infinite-scroller-comp

infinite-scroller-comp is a Vue.js (>=2.5) web component that provides a content area and an associated vertical scroll bar. The developer can specify an array of content and receive an event notice when the vertical scroll moves to the bottom of the array content. Responding to the event by pushing an additional item to the array causes reactivity on the display and a repetitive cycle of scrolling, pushing, and display update.

infinite-scroller-comp can be installed via with the included `package.json` file for a local installation via the npm install command. **infinite-scroller-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 **infinite-scroller-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 **infinite-scroller-comp** instance(s) to an **infinite-scroller-comp** as a child component. **infinite-scroller-comp** has the following prop for a parent to bind and send information to:

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

Styling

The **css_variables** prop is a javascript object that contains a css variable name as keys and associated values. The following is the css variable name along with its default value for a quick styling of **infinite-scroller-comp**:

```
{
    scroller_height: '30rem'
}
```

Events

infinite-scroller-comp has one event that notifies the parent component bottom of the vertical scroll is reached.

infinite-scroller-comp emits the following single named event:

```
'reached_bottom' -- returns a boolean of true if scroller is at the bottom,
otherwise returns false
```

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

The demonstration shows how the event can be incorporated.

Demonstration

One demonstration of **infinite-scroller-comp** is provided in the folder named `dist`. The demo (templated in the `App.vue` file) can viewed by hosting the `index.html` file.

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

Here is some example code for using **infinite-scroller-comp** taken from the `App.vue` file:

```
    <infinite-scroller-comp
        v-on:reached_bottom="add_person"
        :css_variables="css_variables">
        <div slot="content">
          <person-comp
            v-for="(person,index) in persons"
            :key="index"
            :img_source_url="person.picture.large"
            :first_name="person.name.first"
            :last_name="person.name.last"
            :birth_date="person.dob.date"
            :age="person.dob.age"
            :location_city="person.location.city"
            :location_state="person.location.state">
          </person-comp>
        </div>
      </infinite-scroller-comp>
```

Note that we are using `infinite-scroller-comp`'s slot named 'content' to insert an array of `person-comp` web components. Also note that the method `add_person` will respond to `infinite-scroller-comp`'s event 'reached-bottom' . If the event returns `true` then we will add another person to the `persons` array. By pushing another person the display will be reactivity updated and the vertical scroll bar will move up a notch.

The supporting data references:

```
data: function() {
    return {
      url: 'https://randomuser.me/api/',
      persons: null,
      css_variables: {
        scroller_height: '35rem'
      }
    }
  },
```

The supporting methods:

```
methods: {
    add_person(at_bottom){
      if(at_bottom){
        const config = {
          method: 'GET',
          mode: 'cors'
        };
        fetch(this.url,config).then((response) =>{
```

```
          if(response.ok){
            return response.text();
          }
          throw new Error(response.statusText);
        }).then((resp_str) => {
          const person_obj = JSON.parse(resp_str);
          this.persons.push(person_obj.results[0]);
        }).catch((error) => {
          console.log(error.message);
        })
      }
    }
  },
  mounted() {
    this.persons = [];
      try {
        const config = {
          method: 'GET',
          mode: 'cors'
        };
        for(let i = 0; i < 5; i++){
          fetch(this.url,config).then(response => {
            if(response.ok){
              return response.text();
            }
            throw new Error(response.statusText);
          }).then(resp_str => {
            const person_obj = JSON.parse(resp_str);
            this.persons.push(person_obj.results[0]);
          });
        }
      }catch(error){
        console.log(error.message);
      }
    }
    initialize_data();
  }
```

Note the we initialize the `persons` array with 5 person objects at the `mounted` stage of Vue.

Also Note the event responder `add_person` where if the event value of `at_bottom` is `true` then fetch another person and push it onto the `persons` array to start reactivity of the display.