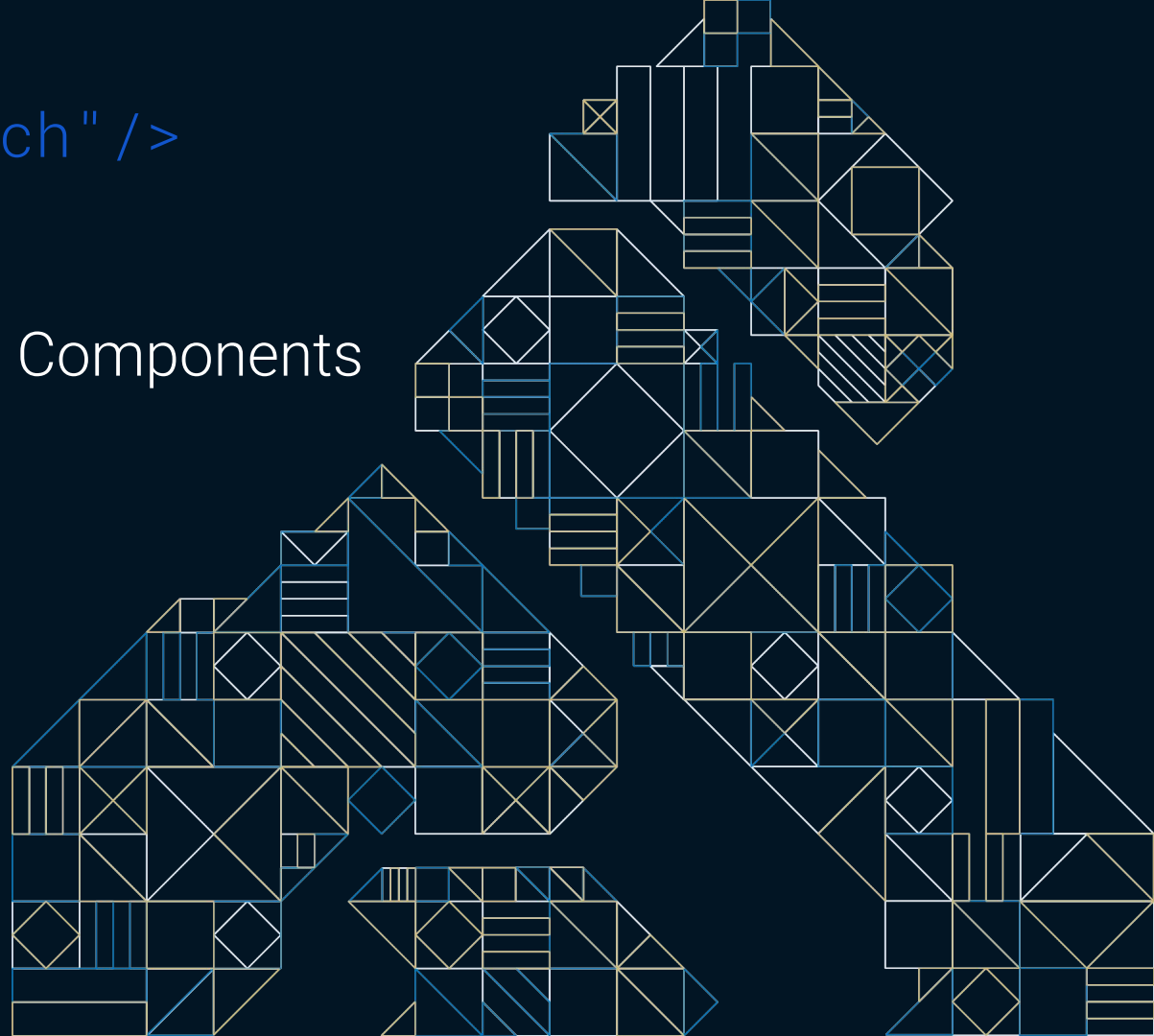
```
<input type="search"/>
<ul>
 <li>
```

Building Autosuggest Components

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

Darren Jennings

June 2019



Kong

$ whoami *darrenjennings*

https://guuu.io

🚨 ✨ https://konghq.com/careers/ ✨

1. What is an autosuggest component? 🤔
2. Vue-autosuggest 🔍🧙
3. Features/interactions 👇⌨️
4. Data Fetching 📊
5. Accessibility ♿
6. Styling 💅

1. What is autosuggest?

ARIA 1.1 Combobox with Listbox Popup https://www.w3.org/TR/wai-aria-practices/#combobox

