DATA PARSING AND DATA ENTRY TECHNIQUES

Learning from my "Chart City" Prototype

by Jeremy Fonte

Many web apps utilize local files from a user's computer, typically using an upload control or file input element. But what about copying and pasting data into a web app? Can the meaning of the data be extracted from a simple pasting action?

In the case of spreadsheets and tables, the answer is yes, copying and pasting can provide all of the necessary data for a web app to parse the spreadsheet selection. When you copy and paste a spreadsheet into a <textarea> element, the spreadsheet is essentially pasted as a TDF or tab-delimited file. This is similar to a CSV or comma-separated value file, but it uses tabs instead of commas to separate each column in any given row.

The <textarea>'s contents can be accessed using something like the following jQuery syntax:

```
$('textarea#myText').val();
```

Once you have the value of the <textarea>, you can split it using the string split function: first, split the value by "\n", the newline character, which will give you an array of lines (or rows) of text; then, split each element of the array by "\t", the tab character, which will give you all of the columns or individual pieces of data for the current row. Repeat this tab splitting for all rows, using a loop, and you'll end up with all of the spreadsheet data easily accessible in a nested array (equivalent to a two-dimensional array).

This data can then be copied to unordered lists of unordered lists – basically, each row is an unordered list of <li> elements that have their CSS set to "float: left", and all of the rows are in turn <li> elements in the master unordered list. Both of the <ul> elements should have their "list-style" set to "none", and all margins and padding should be removed. This will basically look like this:

```
<ul>
        <li>
                <ul>
                        <li></li>
                        <li></li>
                </ul>
        </li>
</ul>
```

Now we have a way to display the spreadsheet data that was pasted in – but what about letting the user edit the data, or perform new data entry? This is where I had the idea of using <input type="text"> elements to make an online spreadsheet.

When the end-user clicks on any given <li> data cell, the text value of the node should be saved. Then, an <input type="text"> element should be placed within the list item, and it's value should be set to the saved text value of the <li> data cell. The end user can now edit the value, and when they're

finished and click elsewhere, the input element's data should be saved, the input element should be removed, and finally, the input element's value should be written to the <li> as text.

To see all of this in action, check out Chart City at http://fonte.me/ChartCity