FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

FRONT-END FRAMEWORK

REACTIVITY VÀ DATA BINDING

http://www.poly.edu.vn

FPT Education
FPT POLYTECHNIC

Mục tiêu

- Nắm được cơ bản về khái niệm state trong vuejs
- Biết cách khai báo sử dụng ref() và reactive()
- Nắm được khái niệm data binding
- Ứng dụng data binding để giao diện tự động cập nhật theo dữ liệu.
- Thực hành áp dụng Data Binding vào ứng dụng Vue.js.
- Nắm được cách sử dụng class binding và style binding

NỘI DUNG

Phần 1

- Tổng quan về Reactivity
- Reactive State là gì?
- Ref()
- Reactive()

Phần 2

- Data binding
- One-way binding
- Two-way binding
- Class và style binding

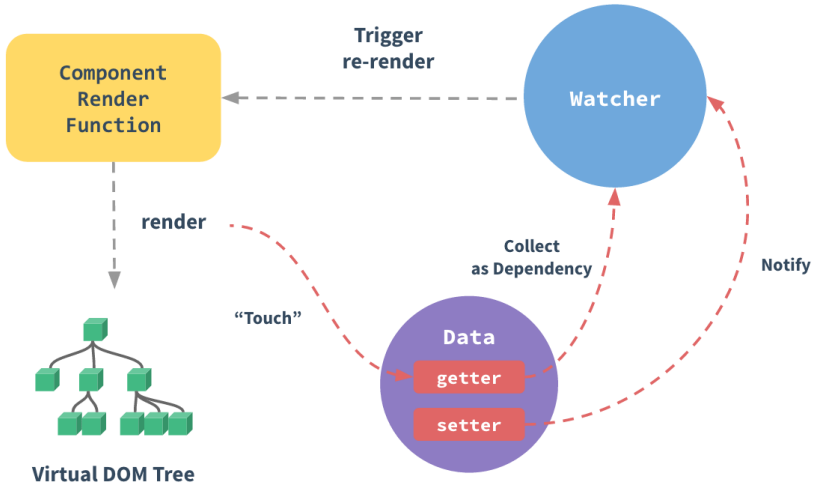
PHẦN 1
REACTIVITY CƠ BẢN

- **Reactivity**: là cơ chế theo dõi và cập nhật giao diện tự động khi dữ liệu thay đổi.

REACTIVITY

Trong các ứng dụng **Single Page App**

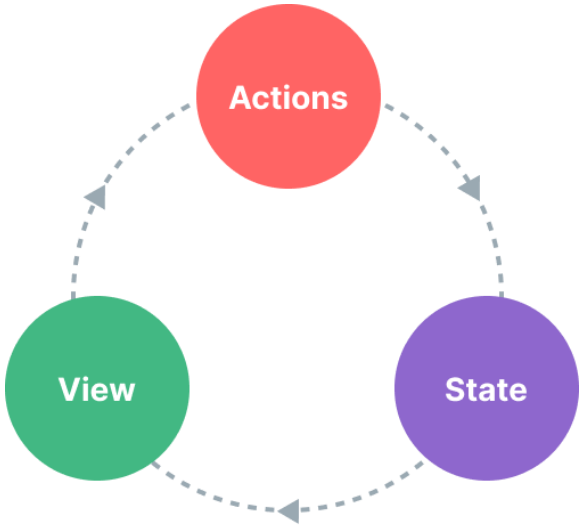
- **Reactive State**: là trạng thái dữ liệu trong Vue.js được quản lý theo **reactive system** (hệ thống phản ứng) mà Vue.js cung cấp, giúp tự động cập nhật các phần của giao diện người dùng khi dữ liệu thay đổi.
- Sử dụng Virtual DOM để hiển thị lại**(rerender)** giao diện khi trạng thái thay đổi.



