CIS 371 Web Application Programming

Vuetify II

Lecturer: Dr. Yong Zhuang

Based on the original version by Professor Hans Dulimarta.

Grid System

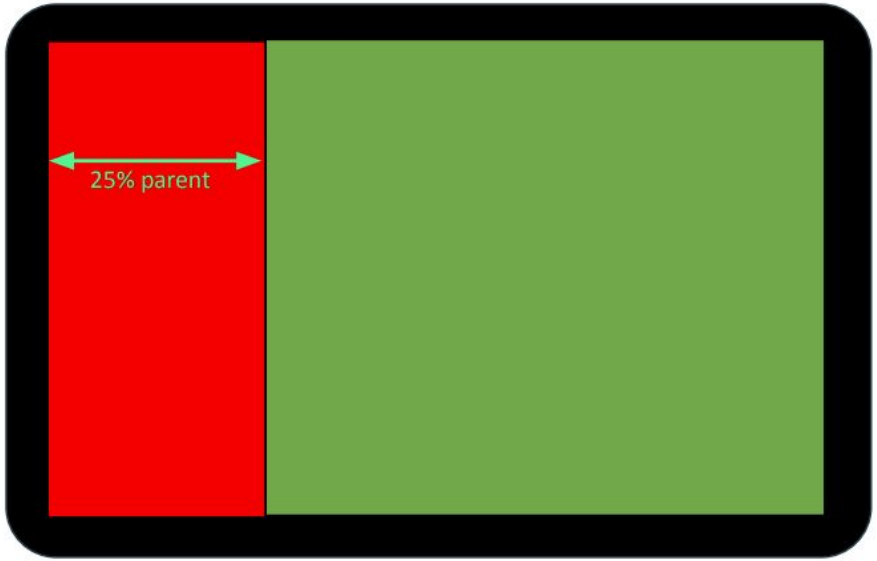
Vuetify Grid System

- Inspired by Bootstrap grid & CSS Flexbox
- Relevant elements: v-row, v-col,
  - 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>
```



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!

Responsive Layout

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	Small to large phone	up to 600 px
sm	Small	Small to medium tablet	up to 960 px
md	Medium	Large tablet to laptop	up to 1280 px
lg	Large	Laptop to desktop	up to 1920 px
xl	Extra Large	1080p to 1440p desktop	up to 2560 px
xxl	Extra extra Large	4k and ultra-wide	> 2560 px

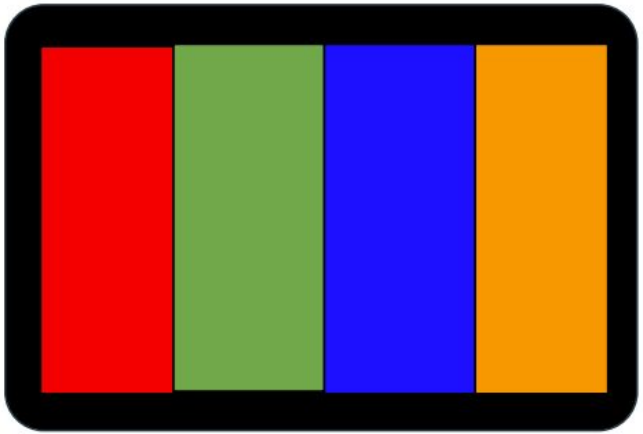
Examples:

- 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

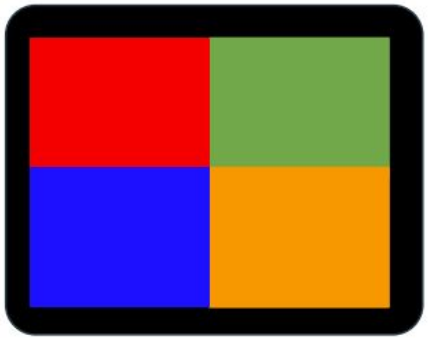
Medium vs. Small

```
<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 medium screens



On small screens



Size Code and Media breakpoints

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

Screen Size	Effective Width	# of images per line?
Large	2	
Medium	3	
Small	4	
Extra Small	6	

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:

[mpg] [tblrseaxyc] - [0-16 | n1-16 | auto]

margin / padding/gap

top / bottom / left / right /start / end/ all
x (left&right) / y (top&bottom) / column-gap

Vuetify Docs - Spacing

Styling Margin/Padding via Class Names
[mpg] [tblrseaxyc] - [0-16 | n1-16 | auto]

Code	Pixels
0	0
1	4px
2	8px
3	12px
4	16px
n1	-4px
n3	-12px

negative number only work for margin.

