Composition API

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

#vue-js

Composition API

Basic setup

What is Composition API?

- Set of APIs that allows us to author Vue components using imported functions instead of declaring options

```
<script setup>
import { ref } from 'vue'

// ref === data()
const count = ref(0)

// function === methods()
function increment() {
  count.value++
}
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

Why we may need it

- **Benefits**
  - Better Logic Reuse
  - More Flexible Code Organization
  - Better Type Inference
  - Smaller Production Bundle and Less Overhead
- **Drawbacks**
  - No "guard rails"
  - Less organized code base
  - Options API does allow you to "think less" when writing component code

Why we may need it

Software University

Options API

Composition API

Composition API usage

```
<script>
import { ref } from 'vue'
export default {
  setup() {
    const count = ref(0);

    function increment() {
      count.value++
    };

    return { count, increment };
  },
}
</script>

<template>
  <button @click="increment">{{ count }}</button>
</template>
```

As an option

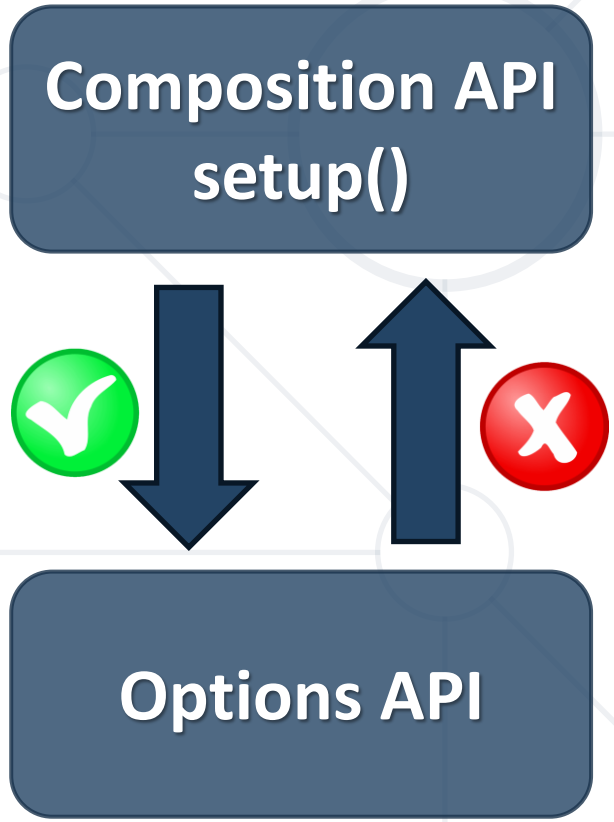
Composition API usage

```
<script setup>
import { ref } from 'vue'

const count = ref(0)

