Vuetify 3.x

https://vuetifyjs.com/en/

Vuetify Online Documentation

- Overview
  - Material Design
  - Vuetify version 2.6.x does not support Vue 3.x
  - Vuetify version 3.0 (Oct 22) supports Vue 3.0
  - Latest version 3.1.14 (April 2023)
- Styles, Typography
  - (Material) Color Names as class names
- UI Components
  - Code snippet showing both <template> and <script>
  - Detailed Element API (properties to HTML attributes and VueJS binding)

Why Vuetify?

NPM Trends

Getting Started

```
$ npm create vue@3
? Project name: first-vuetify
? Add TypeScript? No / Yes
? Add JSX Support? No / Yes
? Add Pinia for state management? No / Yes
? Add Vitest for Unit Testing? No / Yes
? Add an End-to-End Testing Solution?
    No
    Cypress
    Playwright
? Add ESLint for code quality? No / Yes
? Add Prettier for code formatting? No / Yes
$ cd first-vuetify
$ npm install --save vuetify   # Confirm version 3.1.x
$ npm i vite-plugin-vuetify
```


src/main.ts

```
import { createApp } from 'vue'
import { createVuetify } from "vuetify"
import App from './App.vue'
import './assets/main.css'
import 'vuetify/styles'
import * as components from "vuetify/components"
import * as directives from "vuetify/directives"

const vuetify = createVuetify({components, directives})
createApp(App)
  .use(vuetify)
  .mount('#app')
```

Using Icon Fonts

```
npm install @mdi/font -D
```

```
// src/main.ts
import “@mdi/font/css/materialdesignicons.css”
import { aliases, mdi} from “vuetify/iconsets/mdi”
const vuetify = createVuetify({
  icons: {
    defaultSet: “mdi”,
    aliases,
    sets: { mdi }
  }
})
```

```
<!-- using the icon in your UI -->
<v-icon icon=”mdi-cog” />
```

(see complete list of icons)

Vuetify built-in components

- Navigation
  - v-app-bar, v-bottom-navigation, v-footer, v-navigation-drawer, v-tabs, ...
- Containment
  - v-btn, v-card, v-chip, v-dialog, v-expansion-panel, v-list, v-menu, v-toolbar, ...
- Input, Selection, & Controls
  - v-autocomplete, v-checkbox, v-file-input, v-radio, v-select, v-slider, v-switch
  - v-btn-toggle, v-carousel, v-chip-group, v-slide-group
- Grid system
- Feedback
  - v-alert, v-badge, v-snackbar, v-rating, ...

Vuetify Element Names

TitleCase	kebab-case
`VAlert`	`v-alert`
`VNavigationDrawer`	`v-navigation-drawer`

Either style will work, but use them consistently throughout the code

Other UI Framework Alternatives

Framework	Component Names
Element UI	`<el-xxxx>`
Bootstrap Vue	`<b-xxxx>`
Quasar	`<q-xxxx>`
Iconic	`<ic-xxxx>`

- *These frameworks provide **components***
- *Both the **visual style** and **inner logic** (to some extent) of these components are customizable*

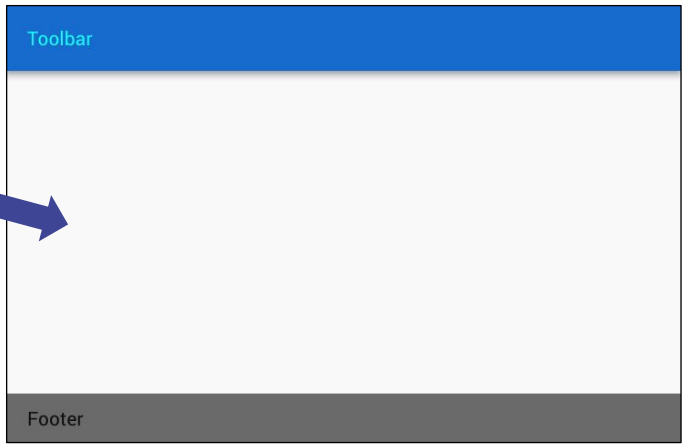
Framework	Component Names
Bootstrap	N/A
Tailwind CSS	N/A