REACTIVE STATE

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

// state
const count = ref(0);

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

<!-- view -->
<template>{{ count }}</template>
```

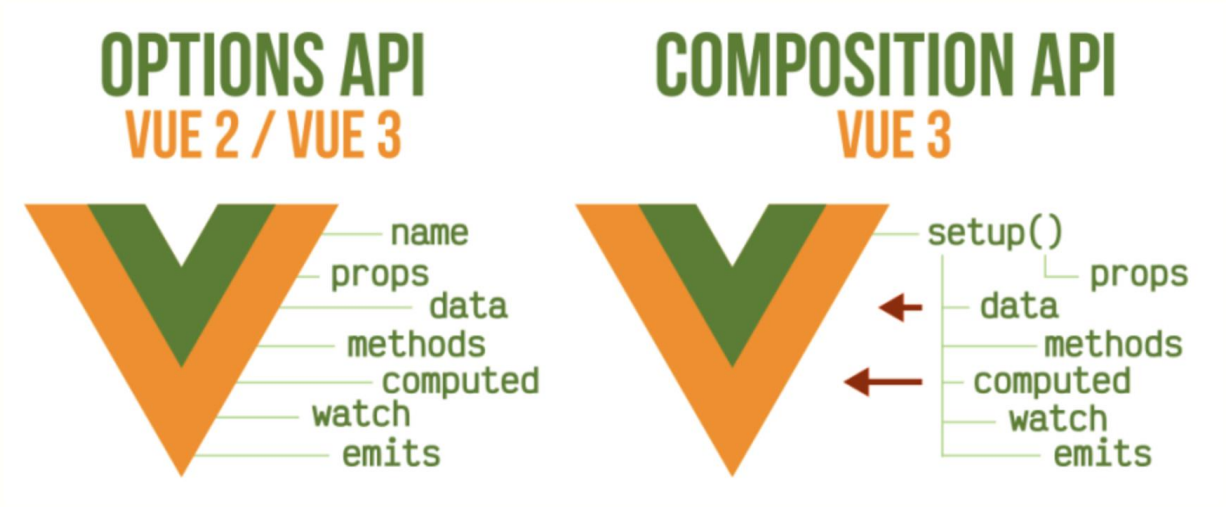
- Trạng thái (**State**): Thông tin chính điều khiển ứng dụng của chúng ta. (ở đây là count = 0)
- Giao diện (**View**): Hiển thị dữ liệu từ state
- Hành động (**Actions**): Các hành động của người dùng hoặc sự kiện có thể thay đổi state



REACTIVE STATE

Có hai cách khai báo reactive state trong Vue.js:

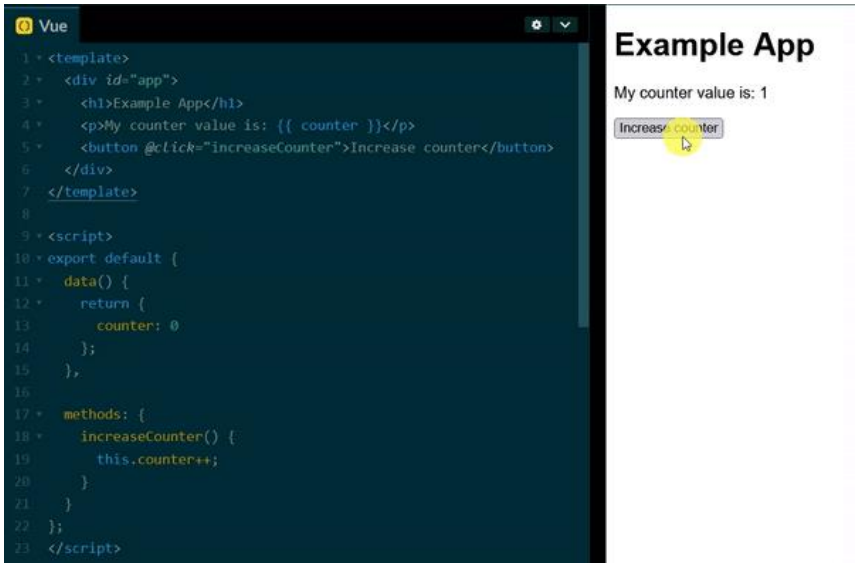
- Options API : Sử dụng phương thức **data()**.
- Composition API :Sử dụng các hàm **ref()** hoặc **reactive().**



OPTIONS API - DATA()

- Phương thức **data()** trả về 1 object chứa tất cả trạng thái phản ứng (reactive state)
- **methods: {}** là đối tượng chứa các hàm để xử lý sự kiện, thay đổi dữ liệu, và thực hiện các tác vụ logic

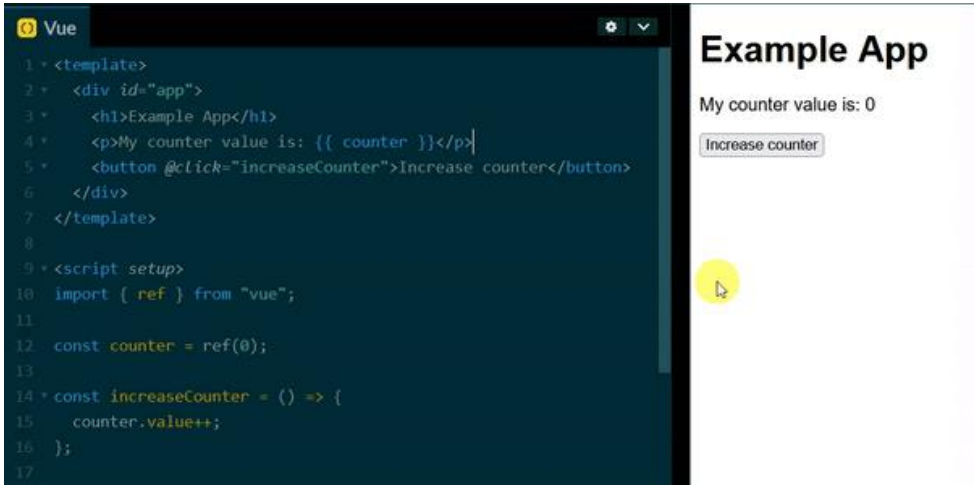
```
<script>
export default {
  data () {
    counter: 0
  },
  methods: {
    increaseCounter() {
      this.counter++
    }
  }
}
</script>
```



COMPOSITION API - REF()

- **ref()** nhận một đối số và trả về một giá trị có thuộc tính **.value**
- **ref()** thường được sử dụng cho các kiểu dữ liệu nguyên thủy như số, chuỗi, boolean
- Sử dụng với <script setup>

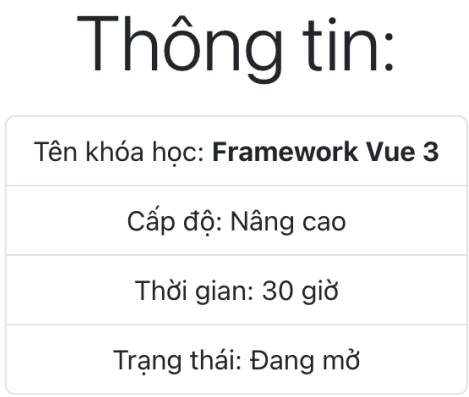
```
<script setup>
import { ref } from "vue";

const counter = ref(0);
console.log(counter);
// giá trị ban đầu { value: 0 }
console.log(counter.value); // 0

// định nghĩa hàm tăng giá trị counter lên 1
const increaseCounter = () => {
    counter.value++; // tăng giá trị counter lên 1
};
</script>
```



Ví dụ - REF()

Sử dụng **Ref** để khai báo các state có kiểu dữ liệu là: number, string, boolean

```
<script setup>
import { ref } from'vue';
// Khai báo các trạng thái sử dụng ref
const courseName = ref('Framework Vue 3');
const courseLevel = ref('Nâng cao');
const courseTime = ref(30); // giờ
const courseIsActive = ref(true);
</script>
<template>
<div class="container my-5">
        <h1 class="display-4 mb-3">Thông tin:</h1>
        <ul class="list-group">
        <li class="list-group-item">
         Tên khóa học: <strong>{{courseName}}</strong>
        </li>
        <li class="list-group-item">Cấp độ: {{courseLevel}}</li>
        <li class="list-group-item">Thời gian: {{courseTime}} giờ</li>
        <li class="list-group-item">
         Trạng thái: {{courseIsActive ? 'Đang mở' : 'Đã đóng'}}
        </li>
        </ul>
</div>
</template>
```

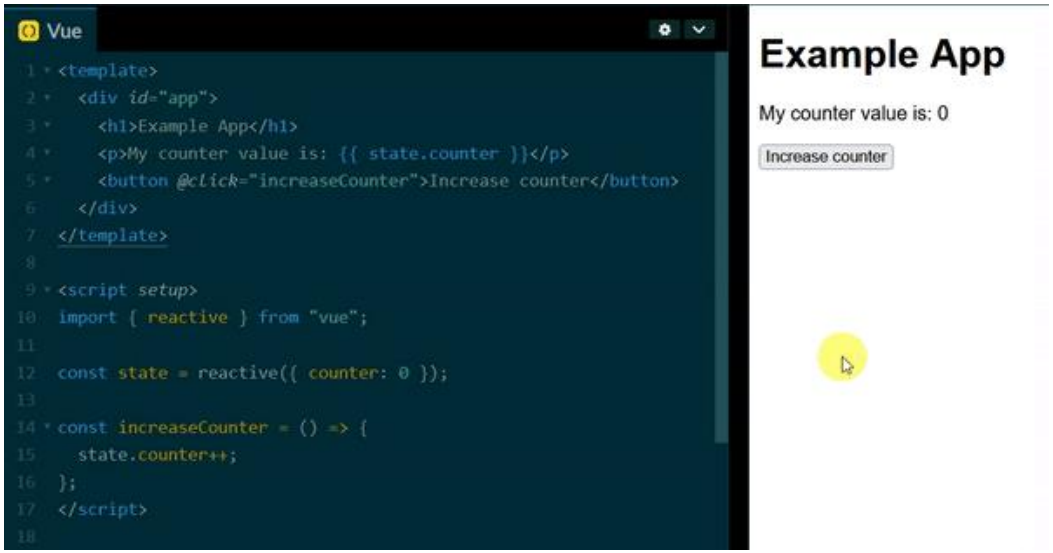


➢ ***Xem ví dụ***

COMPOSITION API - REACTIVE()

- **reactive()** là cách thứ hai để khai báo dữ liệu phản ứng trong Vue 3.
- **reactive()** chỉ áp dụng cho dữ liệu kiểu đối tượng
- Không chấp nhận chuỗi, số, hoặc boolean.
- Khác với **ref()**, **reactive()** không dùng thuộc tính **.value**

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

const state = reactive({ counter: 0 });
console.log(state); // { counter: 0 }

const increaseCounter = () => {
    // tăng giá trị của counter lên 1
    state.counter++;
};
</script>
```



VÍ DỤ - REACTIVE()

Sử dụng **Reactive** để khai báo các state có kiểu dữ liệu là một đối tượng

```
<scriptsetup>
import { ref, reactive } from"vue";
// Khai báo trạng thái với ref: sử dụng cho number, string, boolean
const courseName = ref("Framework Vue 3");
// Khai báo trạng thái với reactive
const student = reactive({
        name:"Nguyễn Văn A",
        gender:"Nam",
        age:20,
        GPA:8.5,
        isActive:true,
});
</script>
<template>
<divclass="container my-5">
        <h1 class="display-4 mb-3">Thông tin:</h1>
        <div class="card">
        <ul class="list-group">
                <li class="list-group-item">Tên khóa học: {{courseName}}</li>
                <li class="list-group-item">Họ tên: {{student.name}}</li>
                <li class="list-group-item">Giới tính: {{student.gender}}</li>
                <li class="list-group-item">Tuổi: {{student.age}}</li>
                <li class="list-group-item">GPA: {{student.GPA}}</li>
                <li class="list-group-item">
                 Trạng thái: {{student.isActive ? "Đang học" : "Ra trường"}}
                </li>
        </ul>
        </div>
</div>
</template>
```