Vuetify CSS Class	Equivalent CSS Declaration
ma-2	
ml-3	
pt-0	
pb-5	
px-1	
me-n2	

Spacing Class Names

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

<style>
  #msg {
    margin-left: 24px;
  }
</style>
```

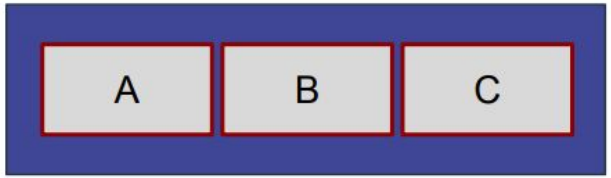
Traditional way

```
<span class="ml-4">Sample Text with left margin</span>
<p class="py-2">text with top and bottom padding…</p>
```

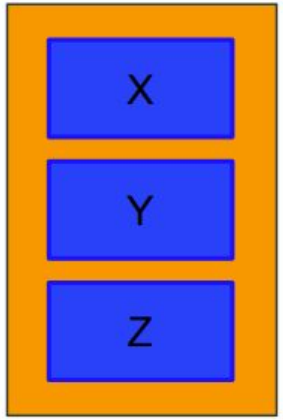
The vuetify way

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

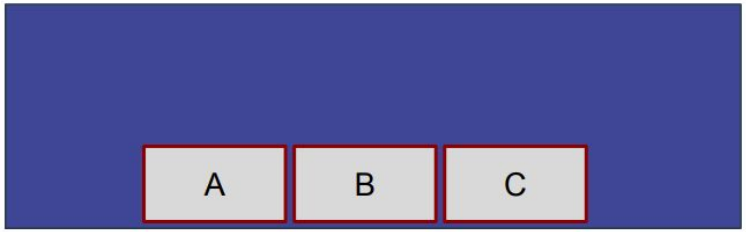
Main-axis ***justification***	`justify-content: flex-start;`
	`justify-content: flex-end;`
	`justify-content: flex-center;`
	`justify-content: flex-space-around;`
	`justify-content: flex-space-between;`
		`justify="start"`
		`justify="end"`
		`justify="center"`
		same as above

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

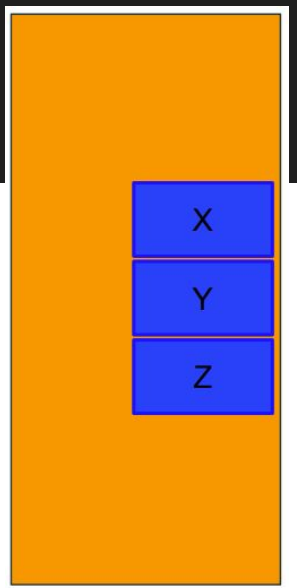
Vuetify accepts both align or align-content as valid attributes

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>
```



Containment: Bottom sheet

```
<template>
  <div class="text-center">
    <v-btn size="x-large" text="Click Me" @click="sheet = !sheet"></v-btn>

    <v-bottom-sheet v-model="sheet" inset>
      <v-card class="text-center" height="200">
        <v-card-text>
          <v-btn variant="text" @click="sheet = !sheet"> close </v-btn>
          <div>This is a bottom sheet that is using the inset prop</div>
        </v-card-text>
      </v-card>
    </v-bottom-sheet>
  </div>
</template>

<script lang="ts" setup>
import { ref } from "vue";
const sheet = ref(false);
</script>
```

With the inset prop, reduce the maximum width of the content area on desktop to 70%.

Demo

```
<template>
  <v-app>
    <v-container>
      <v-bottom-sheet>
        <template v-slot:activator="{ props }">
          <v-btn v-bind="props" text="Click Me"></v-btn>
        </template>
        <v-card
          title="Bottom Sheet"
          text="Lorem ipsum dolor sit amet consectetur,..."
        ></v-card>
      </v-bottom-sheet>
    </v-container>
  </v-app>
</template>
```

Demo

Containment: Buttons

```
<v-btn icon="mdi-account" size="x-small"></v-btn>
```

The **size** property is used to control the size of the button and scales with density.

```
<v-btn density="default" icon="mdi-open-in-new"></v-btn>
```