These frameworks provide CSS classes ONLY for visual styling

Navigation Components

Typical Application Layout

```
<v-app>
   <v-app-bar>
     <span>Toolbar</span>
   </v-app-bar>
   <v-main>
    <!-- UI content goes here -->
   </v-main>
   <v-footer app>
     <span>Footer</span>
   </v-footer>
 </v-app>
```



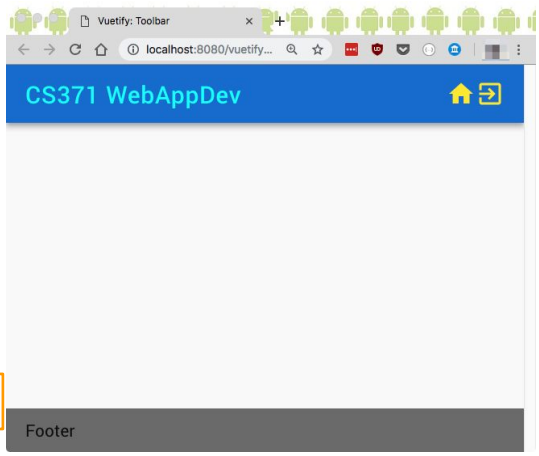
*Important: be sure to include the **app** attribute on v-footer> to pin it the application layout!*

Online Playground

<v-app-bar>

```
<v-app>
  <v-app-bar app>
    <v-app-bar-title>CS371 WebAppDev</v-app-bar-title>
    <v-spacer></v-spacer>
      <v-icon color="yellow" icon="mdi-home" />
      <v-icon color="yellow">mdi-exit-to-app</v-icon>
  </v-app-bar>
  <v-footer app color="grey">
   <span>Footer</span>
  </v-footer>
</v-app>
```

Material icon Name



Navigation Drawer

- <v-navigation-drawer> is a direct child of <v-app>
- Default: floating navigation drawer. When unhidden, the navigation drawer will float above the main content
- Permanent floating navigation drawer: when unhidden, the navigation drawer will push contents aside
- Online playground

Using Color Class Names

Color Usage	Class name	Example
Background Color	bg-{color}	`<div class=”bg-orange”> ... </div>`
Text Color	text-{color}	`<p class=”text-green” />...</p>`
Darker	bg-{color}-darken-{n} text-{color}-darken-{n}	`<div class=”bg-orange-darken-1”> ... </div>`
Lighter	bg-{color}-lighten-{n} text-{color}-lighten-{n}	`<div class=”bg-orange-lighten-1”> ... </div>`

Online playground

Input, Selection, and Controls

User Inputs

- v-autocomplete / v-combobox / v-select
- v-checkbox / v-switch
- v-radio-group and v-radio
- v-slider / v-range-slider
- v-form & v-text-field & v-textarea
- v-date-picker
- v-time-picker
- v-menu

Important attribute: `v-model`

Input Field Validation

```
<!-- in your <template> -->
<v-text-field label="Email"
  :rules="eRules"
/>
```

Online playground

```
<script setup lang="ts">
  const eRules = [
    function(x:string): boolean | string {
      if (x.indexOf("@") >= 1) return true;
      else return "Invalid email format";
    },
    function(x:string): boolean | string {
      if (x.endsWith(".edu") ||
        x.endsWith(".org") ||
        x.endsWith(".net")) return true;
      else return "Only .edu/.org/.net accepted";
    }
  ]
}
</script>
```

<v-slider> & <v-range-slider>

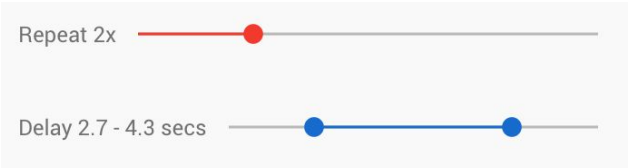
```
<v-slider v-model="repeat" color="red"
  :label="`Repeat ${repeat}x`" min="1" max="5" />

<v-range-slider v-model="delay"
  label="`Delay ${range}`" step="0.1" min="2"
  max="5" />
```

No parentheses in ${range} !!!

```
<script setup lang="ts">
  const repeat = ref(0);
  const delay = ref([2.7, 4.3])

  const range = computed(()): string {
    return this.delay[0].toFixed(1) + "-" +
           this.delay[1].toFixed(1) + "secs";
  })
</script>
```

