CIS 371 Web Application Programming

VueJS 3.x (Vue3) IV

Declarative Component-Based UI Framework

Lecturer: Dr. Yong Zhuang

Based on the original version by Professor Hans Dulimarta.

VueJS Reactive Reference + TypeScript Typing

The TS compiler infers the type from the surrounding context

```
import { ref } from "vue";
const name = ref(""); // name.value is implicitly a string
const year = ref(2001); // year.value is implicitly a number
const names = ref([]); // names.value is an array of UNKNOWN type
```

```
const name: string = ref("");
```

Is this correct?

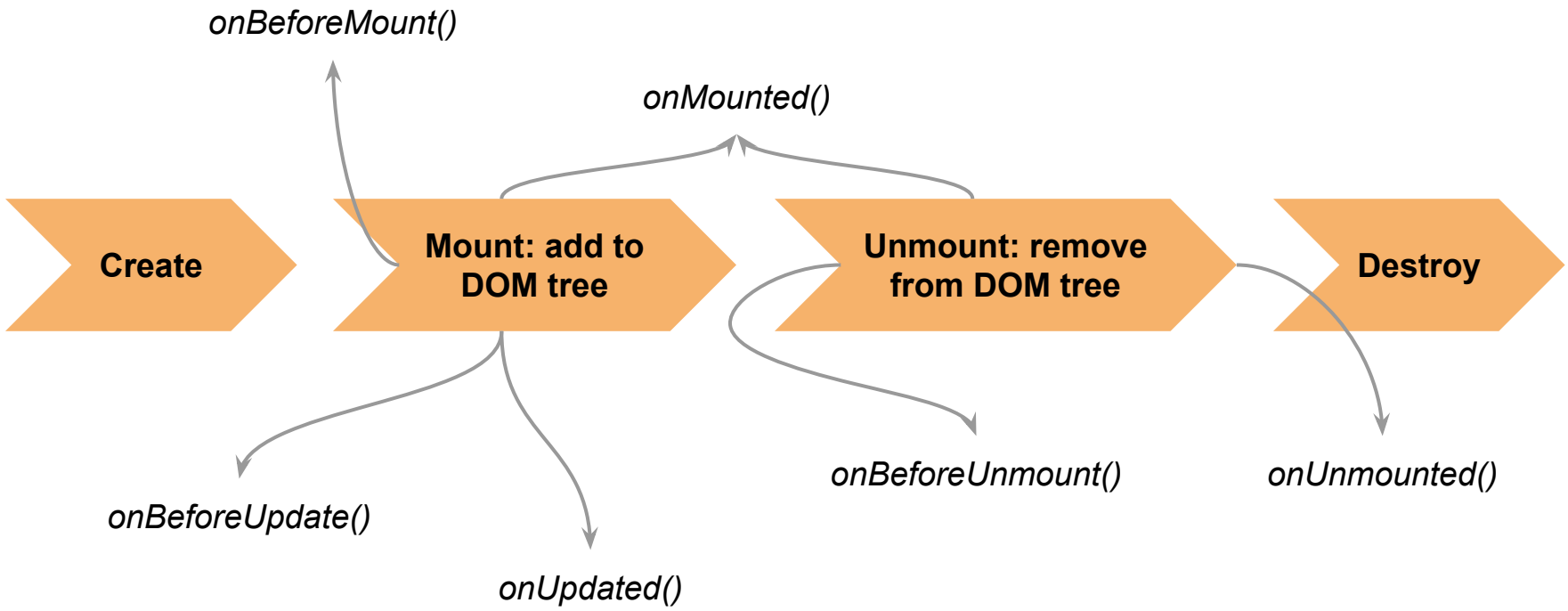
VueJS Reactive Reference + TypeScript Typing

The TS compiler infers the type from the surrounding context

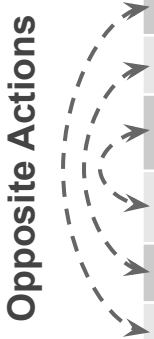
```
import { ref } from "vue";
const name = ref(""); // name.value is implicitly a string
const year = ref(2001); // year.value is implicitly a number
const names = ref([]); // names.value is an array of UNKNOWN type
```

```
import { ref, Ref } from "vue";
const name: Ref<string> = ref("");
const name1 = ref<string>("");
const year: Ref<number> = ref(2001);
const year1 = ref<number>(2001);
const names: Ref<string[]> = ref([]);
const names1 = ref<string[]>([]);
```

