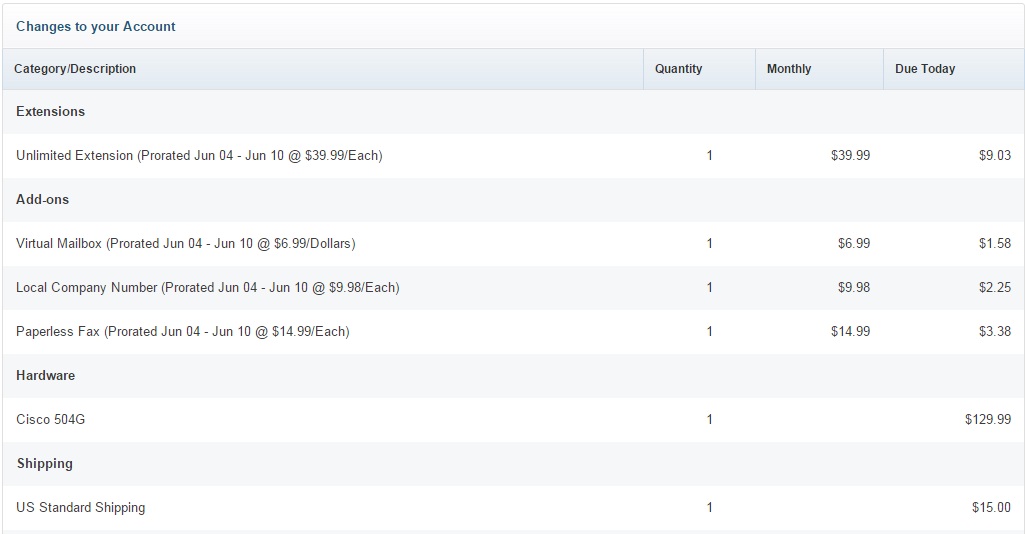
##### **Sorting a grouped table with jQuery**

                jQuery tables like jqGrid and others libraries and plugins provides sorting capabilities by default, but when is about tables which rows are grouped, this is not enough and then a custom sorting behavior is needed.



                        This was the problem I faced few days ago, where I had a table grouped by categories, then I had to override the sorting behavior to sort alphabetically by categories, and then order the items inside every category. Here is the code to accomplish this.

#### Overriding the default click event in the header of the table

                        The first thing I did was remove the default onclick event associated to that jQuery table, and instead attach my own onclick to the header of the table.

$(function(){

$(".reviewTable tr:first").children().unbind('click');

$(".reviewTable tr:first").click(function(){customSort();});

});

#### Preparing the JSP

                        For better understanding I will show you the structure of my JSP page and the modifications I did to accomplish my objective.

<c:foreach items="${services}" var="serviceGroup">

<b>${serviceGroup.key}</b>

<c:foreach items="${serviceGroup.value.items}" var="service">

${service.productName} ${service.computedDescription}

${service.quantity}

${service.computedMonthlyFee}

<fmt:formatnumber type="currency" value="${service.computedChargeAmount}"></fmt:formatnumber>

</c:foreach>

</c:foreach>

                       Here we have one outer foreach loop for iterate the categories, and then an inner foreach loop for iterating the items associated to every category.   
Then we need a way to know later which rows are parent ("categories") and which ones are children ("items"), to accomplish this we use the data element, specifying in the parent elements data-parent="parent". I will need also to know which items are associated to an specific parent, for this I create a new data element: "data-order", where in the case of the parents it will have just the name of the category: data-order="${serviceGroup.key}", and in the children it will have the category + the name of the item, it would be like this: data-order="${serviceGroup.key}-${service.productName}". Whith this ready, then we are ready to get hands dirty in jQuery code:

#### jQuery code:

var flag = 1;

function customSort(){

var trs = $(".reviewTable tr[data-order]");

var parents = $(trs).filter(function(){ return $(this).data('parent') === 'parent';});

parents = $(parents).sort(function(a,b){

return flag \* $(b).data('order').localeCompare($(a).data('order'));

});

var index = 0;

$(".reviewTable tbody tr").remove();

$(parents).each(function(){

insert($(this), index++);

var parentData = $(this).data('order');

var children = $(trs).filter(

function(){

return $(this).data('parent') !== 'parent';

}).filter(

function(){

return $(this).data('order').match('^' + parentData);

}).sort(function(a,b){

return flag \* $(b).data('order').localeCompare($(a).data('order'));

});

$(children).each(function(){

insert($(this), index++);

});

});

flag = flag \* -1;

}

                      This is the whole jQuery function I wrote for my custom sorting, let explain step by step for better understanding, in the first line I take all the "sort-able" rows of the table, that is to say, all the rows with data-order element. And in the second line I apply a filter over that list to take just those who are parents.

var trs = $(".reviewTable tr[data-order]");

var parents = $(trs).filter(function(){ return $(this).data('parent') === 'parent';});

                      Second step is sort my list of parents, for this we use the function "sort" of jQuery, and the criteria used for sorting is the value of the element data-order, compared by the "localeCompare" function, which give me the alphabetical order, and then multiply it by "flag", which give me the result depending on the order I am sorting.

parents = $(parents).sort(function(a,b){

return flag \* $(b).data('order').localeCompare($(a).data('order'));

});

                      Once I have all the data of the table stored in my vars, then I remove all the rows of the table:

$(".reviewTable tbody tr").remove();

                      Then we are ready to start inserting the elements in the table, sorted by parents and then by children,

$(parents).each(function(){

insert($(this), index++);

var parentData = $(this).data('order');

var children = $(trs).filter(

function(){

return $(this).data('parent') !== 'parent';

}).filter(

function(){

return $(this).data('order').match('^' + parentData);

}).sort(function(a,b){

return flag \* $(b).data('order').localeCompare($(a).data('order'));

});

$(children).each(function(){

insert($(this), index++);

});

});

flag = flag \* -1;

                      All I do here is iterate the list of parents, then for each element I insert the parent itself, then filter the list of children, taking just the ones associated to the current parent and ordering them, and once ordered, I insert them in the table too.   
And as last step I switch the var "flag", so when you click in the header of the table next time, it orders the elements in the opposite way.

#### Conclusions

                      And that's it, probably there is some "magical" solution out there to do this, but this is the one I gave to my problem, and I hope it works for you too, if you have some variant of the same problem, let me know and I will be glad to help you. See you in the next hacking...