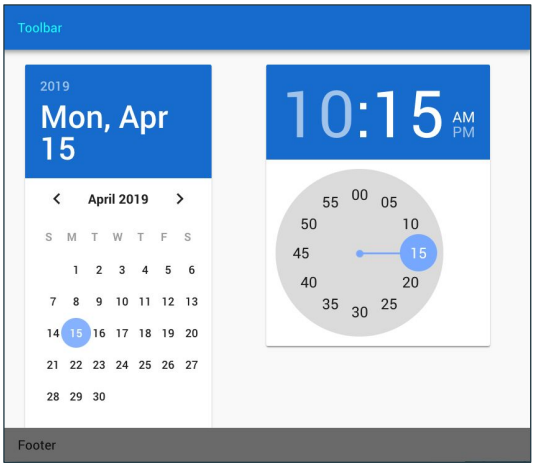


Online playground

Date & Time Pickers

```
<v-date-picker v-model="selDate" width="200" />
<v-time-picker v-model="selTime" width="200" />
```

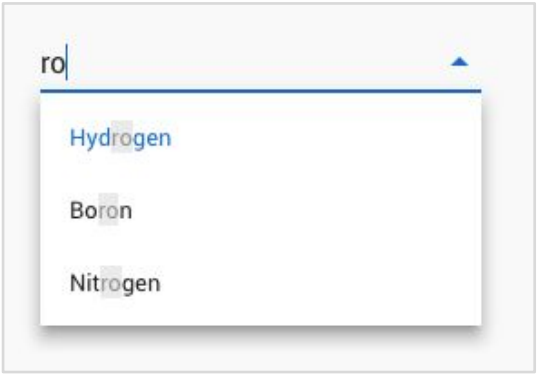
```
<script setup lang="ts">
  const selTime = ref(null); /* HH:mm */
  const selDate = ref(null);  /* yyyy-MM-dd */
</script>
```



<v-autocomplete> / <v-combobox> / <v-select>

```
<v-autocomplete v-model="atom" :items="atoms">
</v-autocomplete>
```

```
<script setup lang="ts">
  const atom = ref()
  const atoms = ref(["Hydrogen", "Lithium",
    "Sodium", "Potassium", "Rubidium",
    "Caesium", "Magnesium", "Calcium",
    "Barium", "Radium", "Boron", "Aluminium",
    "Gallium", "Carbon", "Silicon", "Tin",
    "Lead", "Nitrogen"]);
}
```



Online Foreground

<v-radio-group> & <v-radio>

```
<v-radio-group v-model="taxStatus"
  label="Tax Filing Status">
  <v-radio v-for="(s,opt) in taxFiling"
    :label="s" :value="opt" />
</v-radio-group>
```

```
<script setup lang="ts">
  const taxStatus = ref(0);
  const taxFiling = ref([
    "Single", "Married Filing Jointly",
    "Married Filing Separately",
    "Head of Household"
  ]);
}
```

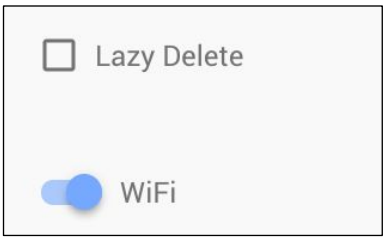
Online playground

<v-checkbox> / <v-switch>

```
<v-checkbox v-model="lazyDel" label="Lazy Delete">
</v-checkbox>

<v-switch v-model="useWifi" label="Wifi">
</v-switch>
```

```
<script setup lang="ts">
  const lazyDel = ref(false);
  const useWifi = ref(true);
}
```