Vue3 Lifecycle Functions



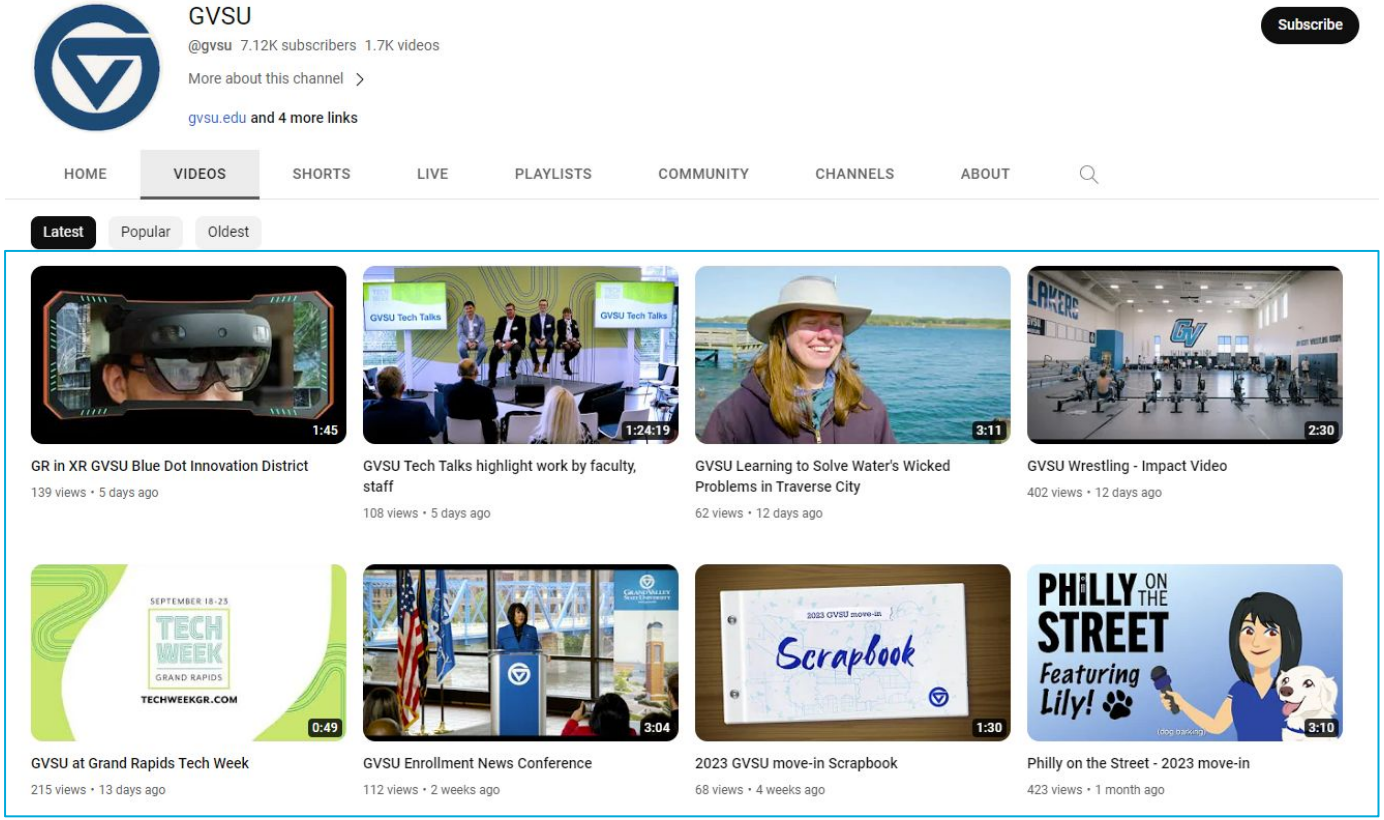
Practical Use of Lifecycle Hooks



Function	Description	Sample Usage
onBeforeMount()	Component will appear	Restore UI from persistent storage (user prefs)
onMounted()	Component appeared	Start timer to monitor user engagement
onBeforeUpdate()	Properties will be updated	Any necessary logic needed • to save any data related to the old props • to restore data related to the new props
onUpdated()	Properties updated	
onBeforeUnmount()	Component will disappear	Stop timer
onUnmounted()	Component disappeared	Save UI details to user preferences

