CIS 371 Web Application Programming

VueJS 3.x (Vue3) III

Declarative Component-Based UI Framework

GRAND VALLEY STATE UNIVERSITY®

Lecturer: Dr. Yong Zhuang

Based on the original version by Professor Hans Dulimarta.

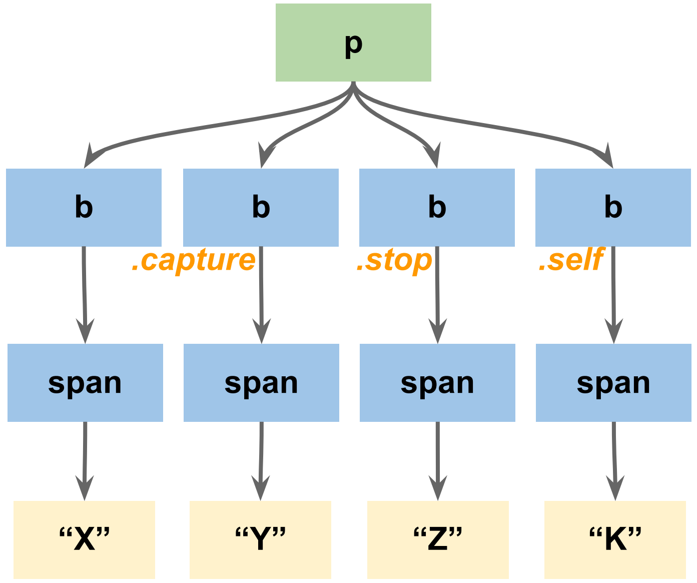
Review

- 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, …
  - Filters: .enter, .tab, .delete, .esc, .space, .up, .down, .left, .right, …
  - Modifiers: .alt, .ctrl, .meta, .shift, .prevent, .stop, .capture, .self, …

Recap: Mouse/Keyboard Event Filters/Modifiers

Event Modifier	Description
.prevent	Prevent browser default action of the event
.stop	Stop propagating event up (bubbling) to ancestor
.capture	Begin here, and propagate event down (capturing) to descendants
.self	Handle events only from self (neither from ancestors nor from descendants)

More Modifiers



- *Events originating in "X" are handled by span > b > p*
- *Events originating in "Y" are handled by b > span > p*
- *Events originating in "Z" are handled by span > b*
- *Events originating in "K" are handled by span > p*

Demo

Template Refs

```
<template>
  <div>
    <h1 ref="bar"></h1>
    <button @click="incrementH1Counter">Plus 1</button>
  </div>
</template>

<script setup lang="ts">
import { ref, onMounted } from 'vue';
const bar = ref(null);
function incrementH1Counter() {
  bar.value.textContent++;

}

onMounted(() => {
  bar.value.textContent = "3";
});
</script>
```

Demo 1

Online doc

Demo 2

Defining and Passing “Argument”

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>
```

```
<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

Demo

Exercise

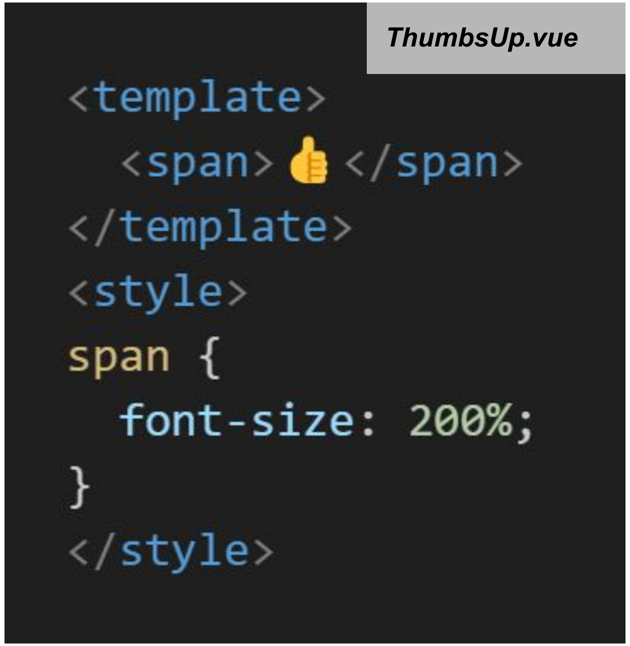
Two-Way Data Binding (v-model)

Using key with v-for in Vue.js 3

- Include key:
  - Whenever you've got something unique
  - You're iterating over more than a simple hard coded array
  - and even when there is nothing unique but it's dynamic data (in which case you need to generate random unique id's)
- Without key:
  - You have nothing unique AND
    - When you're quickly testing something or demonstrating basic functionality with v-for
    - If you’re iterating over a simple hard-coded array that doesn't change and doesn’t need complex updates (not dynamic data from a database, etc)