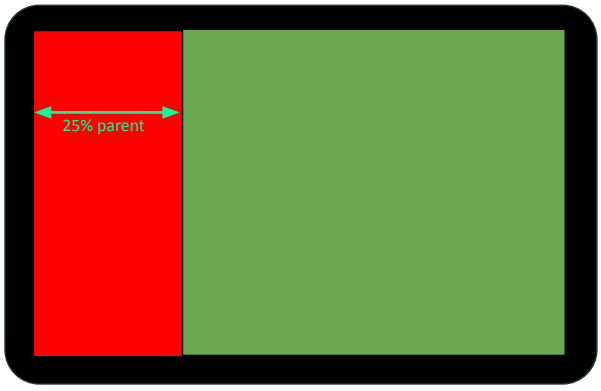
Grid System

Vuetify Grid System

- Inspired by Bootstrap grid & CSS Flexbox
- Relevant elements: `<v-container>`, `<v-row>`, `<v-col>`, `<v-spacer>`
  - Warning: row/col does not imply “horizontal”/”vertical” grouping of elements
- Vuetify 12-column layout?
  - Why 12? It is a relatively small number with **many integer factors**: 2, 3, 4, 6
  - It is easy to divide the screen into
    - Full width (100%)
    - 2 halves (each column is 50% of total screen width => 6/12 each)
    - 3 thirds (33% -> 4/12 each)
    - 4 quarters (25% => 3/12 each)
- Online playground

Vuetify Grid Layout (12-column system)

```
<v-container>
  <v-row>
    <v-col cols=”3” class=”bg-red”>
    </v-col>
    <v-col class=”bg-green”>
    </v-col>
  </v-row>
</v-container>
```



Responsive Layout?

- Problem: Users may use your web app on a variety of screen sizes
  - Smartphones, table, desktop, HDTV, Ultra HDTV, …
  - Can't use one-layout-fits-all approach!
- Traditional Solution: CSS @media query
  - ```
    @media (max-width:  600px) { visual styles #1 }
    @media (max-width: 1024px) { visual styles #2 }
    @media (max-width: 1880px) { visual styles #3 }
    ```
- Vuetify Solution
  - Use **relative dimensions** based on 12-column grid
  - Use **fluid** container (as opposed to fixed size)
  - Size code & media **breakpoints**

Size Code and Media breakpoints

Symbolic names for sizes based on 12-column grid layout

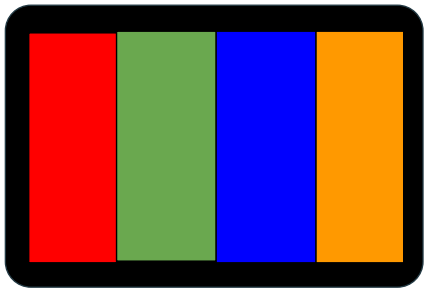
Code	Description	Media	Size Range
xs	eXtra Small	Smartphones	up to 600 px
sm	Small	Tablets	up to 960 px
md	Medium	Small Desktops	up to 1264 px
lg	Large	Large Desktops	up to 1904 px
xl	eXtra Large	Huge Desktops	> 1904 px

Examples

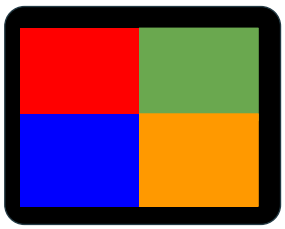
- xs=3 ⇒ 25% (3/12) width on any media
- sm=6 ⇒ 50% (6/12) width on tablets or larger media
- md=12 ⇒ full width on desktops
- lg=9 ⇒ 75% (9/12) width on large desktops or above
- xl=2 ⇒ 16% (2/12) width on huge desktops
    ```

Vuetify Grid Layout (12-column system)

```
<v-container>
  <v-row>
    <v-col md="3" class="red"    sm="6"></v-col>
    <v-col md="3" class="green"  sm="6"></v-col>
    <v-col md="3" class="blue"   sm="6"></v-col>
    <v-col md="3" class="orange" sm="6"></v-col>
  </v-row>
</v-container>
```



On small screens



Online playground

Vuetify Grid Layout (12-column system)