The **density** prop is used to control the vertical space that the button takes up.

```
<v-btn append-icon="mdi-account" size="x-small">Tom</v-btn>
```

```
<v-btn prepend-icon="mdi-account" size="x-small">Tom</v-btn>
```

Props like prepend-icon and append-icon offer a straightforward approach to incorporate positioned icons,

```
<v-btn append-icon="mdi-account-circle" prepend-icon="mdi-check-circle">
  <template v-slot:prepend>
    <v-icon color="success"></v-icon>
  </template>
  Button
  <template v-slot:append>
    <v-icon color="warning"></v-icon>
  </template>
</v-btn>
```

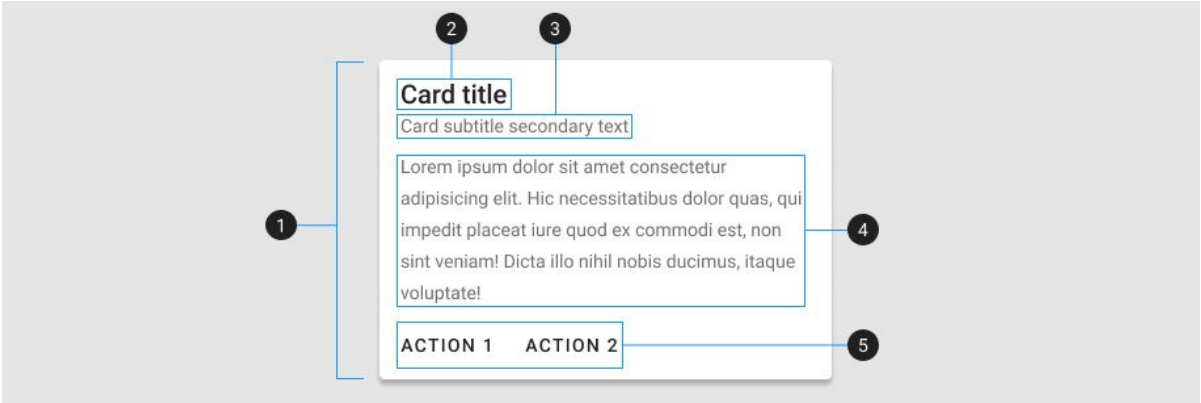
For the prepend-icon and append-icon, the corresponding slots are prepend and append

Demo

Variants

Value	Example
elevated	BUTTON
flat	BUTTON
tonal	BUTTON
outlined	BUTTON
text	BUTTON
plain	BUTTON

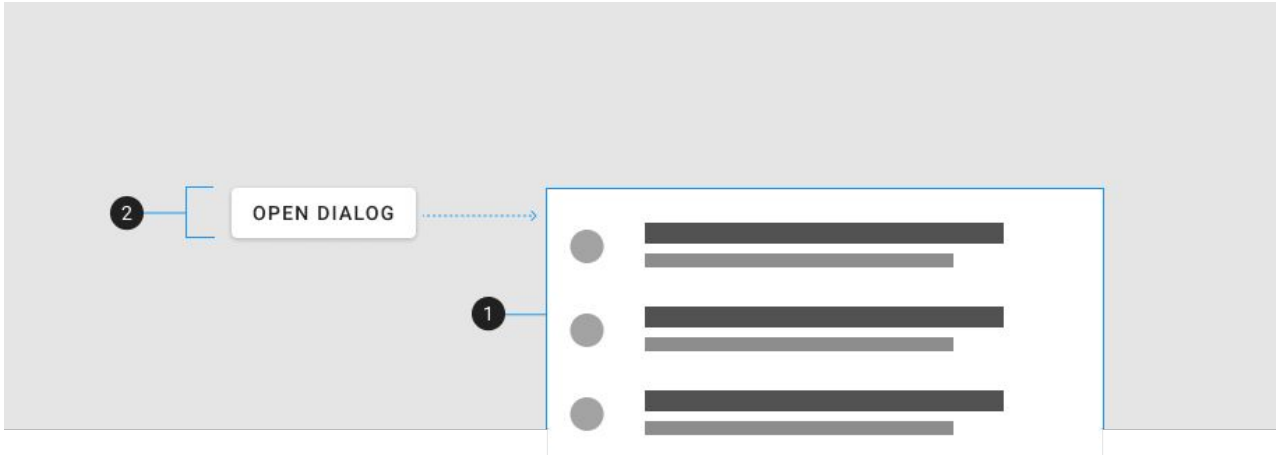
Containment: Cards



Element / Area	Description
1. Container	The Card container holds all `v-card` components. Composed of 3 major parts: `v-card-item`, `v-card-text`, and `v-card-actions`
2. Title (optional)	A heading with increased **font-size**
3. Subtitle (optional)	A subheading with a lower emphasis text color
4. Text (optional)	A content area with a lower emphasis text color
5. Actions (optional)	A content area that typically contains one or more v-btn components

