table-comp

table-comp is a Vue.js (version >= 2.5) component with many features including:

- variable column widths and alignments
- built-in vertical scrolling
- various css styling variables
- modify an individual cell's classList
- responsive event notification to the component parent on row/cell clicks

table-comp can be installed via with the included `package.json` file for a local installation via the npm install command. **table-comp** depends on the vue.js framework. A demo folder is provided that used Parcel together with its associated `package.json` file to bundle together **table-comp** along with its vue.js dependency for a simple application. Further details are provided below for running the demo.

Props

A prop in Vue.js is a custom attribute for passing information from a parent component hosting **table-comp** instance(s) to a **table-comp** as a child component.

table-comp has the following props for a parent to bind to child:

`rows` -- an array of table rows where each table cell is itself an array containing the cell's value and class value (see below)

`title` -- a title to be placed above the table (string, default: null)

`table_height` -- the overall height of the table in pixels (number, 300)

`headings` -- the variable headings that appear at the top of the table (array, default: null)

`column_widths` -- the column widths (in pixels) for each of the variables (array of numbers, default: [120,120,...]

`css_variables` -- defines the css variables for **table-comp** (object, default: null)

Note that if the total number of row heights exceeds the `table_height` then a vertical scroll bar will automatically appear. In the horizontal direction, keep in mind that there is no horizontal scroll bar as you add columns. For a table with numerous columns use the WorkSheetComp component.

Styling

The **css_variables** prop is a javascript object that contains any combination of css variable names as keys and associated values. The following list are the css variable names along with their default values:

```
{
    table_comp_font_family: 'Verdana,serif',
    table_comp_title_font_size: '1rem',
    table_comp_title_color: 'black',

    table_comp_thead_color: 'black',
```

```
      table_comp_thead_border_bottom: '1px solid black',
      table_comp_thead_background: 'linear-gradient(to bottom, #f5f6f6
0%,#dbdce2 21%,#b8bac6 49%,#dddfe3 80%,#f5f6f6 100%)',
      table_comp_thead_text_align: 'left',

      table_comp_row_color: 'black',
      table_comp_row_selected_color: 'green',
      table_comp_row_border_bottom: '1px solid black',
      table_comp_row_odd_background: 'linear-gradient(to bottom, #cedce7
0%,#596a72 100%)',
      table_comp_row_even_background: 'linear-gradient(to bottom, #cedce7
0%,#596a72 100%)',
      table_comp_row_hover_color: 'linear-gradient(to bottom, #b4e391 0%,#61c419
50%,#b4e391 100%)',
      table_comp_cell_font_size: '.75rem'
  }
```

As an example you could bind from the parent the following object to the `css_variables` prop for setting the title font size and the header color of **table-comp**: {table_comp_title_font_size: '24px', table_comp_thead_color: 'purple'}

Note that multiple **table-comp** children of the parent can each be bound to a unique set of `css_variable` prop objects. To enforce the same styling across all **table-comp** children, simply bind just one `css_variable` prop object to all the **table-comp** children.

Events

table-comp has events that notify the parent component when a table row or cell is clicked.

table-comp emits the following named events:

```
'table_comp_row' -- returns an object with selected row index, array of the cell
values in the selected row, and an array of the css classes for each cell

'table_comp_cell' -- returns an object with the selected cell's row and column
index,value,and class
```

The parent component can listen to these events and provide a callback for further processing. Events emitted from a child component back to the parent is explained at Vue Custom Events.

Demonstration

A demonstration of **table-comp** is provided in the folder named `demo/dist` and can be viewed by hosting the `index.html` file. The demo was templated from the `App.vue` file.

The demo ---

- sets the table rows
- sets a restricted table height so the vertical scroll bar appears
- dynamically sets classes for cells
- dynamically add/delete rows
- defines a bunch of css variables
- listens for row/cell clicks and displays the values to console.log

As a suggestion, install http-server locally/globally via npm then enter the command `http-server` in the **table-comp** `dist` directory. From a browser enter the url: `localhost:8080/` to view the demo.

Here is some example code for using **table-comp** taken from `App.vue`'s template:

```
<div class="demo_container">
   <div class="buttons_box">
     <button v-on:click="check_cell">Check Cell [3,1]</button>
     <button v-on:click="change_snake_class">Set class of 'snake'</button>
     <button v-on:click="delete_row">Delete 'bird'</button>
     <button v-on:click="add_row">Add 'goldfish'</button>
   </div>
   <table-comp
       title="Pets (Add/Delete/Change cell style)"
       :rows="rows"
       :headings="headings"
       :table_height="table_height"
       :column_widths="column_widths"
       :css_variables="css_variables"
       v-on:table_comp_row="show_row"
       v-on:table_comp_cell="show_cell">
   </table-comp>
 </div>
```

And the supporting data references:

```
   data: function(){
     return {
       headings: ['My Pets', 'Favorite Food', "Pet's Name", "Pet's Age"],
       rows: [
         [['dog',''], ['biscuits',''], ['Fido',''], [7.09,'']],
         [['cat',''], ['fish',''], ['Pussy',''], [3.345,'']],
         [['snake',''], ['rats',''], ['Sneaky',''], [10,'']],
         [['hamster',''], ['popcorn',''], ['Tumbler',''], [5.60,'']],
         [['bird',''], ['seeds',''], ['Sweety',''], [12,'']],
         [['turtle',''], ['lettuce',''], ['Speedy',''], [20.33,'']],
         [['pig',''], ['potatoes',''], ['Porky',''], [13.5,'']],
         [['cow',''], ['grass',''], ['Molly',''], [14,'']]
       ],
       table_height: 80,
       column_widths: [100,160,140,110],
       css_variables: {
         table_comp_title_color: 'white',
         table_comp_thead_color: 'white',
         table_comp_thead_background: 'black',
         table_comp_thead_border_bottom: '2px solid white',
         table_comp_row_color: 'gray',
         table_comp_row_selected_color: 'gold',
         table_comp_row_border_bottom: '1px solid white',
         table_comp_row_odd_background: 'black',
         table_comp_row_even_background: 'black'
       }
     };
   },
   components: {
```

```
      TableComp
    },
    methods: {
      add_row: function(){
        this.rows.push([['goldfish',''],['fish food',''],['Goldie',''],
[4.3,'']]);
      },
      delete_row: function(){
        this.rows.splice(4,1);
      },
      //parent to child - modify a cell class
      check_cell: function(){
        this.rows[3].splice(1,1,[this.rows[3][1][0],'cell_check']);
      },
      change_snake_class: function(){
        this.rows[2].splice(0,1,['snake','snake_cell']);
      },
      //child to parent -- respond to row event
      show_row: function(obj){
        console.log(`Row clicked: ${obj.row_values}`);
      },
      //child to parent -- respond to cell event
      show_cell: function(obj){
        console.log(`Cell clicked:
        Row Index: ${obj.row_index}
        Cell Column: ${obj.col_index}
        Cell Value: ${obj.cell_value}
        Cell Class: ${obj.cell_class}`);
    }
```

Note how the cell's for the table are defined. Each cell is a 2 element array with the cell's value and a string referencing the name of a class defined in the application's css. We've initialized all the cells with a empty string for the class. The two methods `check_cell` and `change_snake_class` dynamically assign class names in response to a button click. Note the use of `splice` in order for the component to react to the binding.