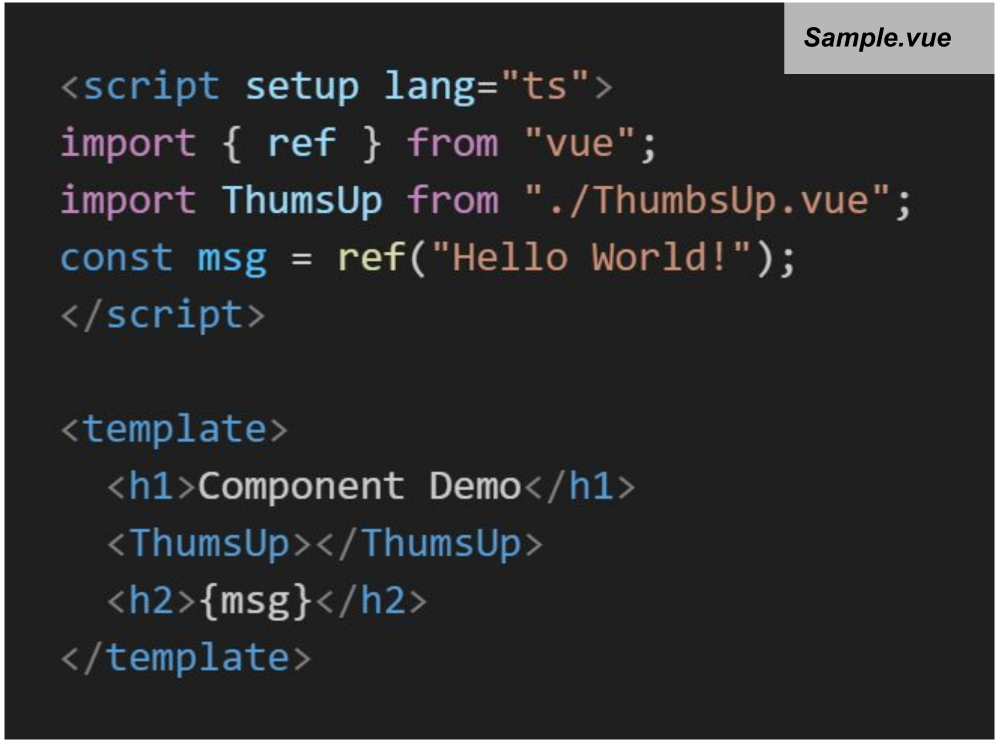
Defining and Using Components

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 ThumsUp from "./ThumbsUp.vue";
const msg = ref("Hello World!");
</script>

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

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

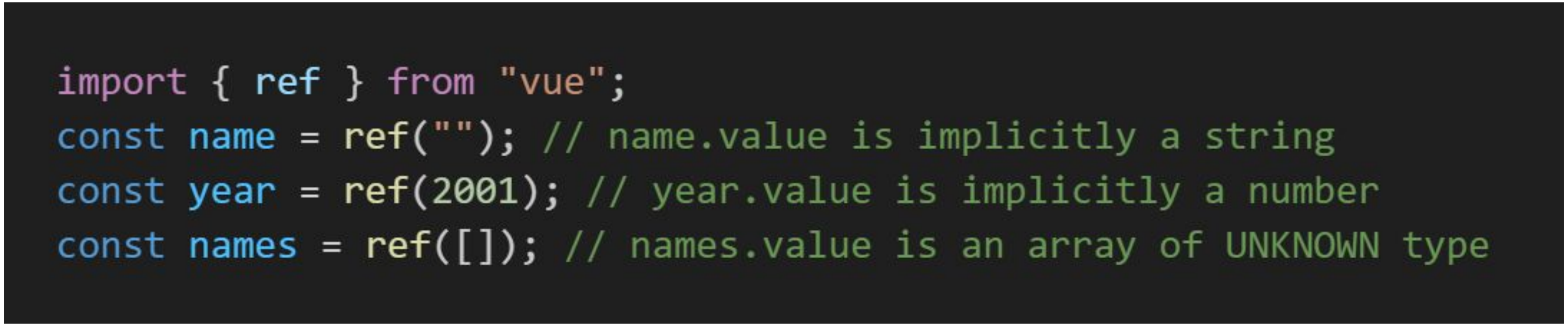
```
<script setup lang="ts">
import { defineProps, withDefaults } from "vue";
type TimerProp = {
  updateInterval: number;
};
const props = withDefaults(defineProps<TimerProp>(), {
  updateInterval: 1000,
});
// more code here
</script>
```

