

CIS 658 Web Architectures

Vue.js III

Declarative Component-Based UI Framework



GRAND VALLEY
STATE UNIVERSITY®

Lecturer: **Dr. Yong Zhuang**

Review

- Two-Way Data Binding (v-model) :
 - .lazy, .number,
 - color, and date picker
 - radio buttons & dropdown list & checkbox list
- Event Handling:
 - v-on:domEvents="function"; @domEvents="function"
 - Event Names: keypress, keydown, keyup, wheel, click, blur, focus mousedown, mouseenter, mousemove, mouseup, ...
- Event Modifiers
 - Key Modifiers: .enter, .tab, .delete, .esc, .space, .up, .down, .left, .right, ...
 - System Modifiers: .alt, .ctrl, .meta, .shift,
 - .prevent, .stop, .capture, .self, ...
 - .exact modifier
- computed
- watch

Exercise from the previous class

Two-Way Data Binding (v-model)

Answer

Topics

- Two-Way Data Binding (v-model) :
 - .lazy, .number,
 - color, and date picker
 - radio buttons & dropdown list & checkbox list
- Event Handling:
 - v-on:domEvents="function"; @domEvents="function"
 - Event Names: keypress, keydown, keyup, wheel, click, blur, focus, mousedown, mouseenter, mousemove, mouseup, ...
- Event Modifiers
 - Key Modifiers: .enter, .tab, .delete, .esc, .space, .up, .down, .left, .right, ...
 - System Modifiers: .alt, .ctrl, .meta, .shift,
 - .prevent, .stop, .capture, .self, ...
 - .exact modifier
- computed
- watch

Defining and Using Components

Defining and Using Components

ThumbsUp.vue

```
<template>
  <span>👍 </span>
</template>
<style>
span {
  font-size: 200%;
}
</style>
```

Sample.vue

```
<script setup lang="ts">
import { ref } from "vue";
import ThumbsUp from "../ThumbsUp.vue";
const msg = ref("Hello World!");
</script>

<template>
  <h1>Component Demo</h1>
  <ThumbsUp></ThumbsUp>
  <h2>{msg}</h2>
</template>
```

[Demo](#)

Defining and Passing "Argument"

```
<script setup lang="ts">
import { ref } from "vue";
import ThumbsUp from "../ThumbsUp.vue";
const numThumbsUp = ref(1);
</script>
```

```
<template>
  <h1>Component Demo</h1>
  <p>
    Pick a number
    <input type="range" v-model.number="numThumbsUp" min="1" max="20" />
  </p>
  <ThumbsUp :repeat="numThumbsUp" ></ThumbsUp>
</template>
```

Sample.vue

```
ThumbsUp.vue
<template>
  Can you show {{ props.repeat }} thumbs?
  <span v-for="k in props.repeat">👍 </span>
</template>
<script setup lang="ts">
type ThumbProp = {
  repeat: number;
};
const props = defineProps<ThumbProp>();
</script>
<style>
span {
  font-size: 200%;
}
</style>
```

[Demo](#)

Simple Timer

```
<script setup lang="ts">
import { ref } from "vue";
const seconds = ref(0);
const minutes = ref(0);
let timerInterval: number | null = null; // This variable will hold the interval ID

function twoDigitSeconds() {
  return seconds.value.toLocaleString("en-US", { minimumIntegerDigits: 2 });
}

function updateTime() {
  seconds.value++;
  if (seconds.value === 60) {
    minutes.value++;
    seconds.value = 0;
  }
}

function runTimer() {
  if (!timerInterval) {
    // Check if the timer isn't already running
    timerInterval = setInterval(updateTime, 1000);
  }
}

function stopTimer() {
  if (timerInterval) {
    clearInterval(timerInterval);
    timerInterval = null;
    minutes.value = 0;
    seconds.value = 0;
  }
}
</script>
```

```
<template>
  <div id="timer">
    <div id="timedisplay">{{ minutes }}:{{ twoDigitSeconds() }}</div>
    <button @click="runTimer">Start</button>
    <button @click="stopTimer">Stop</button>
  </div>
</template>
```

[Demo](#)

Setting Default Value on Properties

```
<script setup lang="ts">
import { defineProps } from "vue";
type TimerProp = {
  updateInterval: number;
};
const props = defineProps<TimerProp>();
// more code here
</script>
```

set default value

```
<script setup lang="ts">
import { defineProps, withDefaults } from "vue";
type TimerProp = {
  updateInterval: number;
};
const props = withDefaults(defineProps<TimerProp>(), {
  updateInterval: 1000,
});
// more code here
</script>
```

Customization of Components Via Props

- Injection into component variable(s):
 - Timer update speed vs. Timer update speed with default value
- Injection into UI <template> & <style>: Stylish Timers
- what else?

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?