```
<v-container>
  <v-row>
    <v-col v-for="url in imageURLs"
           lg="2" md="3" sm="4" xs="6">
      <img v-bind:src="url" />
    </v-col>
  </v-row>
</v-container>
```

Screen Size	Effective Width	# of images per line
Large	2 (17% of total)	6
Medium	3 (25% of total)	4
Small	4 (33% of total)	3
Extra Small	6 (50% of total)	2

Online playground

Styling via Class Names

Vuetify provides predefined class names for styling the following properties:

- Material Color classes (background and foreground)
- Space classes (margin & padding)
- Media size breakpoints

Styling Margin/Padding via Class Names

- Vuetify provides predefined classnames (`zz-n`) as shortcuts for defining margins and paddings
- Regex for these class names: `[mp] tblraxy 0-5`

margin / padding

top bottom left right all
x (left&right) y (top&bottom)

Styling Margin/Padding via Class Names

Code	Pixels*	General
0	0	0
1	4px	0.25*S
2	8px	0.5*S
3	16px	S
4	24px	1.5*S
5	48px	3*S
Desktop: S = 16px		

[mp] tblraxy 0-5

Vuetify CSS Class	Equivalent CSS Declaration
ma-2	margin: 8px;
ml-3	margin-left: 16px;
pt-0	padding-top: 0;
pb-5	padding-bottom: 48px;
px-1	padding-left: 4px; padding-right: 4px;

Spacing Class Names

```
<!-- traditional HTML file --->
<span id="msg">Sample Text</span>
```

```
#msg {
  margin-left: 24px;
}
```

The traditional way of styling

OR the vuetify way

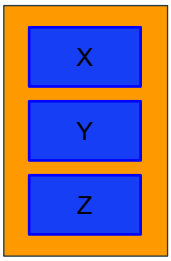
```
<span class="ml-4">Sample Text with left margin</span>

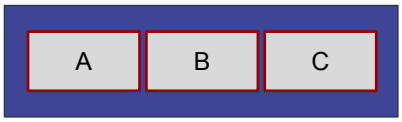
<p class="py-2">text with top and bottom padding…</p>
```

Row- vs. Column-Oriented Box

```
<v-container>
  <v-row>
    <v-col>A</v-col>
    <v-col>B</v-col>
    <v-col>C</v-col>
  </v-row>
</v-container>
```

```
<v-container>
  <v-row class="flex-column">
    <v-col>X</v-col>
    <v-col>Y</v-col>
    <v-col>Z</v-col>
  </v-row>
</v-container>
```





v-row: justify-* and align-*

CSS3 FlexBox	CSS3 Flexbox Properties	Vuetify Attribute

Main-axis ***justification***	`justify-content: flex-start;`
	`justify-content: flex-end;`
	`justify-content: flex-center;`
	`justify-content: flex-space-around;`
	`justify-content: flex-space-between;`
		`justify="start"`
		`justify="end"`
		`justify="center"`
		same as above

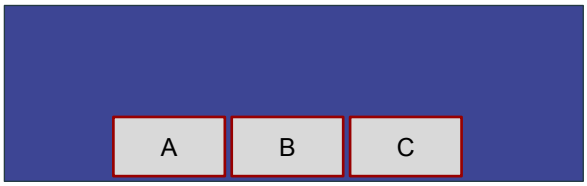
Cross-axis ***alignment***	`align-items: flex-start;`
	`align-items: flex-end;`
	`align-items: flex-center;`
	`align-items: flex-baseline;`
		`align="start"`
		`align="end"`
		same as above

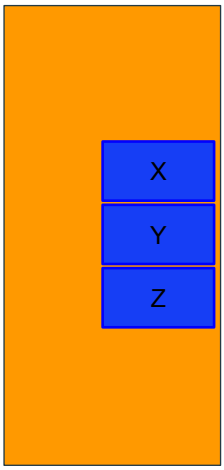
Vuetify accepts both `align` *or* `align-content` *as valid attributes*

Justification vs. Alignment

```
<v-container>
  <v-row justify="center"
         align="end">
    <v-col>A</v-col>
    <v-col>B</v-col>
    <v-col>C</v-col>
  </v-row>
</v-container>
```

```
<v-container>
  <v-row class="flex-column"
    justify="center" align="end">
    <v-col>X</v-col>
    <v-col>Y</v-col>
    <v-col>Z</v-col>
  </v-row>
</v-container>
```





Props vs. Slots

```
<v-btn append-icon="mdi-lock"
       color="primary">
  Click me
</v-btn>
```

*Use props (append-icon, color, ….) to supply **simple data** to a component*

Slots — v-btn API

Name	Description
default	The default Vue slot.
loader	Custom loader.