Demo

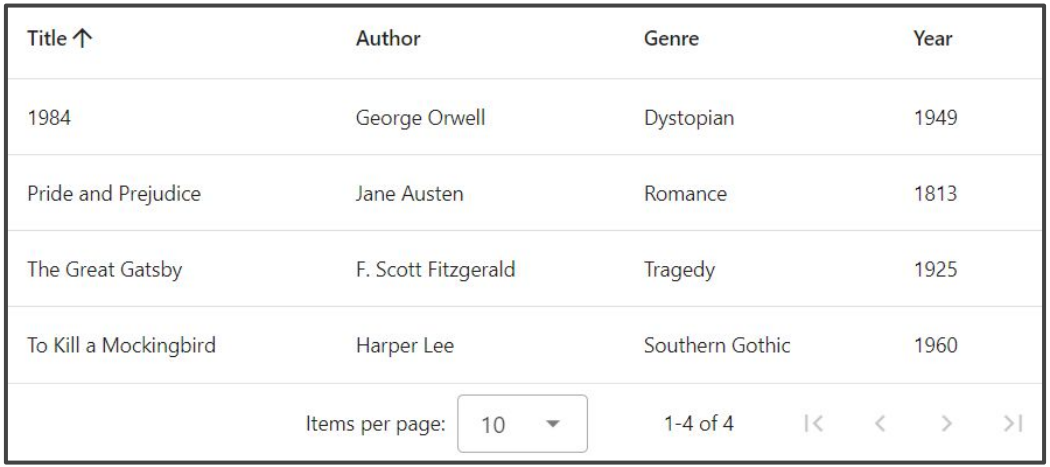
Containment: Dialogs



Element / Area	Description
1. Container	The dialog's content that animates from the activator
2. Activator	The element that activates the dialog

Demo

<v-data-table>



Title ↑	Author	Genre	Year
1984	George Orwell	Dystopian	1949
Pride and Prejudice	Jane Austen	Romance	1813
The Great Gatsby	F. Scott Fitzgerald	Tragedy	1925
To Kill a Mockingbird	Harper Lee	Southern Gothic	1960

Items per page: 10 1-4 of 4

```
<template>
  <v-app>
    <v-container>
      <v-data-table :items="books"></v-data-table>
    </v-container>
  </v-app>
</template>
```

Vuetify Docs - Data Tables

Demo

Custom Headers for Vuetify <v-data-table>

```
<v-data-table
  :items="books"
  :headers="[
    { title: 'title', align: 'start', sortable: false, key: 'title' },
    { title: 'author', align: 'end', sortable: true, key: 'author' },
    { title: 'year', align: 'end', sortable: true, key: 'year' },
  ]"
></v-data-table>
```

Demo

Adding Selection to Vuetify <v-data-table>

```
<v-data-table
  :items="books"
  show-select
  :headers="[
    { title: 'title', align: 'start', sortable: false, key: 'title' },
    { title: 'author', align: 'end', sortable: true, key: 'author' },
    { title: 'year', align: 'end', sortable: true, key: 'year' },
  ]"
  v-model="selected"
  item-value="title"
></v-data-table>
```

```
import { ref } from "vue";
const selected = ref([]);
```

Demo

Adding Search to Vuetify <v-data-table>

```
<v-text-field
  v-model="search"
  append-icon="mdi-magnify"
  label="Search"
  single-line
  hide-details
></v-text-field>
<v-data-table
  :items="books"
  show-select
  :headers="[
    { title: 'title', align: 'start', sortable: false, key: 'title' },
    { title: 'author', align: 'end', sortable: true, key: 'author' },
    { title: 'year', align: 'end', sortable: true, key: 'year' },
  ]"
  v-model="selected"
  item-value="title"
  :search="search"
></v-data-table>
```

```
import { ref } from "vue";
const selected = ref([]);
const search = ref("");
```

Demo