Demo

Using Multiple Vue Components


GVSU
@gvsu 7.12K subscribers 1.7K videos
More about this channel
gvsu.edu and 4 more links
Subscribe
HOME
VIDEOS
SHORTS
LIVE
PLAYLISTS
COMMUNITY
CHANNELS
ABOUT
Latest
Popular
Oldest
1:45
GR in XR GVSU Blue Dot Innovation District
139 views • 5 days ago
GVSU Tech Talks
GVSU Tech Talks
1:24:19
GVSU Tech Talks highlight work by faculty, staff
108 views • 5 days ago
3:11
GVSU Learning to Solve Water's Wicked Problems in Traverse City
62 views • 12 days ago
LAKERS
GV
2:30
GVSU Wrestling - Impact Video
402 views • 12 days ago
SEPTEMBER 18-23
TECH WEEK
GRAND RAPIDS
TECHWEEKGR.COM
0:49
GVSU at Grand Rapids Tech Week
215 views • 13 days ago
3:04
GVSU Enrollment News Conference
112 views • 2 weeks ago
2023 GVSU move-in
Scrapbook
1:30
2023 GVSU move-in Scrapbook
68 views • 4 weeks ago
PHILLY ON THE STREET
Featuring Lily!
(dog barking)
3:10
Philly on the Street - 2023 move-in
423 views • 1 month ago

video cover

coverImage

videoDuration

Philly on the Street - 2023 move-in

videoTitle

423 views • 1 month ago

releaseDate

numberOfViews

GR in XR GVSU Blue Dot Innovation District

139 views • 5 days ago

GVSU Tech Talks highlight work by faculty, staff

108 views • 5 days ago

GVSU Learning to Solve Water's Wicked Problems in Traverse City

62 views • 12 days ago

GVSU Wrestling - Impact Video

402 views • 12 days ago

GVSU at Grand Rapids Tech Week

215 views • 13 days ago

GVSU Enrollment News Conference

112 views • 2 weeks ago

2023 GVSU move-in Scrapbook

68 views • 4 weeks ago

Philly on the Street - 2023 move-in

423 views • 1 month ago

src/YouTubeApp.vue

```vue
<template>
  <div>
    <YouTubeCover v-for="z in availableVideos" :key="z.id"
    :coverImage="z.imgURL"
    :title="z.videoTitle"
    :duration="z.videoDuration"
    :views="z.numberOfViews"
    :release="z.releaseDate" />
  </div>
</template>

<script setup lang="ts">
import { ref } from 'vue';
import YouTubeCover from './YouTubeCover.vue';

const availableVideos = ref([
  {
    id: 1,
    imgURL: 'http://img1',
    videoTitle: 'First Video',
    videoDuration: '12:34',
    numberOfViews: 123456,
    releaseDate: '2021-01-01',
  },
  {
    id: 2,
    imgURL: 'http://img2',
    videoTitle: 'Second Video',
    videoDuration: '5:67',
    numberOfViews: 78910,
    releaseDate: '2021-02-02',
  },
  // more video objects
]);
</script>
```

src/YouTubeApp.vue

```vue
<template>
  <div>
    <YouTubeCover v-for="z in availableVideos" :key="z.id"
    :coverImage="z.imgURL"
    :title="z.videoTitle"
    :duration="z.videoDuration"
    :views="z.numberOfViews"
    :release="z.releaseDate" />
  </div>
</template>

<script setup lang="ts">
import { ref } from 'vue';
import YouTubeCover from './YouTubeCover.vue';

const availableVideos = ref([
  {
    id: 1,
    imgURL: 'http://img1',
    videoTitle: 'First Video',
    videoDuration: '12:34',
    numberOfViews: 123456,
    releaseDate: '2021-01-01',
  },
  {
    id: 2,
    imgURL: 'http://img2',
    videoTitle: 'Second Video',
    videoDuration: '5:67',
    numberOfViews: 78910,
    releaseDate: '2021-02-02',
  },
  // more video objects
]);
</script>
```