Customization of Components Via Props

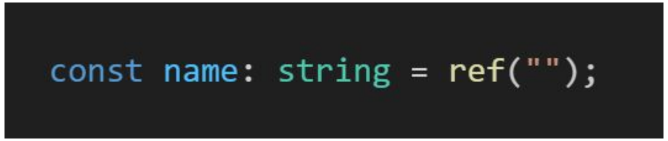
- Injection into component variable(s):
  - Timer update speed vs. Timer update speed with default value
- Injection into UI <template> & <style>: Stylish Timers
- what else?

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
```

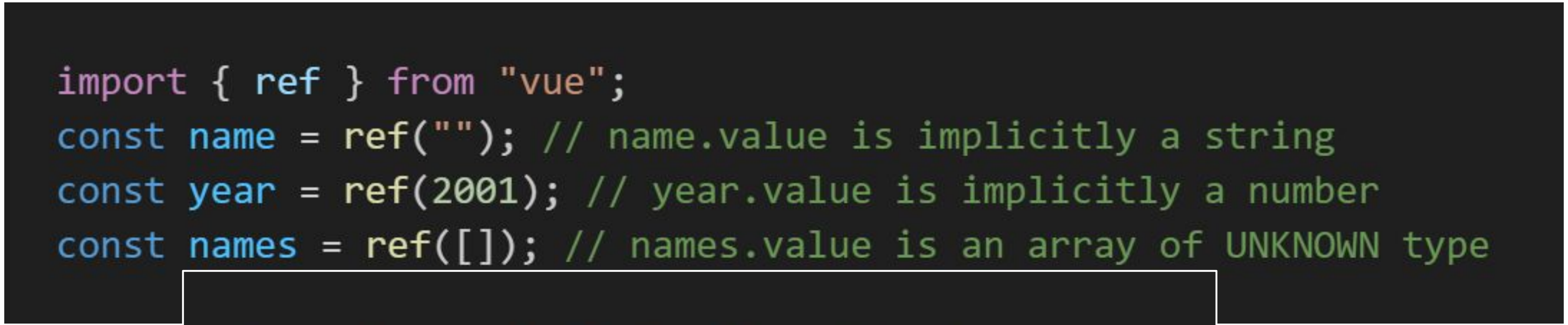


```
const name: string = ref("");
```

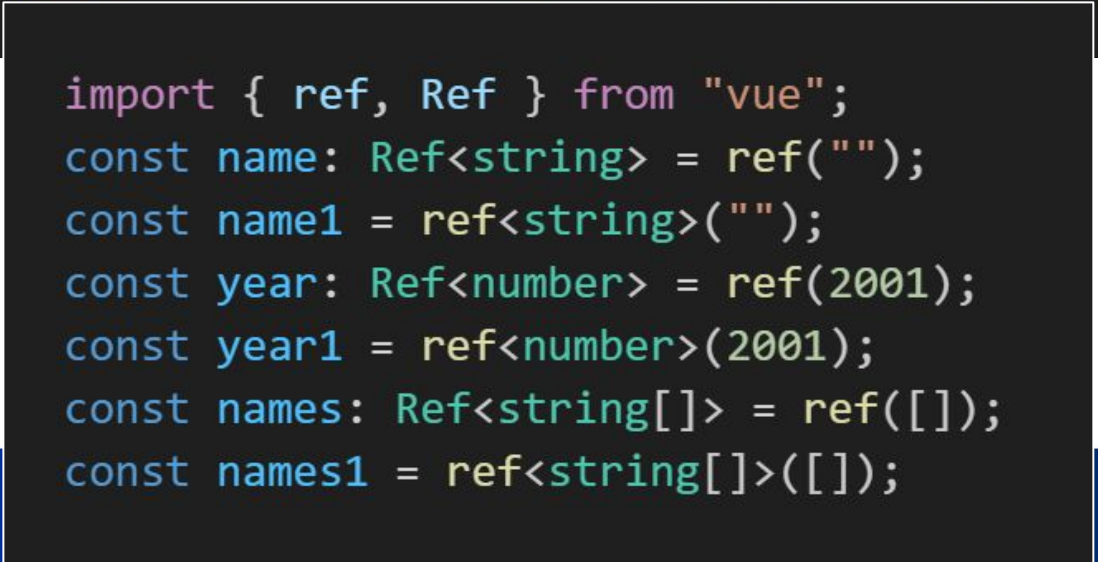
Is this correct?

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[]>([]);
```