function increment() {
  count.value++
}
</script>

<template>
  <button @click="increment">{{ count
}}</button>
</template>
```

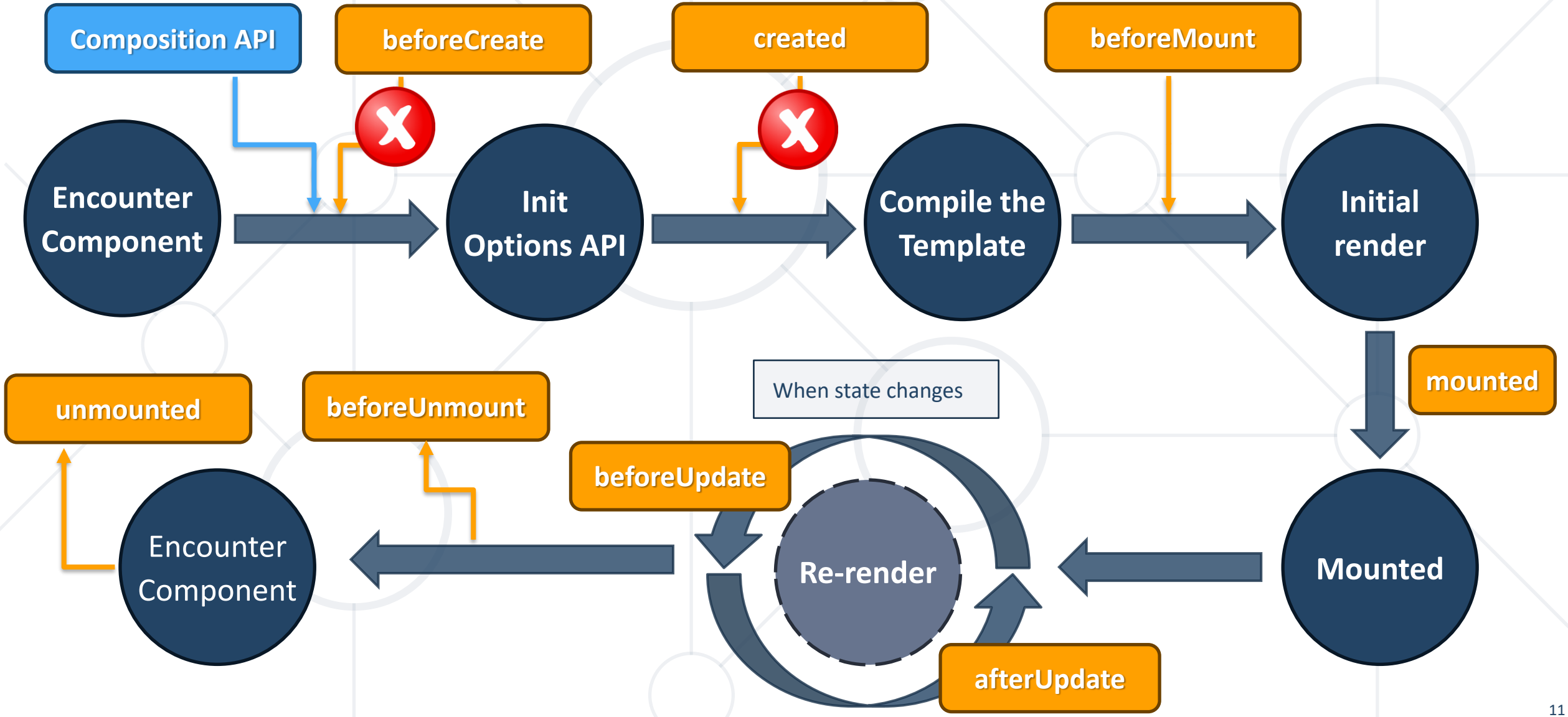
With <script setup>

Composition API with Options API

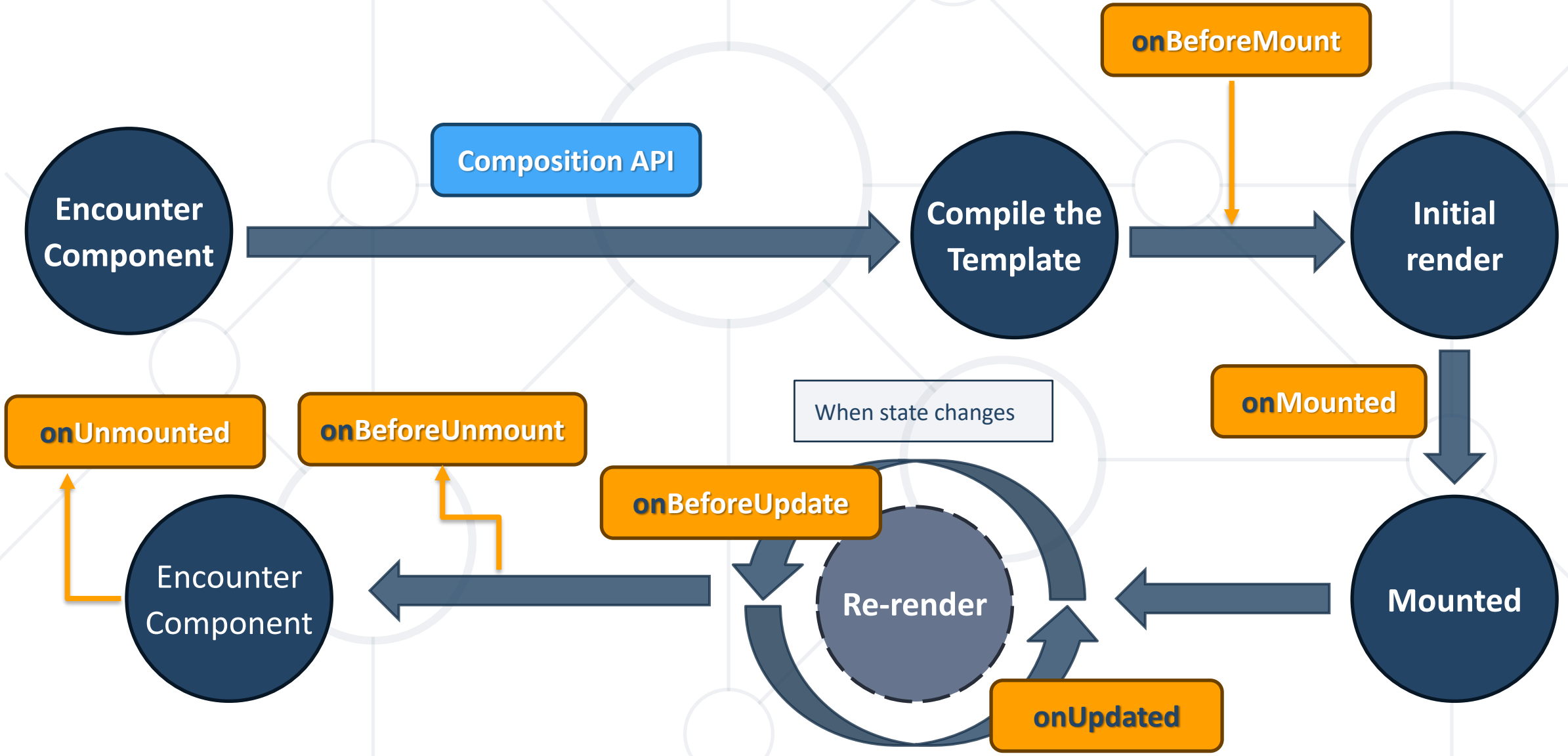
```
<script>
import { ref } from "vue";
export default {
  setup() {
    const count = ref(0);