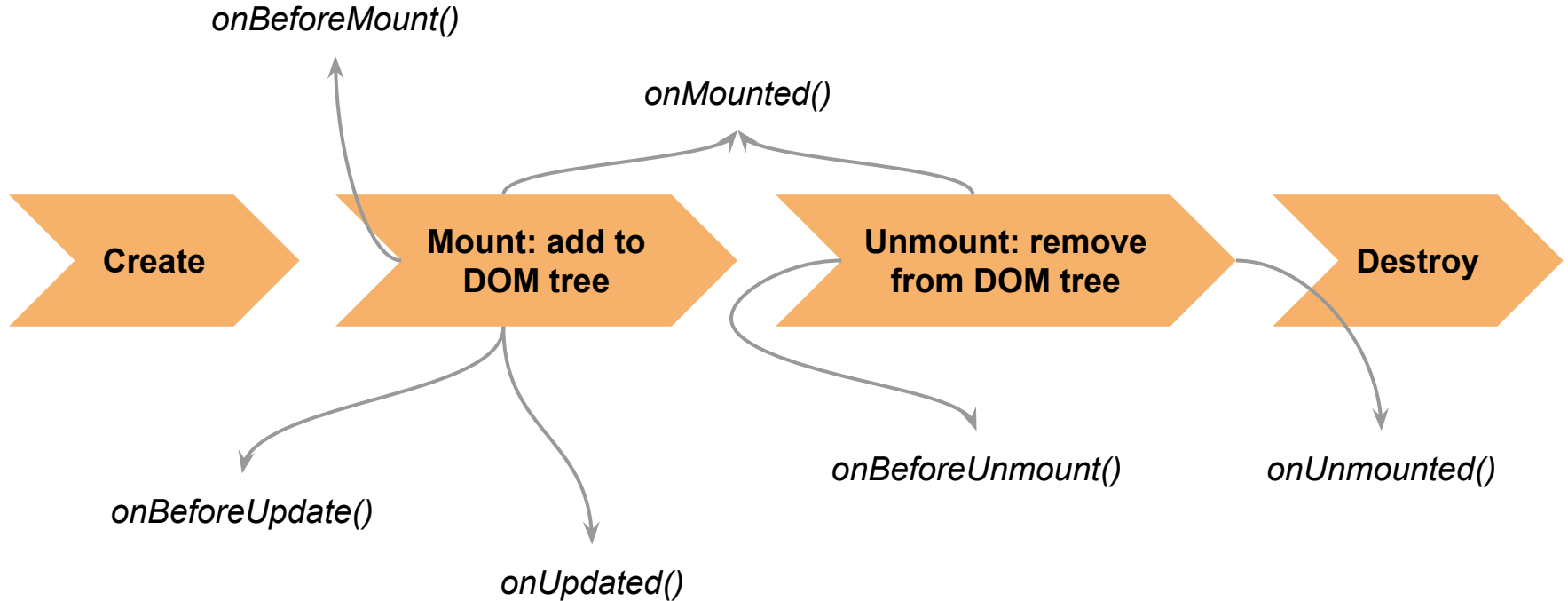
VueJS Reactive Reference + TypeScript Typing

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
Opposite Actions	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 <ul style="list-style-type: none">• 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



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

video cover



coverImage

videoDuration

Philly on the Street - 2023 move-in → *videoTitle*

423 views • 1 month ago → *releaseDate*

↪ *numberOfViews*



1:45

GR in XR GVSU Blue Dot Innovation District

139 views • 5 days ago



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



2:30

GVSU Wrestling - Impact Video

402 views • 12 days ago



0:49

GVSU at Grand Rapids Tech Week

215 views • 13 days ago



3:04

GVSU Enrollment News Conference

112 views • 2 weeks ago



1:30

2023 GVSU move-in Scrapbook

68 views • 4 weeks ago



3:10

Philly on the Street - 2023 move-in

423 views • 1 month ago

<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

```
<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

```
<template>
  <div>
    <!-- UI design goes here -->
    
    <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)

Demo

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

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

App.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

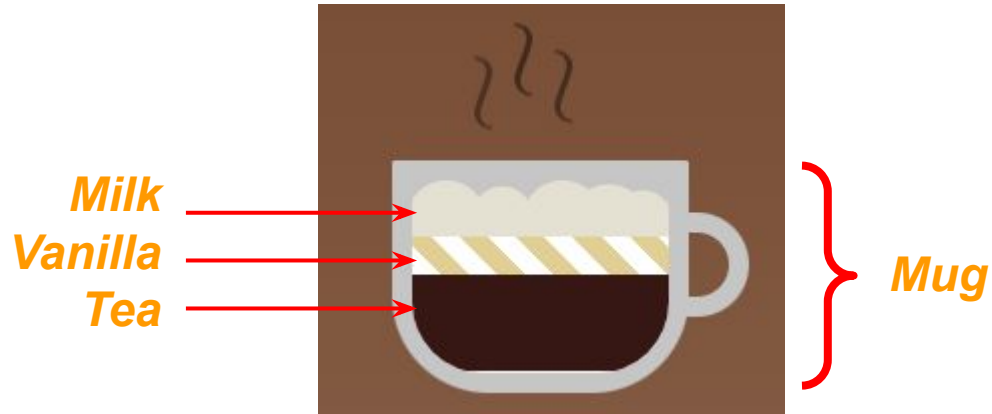
kebab-case vs. camelCase

kebab-case (in HTML)	camelCase (in TypeScript)
<code>image</code>	<code>image</code>
<code>cover-image</code>	<code>coverImage</code>
<code>cover-image-url</code>	<code>coverImageUrl</code>

Example: Beverage: London Fog



London Fog



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

London Fog



Hot



Cold

Code

Demo