§ 3.8 Combo Box

A combobox is a widget made up of the combination of two distinct elements: 1) a single-line textbox, and 2) an associated pop-up element for helping users set the value of the textbox. The popup may be a listbox, grid, tree, or dialog. Many implementations also include a third optional element -- a graphical button adjacent to the textbox, indicating the availability of the popup. Activating the button displays the popup if suggestions are available. The popup is hidden by default, and the conditions that trigger its display are specific to each implementation. Some possible popup display conditions include:

- It is displayed only if a certain number of characters are typed in the textbox and those characters match some portion of one of the suggested values.
- It is displayed as soon as the textbox is focused, even if the textbox is empty.
- It is displayed when the Down Arrow key is pressed or the show button is activated, possibly with a dependency on the content of the textbox.
- It is displayed if the value of the textbox is altered in a way that creates one or more partial matches to a suggested value.

The nature of the suggested values and the way the suggestions are presented is called the autocomplete behavior. Comboboxes can have one of four forms of autocomplete:

1. **No autocomplete:** When the popup is triggered, the suggested values it contains are the same regardless of the characters typed in the textbox. For example, the popup suggests a set of recently entered values, and the suggestions do not change as the user types.
2. **List autocomplete with manual selection:** When the popup is triggered, it presents suggested values that complete or logically correspond to the characters typed in the textbox. The character string the user has typed will become the value of the textbox unless the user selects a value in the popup.
3. **List autocomplete with automatic selection:** When the popup is triggered, it presents suggested values that complete or logically correspond to the characters typed in the textbox, and the first suggestion is automatically highlighted as selected. The automatically selected suggestion becomes the value of the textbox when the combobox loses focus unless the user chooses a different suggestion or changes the character string in the textbox.
4. **List with inline autocomplete:** This is the same as list with automatic selection with one additional feature. The portion of the selected suggestion that has not been typed by the user, a completion string, appears inline after the input cursor in the textbox. The inline completion string is visually highlighted and has a selected state.

SO WHAT DO I DO?

1.a Building blocks: Autosuggest using HTML elements

`<input type="text"/>`

- autocomplete="on,off" type="search", accessibility (a11y) WAI-ARIA attributes

`<ul>, <ol>`

- unordered/ordered list. UL = bullets, OL = numbers

`<li>`

- list items

`<datalist>`

- Built in autocomplete? https://codepen.io/darrenjennings/pen/zQgyBw

```
<input list="frameworks">
<datalist id="frameworks">
  <option>React</option>
  <option>Vue</option>
  <option>Angular</option>
  <option>Svelte</option>
</datalist>
```

You vs. the DOM

2. Vue-autosuggest

https://github.com/darrenjennings/vue-autosuggest

- Handles all the hard stuff (keyboard events, accessibility, etc.)
- WAI-ARIA complete autosuggest component built in Vue
- Full control over rendering with built in defaults or custom components for rendering
- Easily integrate data fetching
- Supports multiple sections
- No opinions on CSS, full control over styling

$ yarn add vue-autosuggest
$ npm install vue-autosuggest

```
<div id="autosuggest">
  <input type="text"
    autocomplete="off"
    role="combobox"
    aria-autocomplete="list"
    aria-owns="autosuggest__results"
    aria-activedescendant="autosuggest__results-item--2"
    aria-haspopup="true"
    aria-expanded="true"
    id="autosuggest__input"
    placeholder="Type 'G'"
    value="phonics"
    class="autosuggest__input--open">
  <div class="autosuggest__results-container">
    <div aria-labelledby="autosuggest" class="autosuggest__results">
      <ul role="listbox" aria-labelledby="autosuggest">
        <li role="option" data-suggestion-index="0" data-section-name="default" id="autosuggest__results-item--0"
class="autosuggest__results-item">
          clifford kits
        </li>
        <li role="option" data-suggestion-index="1" data-section-name="default" id="autosuggest__results-item--1"
class="autosuggest__results-item">
          friendly chemistry
        </li>
        <li role="option" data-suggestion-index="2" data-section-name="default" id="autosuggest__results-item--2"
class="autosuggest__results-item">
          phonics
        </li>
        <li role="option" data-suggestion-index="3" data-section-name="default" id="autosuggest__results-item--3"
class="autosuggest__results-item">
          life of fred
        </li>
        <li role="option" data-suggestion-index="4" data-section-name="default" id="autosuggest__results-item--4"
class="autosuggest__results-item">
          life of fred math
        </li>
      </ul>
    </div>
  </div>
</div>
```

input

listbox

https://darrenjennings.github.io/vue-autosuggest/

3. Features - multiple sections / slots

Pseudocode html output of vue-autosuggest:

```
<div>
  <span class="before-results />
  <input />
  <ul>
    <li class="before-section-users">Human</li>
    <li>Angamite</li>
    <li>Egalmoth</li>
    <li>Frumgar</li>
  </ul>
  <ul>
    <li class="before-section-elf">Elf</li>
    <li>Galadriel</li>
    <li>Galdor of the Havens</li>
    <li>Galion</li>
  </ul>
  <span class="after-results"v-if="currentState === 'PENDING'">Loading...</span>
</div>
```

3. multiple sections cont.

```
sectionConfigs: {
  destinations: {
    limit: 6,
    label: "Destination",
    onSelected: selected => {
      this.selected = selected.item;
    }
  },
  hotels: {
    limit: 6,
    label: "Hotels",
    onSelected: selected => {
      this.selected = selected.item;
    }
  },
  default: {
    onSelected: selected => {
      console.log(
      this.doSearch(selected.item);
    }
  }
}
```

```
<vue-autosuggest
  v-model="query"
  @input="fetchResults"
  :component-attr-id-autosuggest="'unique-autosuggest-value-for-aria'"
  :suggestions="suggestions"
  :inputProps="{ id: 'autosuggest__input', placeholder: 'Search...' }"
  :sectionConfigs="sectionConfigs"
  :getSuggestionValue="getSuggestionValue"
  :should-render-suggestions="(size, loading) => size >= 0 && !loading"
>
  <template #header="{ suggestion }">
    <div
      v-if="suggestions.length === 0"
      class="autosuggest__results_message"
    >
      No Results
    </div>
  </template>

  <template #default="{ suggestion }">
    <div v-if="suggestion.name == 'hotels'">
      <img
        :class="avatar"
        :alt="`Avatar of suggestion.item.name`"
        :src="suggestion.item.thumbnailUrl"
      />
      {{ suggestion.item.title }}
    </div>
    <div v-else>
      {{ suggestion.item.name }}
    </div>
  </template>
</vue-autosuggest>
```

3. Features - mouse/keyboard interactions

➔ @click 

- input
  - show suggestions or wait until user starts typing? - this is optional but could be useful e.g. show some suggestions based on “most popular”
- Result item
  - selects a result from the listbox, emits @selected event
- document
  - Close listbox

➔ Keydown 

- Character keys
  - Show listbox, filter the results!
- Exclude non-character keys such as shift, tab, alt/option, OS Keys
- down, up (keycodes: `40`, `38`)
  - Key through results in the listbox
  - Input changes to preview what is being selected. Keying up to the top resets the input value back to what was originally typed
  - Items are always visible (scrollTo)
  - Keying down when reach last item resets to the first item
  - Update aria attributes
- Enter (13)
  - Select a result item from the listbox
- Escape (27)
  - Close listbox
  - If user closes listbox, allow them to press a key to reopen the listbox to continue checking results

4. Data Fetching

- **Developer Experience (DX)** - developer user that is using vue-autosuggest to build an app
  - Separation of concerns (SoC)
    - Keep the combobox worried about what it needs to worry about.
      - List presentation, user interactions, ARIA state, etc.
  - Developers want to fetch their own data (Fetch API, axios, 3rd party api client libraries etc.)
- **User Experience (UX)** - user of app that is using vue-autosuggest
  - Give user feedback of loading state, so that they're not frustrated by why it's not showing up
    - Right side spinner
    - "Loading…" text as result item
  - Debounce API calls `@input`
    - Wait until user has stopped typing before firing XHR requests (but not too long < 200ms)
  - Ask the question: is the listbox needed for functionality or can you "just search"? E.g. google

Third Party Data Fetching

Algolia: https://codesandbox.io/s/mystifying-cray-2vr87nyvkp

https://drive.google.com/open?id=1C_yCV_AfHyeiEWcUhw99fYFVLqqmIaH5

```
<ais-instant-search :search-client="searchClient" index-name="demo_ecommerce">
  <ais-configure :restrictSearchableAttributes="['name']"/>
  <ais-autocomplete v-slot="{ currentRefinement, indices, refine }">
    <vue-autosuggest
      v-model="query"
      v-slot="{ suggestion }"
      :suggestions="indicesToSuggestions(indices)"
      @input="refine"
      @selected="onSelect"
      :input-props="{id: 'autosuggest-input', autocomplete: 'off', type: 'search'"
    >
      <ais-highlight :hit="suggestion.item" attribute="name" v-if="suggestion.item.name"/>
      <strong>$ {{ suggestion.item.price }}</strong>
      <img :src="suggestion.item.image">
    </vue-autosuggest>
  </ais-autocomplete>
</ais-instant-search>
```