    function increment() {
      count.value++;
    }
    return { count, increment };
  },
  watch: {
    count(newCountValue) {
      console.log(`Let's double it: ${newCountValue * 2}`);
    },
  },
};
</script>
```



Lifecycles


Composition API
beforeCreate
created
beforeMount
Encounter Component
Init Options API
Compile the Template
Initial render
mounted
unmounted
beforeUnmount
When state changes
beforeUpdate
Encounter Component
Re-render
Mounted
afterUpdate

Lifecycles – Options API



Props and Events

- **defineProps()**
- **defineEmits()**
- Automatically available inside **<script setup>**
- Only usable inside **<script setup>**
- **Do not deconstruct props!**

```
<script setup>
import { ref } from "vue";

const props = defineProps({ initialValue: Number
});
const emit = defineEmits(["change"]);

const count = ref(props.initialValue);

function increment() {
  count.value += 1;
  emit("change", count.value);
}
</script>

<template>
  <button @click="increment">{{ count }}</button>
</template>
```

Reactivity

State, Computed, Watchers

State – ref()

- **ref()**
  - Reactive state
  - Returns the value wrapped within a ref object with a **.value** property
  - **No need** to append **.value** when using the ref in the template

```
<script setup>
import { ref } from "vue";

const count = ref(0);

function increment() {
  count.value += 1;
}
</script>

<template>
  <button @click="increment">{{ count }}</button>
</template>
```

Deep reactivity – ref()

```
const count = ref({
  lets: {
    have: {
      very: {
        nested: {
          number: {
            here: 0,
          },
        },
      },
    },
  },
});

function increment() {
  count.value.lets.have.very.nested.number.here
+= 1;
}
```

```
<template>
  <button @click="increment">
    {{
count.lets.have.very.nested.number.h
ere }}
  </button>
</template>
```

State – reactive()

- **reactive()**
  - Another way to declare reactive state
  - Makes an object itself reactive (no wrapping in **.value**)
  - Limited value types - it only works for object types
  - Cannot replace entire object
  - Not destructure-friendly - when we destructure a reactive object's property we will lose the reactivity connection

```
<script setup>
import { reactive } from "vue";

const state = reactive({ count: 0 });

function increment() {
  state.count += 1;
}
</script>

<template>
  <button @click="increment">{{ count
}}</button>
</template>
```

Computed

```
<script setup>
import { reactive, computed } from "vue";

const author = reactive({
  name: "Jane Smith",
  books: [],
});

const addBook = () => {
  author.books.push("New book");
};

const booksState = computed(() => {
  return author.books.length > 0 ? "Yes" :
"No";
});
</script>
```

```
<template>
  <h2> Does {{ author.name }} have any
published    books: {{ booksState }}
  </h2>

  <button @click="addBook">
    Add a book
  </button>
</template>
```

Watchers

```
<script setup>
import { ref, watch } from "vue";

const count = ref(0);

function increment() {
  count.value += 1;
}

watch(count, (newVal, oldVal) => {
  if (newVal > 4) {
    console.log(`The new value is ${newVal} and was ${oldVal} earlier`);
  }
});
</script>
```

Watching a nested property

- You should use a "getter" function

```
watch(
  () => data.address.street,
  (newVal) => {
    // logic …
  }
);
```

```
<template>
  <label for="street">
    Street
    <input v-model="data.address.street"
id="street"    type="text"  />
  </label>
  <label for="appartment">
   Appartment
   <input v-model="data.address.appartment"
        id="appartment" type="text"  />
   </label>
</template>
```

```
<script setup>
import { reactive, watch } from "vue";

const data = reactive({ address: { street:
"", appartment: "" } });

watch(data.address.street, (newVal) => {
  console.log("Ho, we have something new",
newVal);
});
</script>
```

Registering components

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

<template>
  <ComponentA />
</template>
```

Summary

- Composition API is alternative to Options API, but not required
- Helps write composable and extendable business logic
- Deeper reactivity control with ref() and reactive()

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank
Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg