → AccountName). - v.columns, v.contacts, v.draftValues must be defined as attributes. - Always refresh the table after save or cancel to match the server data. - Consider adding toast messages and spinners for better UX.

JavaScript Controller Code:

```
({
    doInit: function(component, event, helper) {
        component.set('v.columns', [
            {label: 'First Name', fieldName: 'FirstName', type: 'text', editable: true},
            {label: 'Last Name', fieldName: 'LastName', type: 'text', editable: true},
            {label: 'Phone', fieldName: 'Phone', type: 'phone', editable: true},
            {label: 'Email', fieldName: 'Email', type: 'email', editable: true},
            {label: 'Account Name', fieldName: 'AccountName', type: 'text'}
        ]);
        var action = component.get("c.getContacts");
        action.setCallback(this, function(response) {
            if (response.getState() === "SUCCESS") {
                let contacts = response.getReturnValue();
                contacts.forEach(function(c) {
                    c.AccountName = c.Account ? c.Account.Name : '';
                });
                component.set("v.contacts", contacts);
            } else {
                console.error("Error loading contacts:", response.getError());
            }
        });
        $A.enqueueAction(action);
    },

    handleSave: function(component, event, helper) {
        const draftValues = event.getParam('draftValues');
        var action = component.get("c.saveContacts");
        action.setParams({ updatedContacts: draftValues });
        action.setCallback(this, function(response) {
            if (response.getState() === "SUCCESS") {
                component.set("v.draftValues", []);
                this.doInit(component);
            } else {
                console.error("Error saving contacts:", response.getError());
            }
        });
        $A.enqueueAction(action);
    },

    handleCancel: function(component, event, helper) {
        component.set("v.draftValues", []);
        this.doInit(component);
    }
})
```

Apex Controller Code:

```
public with sharing class ContactController {
    @AuraEnabled(cacheable=true)
    public static List<Contact> getContacts() {
        return [
            SELECT Id, FirstName, LastName, Phone, Email, Account.Name
            FROM Contact
            LIMIT 200
        ];
    }

    @AuraEnabled
    public static void saveContacts(List<Contact> updatedContacts) {
        update updatedContacts;
    }
}
```