Parent Component

src/components/YTCover.vue

```vue
<template>
    <div>
        <!-- UI design goes here -->
        <img :src="coverImage" alt="Video cover">
        <h1>{{ title }}</h1>
        <p>Duration: {{ duration }} minutes</p>
        <p>Views: {{ views }}</p>
        <p>Released: {{ release }}</p>
    </div>
</template>
<script setup lang="ts">
type VideoBlock = {
    coverImage: string;
    title: string;
    duration: string;
    views: number;
    release: string;
}
defineProps<VideoBlock>()
</script>
```

Child Component(s)

Slots

In some cases, we may want to pass a template fragment to a child component, and let the child component render the fragment within its own template.

```
<div class="container">
    <header>
      <!-- We want header content here -->
    </header>
    <main>
      <!-- We want main content here -->
    </main>
    <footer>
      <!-- We want footer content here -->
    </footer>
</div>
```

```
<template>
    <div class="container">
      <header>
        <slot name="header"></slot>
      </header>
      <main>
        <slot></slot>
      </main>
      <footer>
        <slot name="footer"></slot>
      </footer>
    </div>
</template>
```

Slots (v-slot)

BaseLayout.vue

```vue
<template>
    <div class="container">
      <header>
        <slot name="header"></slot>
      </header>
      <main>
        <slot></slot>
      </main>
      <footer>
        <slot name="footer"></slot>
      </footer>
    </div>
</template>
```

App.vue

```vue
<script setup>
import BaseLayout from './BaseLayout.vue'
</script>

<template>
  <BaseLayout>
    <template v-slot:header>
      <h1>Here might be a page title</h1>
    </template>
    <p>A paragraph for the main content.</p>
    <p>And another one.</p>
    <template #footer>
      <p>Here's some contact info</p>
    </template>
  </BaseLayout>
</template>
```

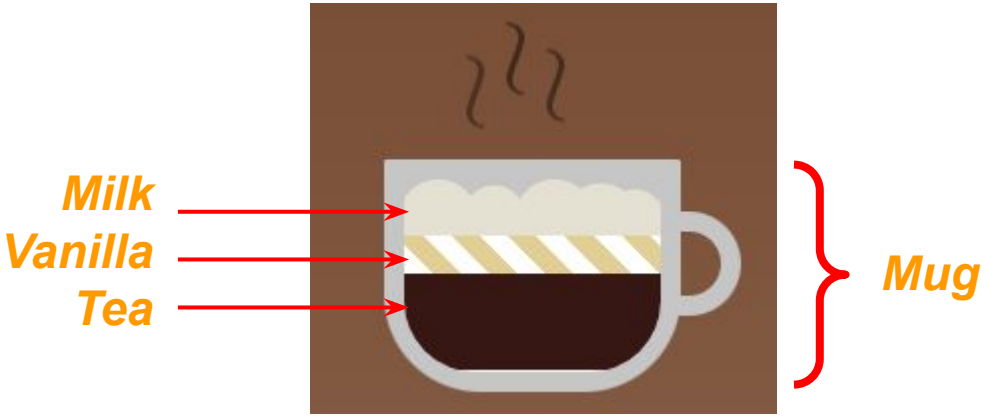
Demo

kebab-case vs. camelCase

kebab-case (in HTML)	camelCase (in TypeScript)
image	`image`
`cover-image`	`coverImage`
`cover-image-url`	`coverImageUrl`

Example: Beverage: London Fog

London Fog




```
<Mug>
    <Milk></Milk>
    <Vanilla></Vanilla>
    <Tea></Tea>
</Mug>
```

London Fog

Hot

Cold

Code

Demo