```
<v-btn append-icon="mdi-lock"
       color="primary">
  Click me
  <template v-slot:loader>
    <-- HTML contents here -->
  </template>
</v-btn>
```

- *Use slots to supply **a large HTML content** a component*
- *Use the slot name given in the API*

Data Presentation: Collections (Advanced)

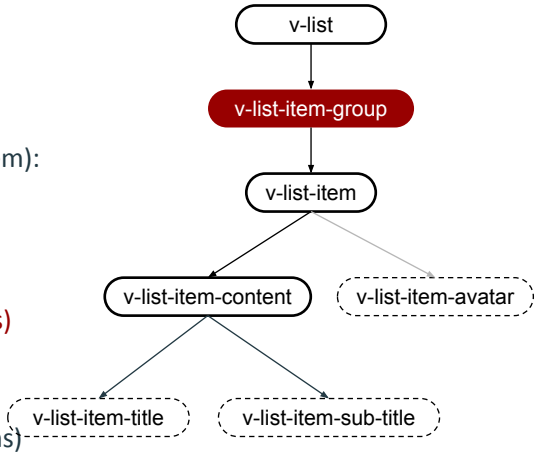
Components for Data Presentation

- v-list (and friends)
- v-data-table: more elaborate control than v-list (header, footer, paging)
- v-carousel

Important attribute: `v-slot`

Using <v-list>

- `v-list`: the entire list container
- Minimal elements
  - `v-list-item`: “template” for each item
  - `v-list-item-content` (a child of v-list-item): the actual content for each item
  - `v-list-item-title` (1-item list)
- Additional elements
  - **`v-list-item-group`** (for selectable items)
  - `v-list-item-sub-title` (2-item list)
  - `v-list-item-avatar` (3-item list)
  - `v-list-item-action` (for actionable items)



Data Table

Minimalist <v-data-table>

```
<template>
  <v-data-table :headers="hArray" :items="cities">
  </v-data-table>
</template>
<script lang="ts">
export default class Sample extends Vue {
  hArray = [
    {text: "US City", value:"name"},
    {text: "Population (in thousands),
       value:"population",
       sortable:true, align: "end"}];
  cities = [
    {name: "New York City", population:  8_704},
    {name: "Chicago",       population:  2_712}
    {name: "Los Angeles",   population: 12_447}
    {name: "Grand Rapids",  population:    198},
    {name: "Boston",        population:  4_309},
    {name: "San Francisco", population:  3_314}]
</script>
```

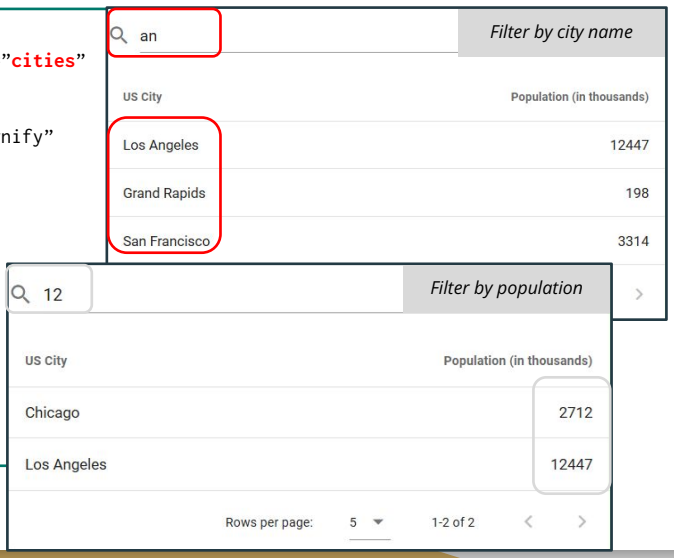
Minimum attributes: `:headers` *and* `:items`

<v-data-table> with filter

```
<template>
  <v-data-table :headers="hArray" :items="cities"
    :search="searchKey">
    <template v-slot:top>
      <v-text-field prepend-icon="mdi-magnify"
         v-model="searchKey"
    <template>


  searchKey
  hArray

   cities
```



<v-data-table> with custom cell

```
<template>
  <v-data-table :headers="hArray" :items="cities">
    <template v-slot:[`item.population`]="whatever">
      <v-icon v-if="whatever.value > 5000"
              color="red">mdi-city</v-icon>
      {{whatever.value}}
    </template>
  </v-data-table>
</template>
<script lang="ts">
  hArray = [
    {text: "US CIty", value: "name"},
    {text: "Population", value: "population"}
</script>
```