Data Fetching

https://drive.google.com/open?id=1GlYDOu-MEcFWYOHSVAtpSWD7iRFdi9LO

5. Accessibility (A11y)

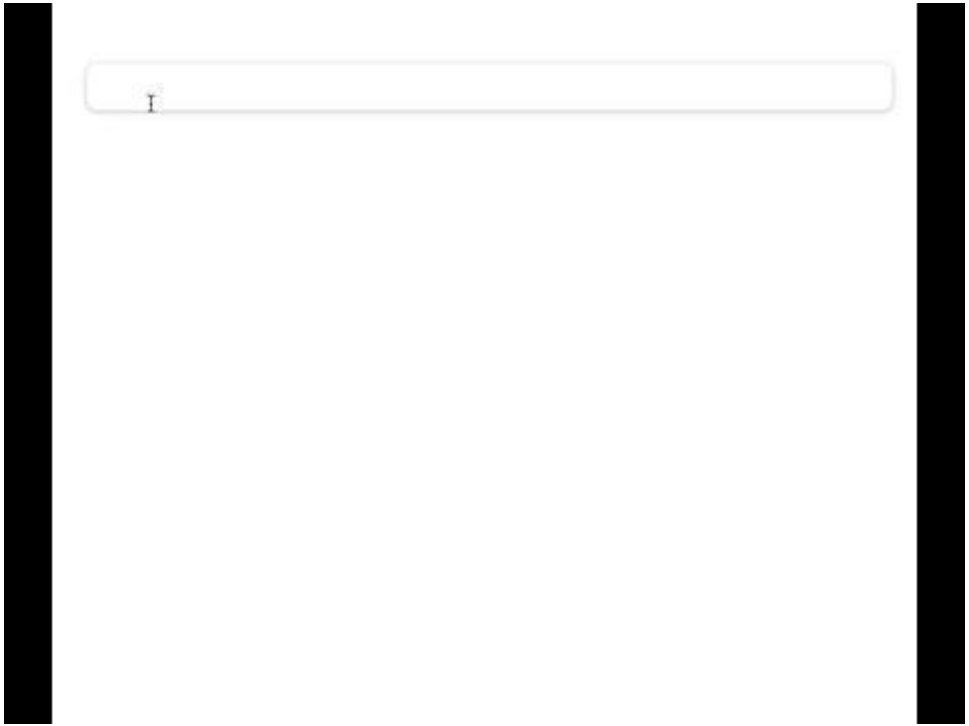
```
<div id=”autosuggest”>
  <input
    type="text"
    autocomplete="off"
    role="combobox"
    aria-autocomplete="list"
    aria-owns="autosuggest__results"
    aria-activedescendant=""
    aria-haspopup="true/false"
    aria-expanded="true/false"
    id="autosuggest__input"
    placeholder="Type your search here”>
  <ul role="listbox" aria-labelledby="autosuggest">
    <li
      role="option"
      data-suggestion-index="0"
      data-section-name="default"
      id="autosuggest__results-item--0"
      class="autosuggest__results-item"
    >
      Aragorn
    </li>
  </ul>
</div>
```

- Should be screen reader friendly
  - Aria attributes reflect state of listbox
  - WAI-ARIA complete (Web Accessibility Initiative – Accessible Rich Internet Applications)
    - https://www.w3.org/TR/wai-aria-1.1/
      - Big document on how to author components to be accessible to specifications
- keyboard interactions
  - Intuitive
  - follow specs
  - customizable

6. Styling

https://drive.google.com/open?id=16eb6q2xG5sM9OqWeiYiYJCFPSwKPxqF9

DEMO



- Allow scroll
  - `overflow-y`
- Always plan for the states, e.g. the fetch machine
  - Idle
  - Pending
  - Reject
  - Resolve
    - Results / no results
- Result items should be able to allow any html and auto adjust height accordingly
- The chrome “omnibox”
  - Listbox results to be an extension of the input
  - Inlines important results
  - Ghost riding the input
  - No loading state?! Ask the question: Is the autosuggest needed for functionality?

Thank you