Thông tin:

- Tên khóa học: **Framework Vue 3**
- Cấp độ: Nâng cao
- Thời gian: 30 giờ
- Trạng thái: Đang mở

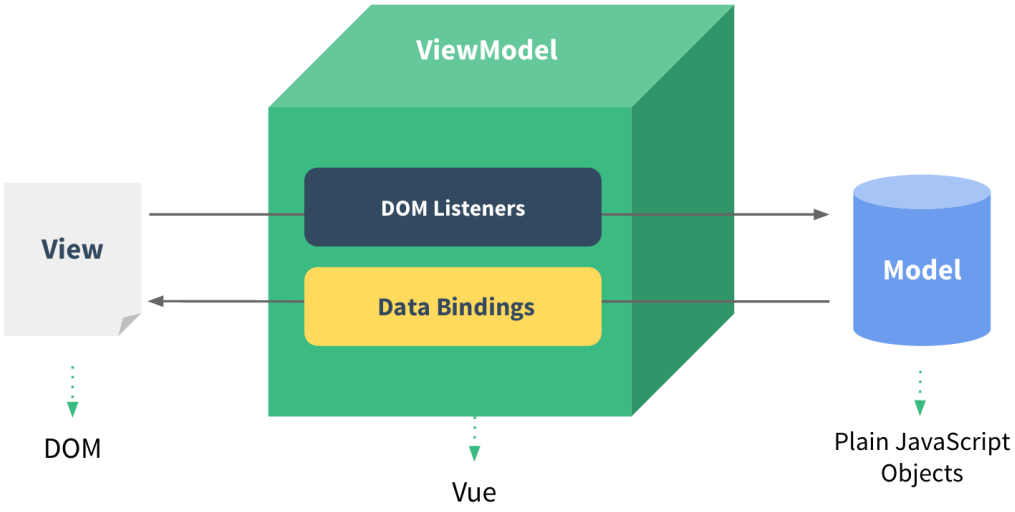
➤ ***Xem ví dụ***


demo

PHẦN 2: DATA BINDING

DATA BINDING

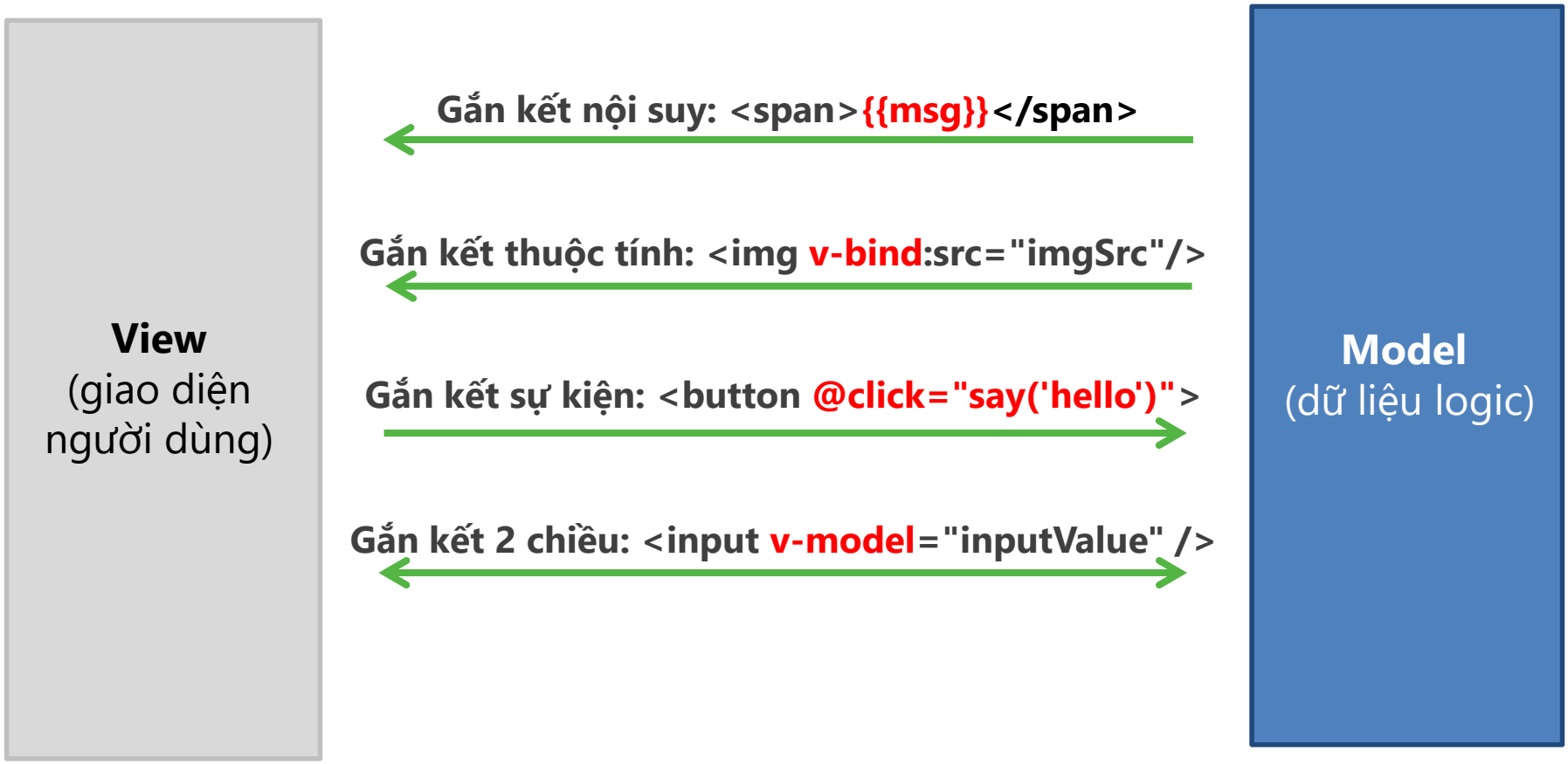
- Data binding **(ràng buộc dữ liệu)** là một trong những khái niệm cốt lõi trong Vue.js.
- Nó là một cơ chế cho phép thiết lập một mối liên kết hai chiều giữa **model** (logic) và **giao diện** (HTML).
- Data binding giúp giao diện tự động cập nhật khi dữ liệu thay đổi và ngược lại.



DATA BINDING

Có 2 loại data binding

- One way binding (Binding một chiều):
- Two way binding (Binding hai chiều):



CHỈ THỊ - DIRECTIVE

- Directive là những thuộc tính đặc biệt trong Vue bắt đầu bằng **v-** được thêm vào các thẻ HTML.
- Mỗi Directive cung cấp một tính năng tương ứng hổ trợ người phát triển web code trên giao diện dễ dàng hơn
- Trong bài này, chúng ta chỉ tập trung vào directive **v-bind** với ngữ cảnh **ràng buộc một chiều**
- Các directive như : v-if, v-for, v-on sẽ được giới thiệu sau.

Xem thêm

ATTRIBUTE BINDING

- **v-bind** là một directive trong Vue.js cho phép kết nối một thuộc tính HTML với một giá trị từ Model trong Vue instance.
- Khi dữ liệu thay đổi, thuộc tính HTML sẽ tự động cập nhật.

➢ Cú pháp:

<element v-bind:attribute="dataValue"></element>

ví dụ:

➢ Rút gọn:

<element :attribute="dataValue"></element>

ví dụ:

Ví dụ

View

model

Ràng buộc thuộc tính src

```
<template>
  <div class="container my-5">
    <h1 class="display-4 mb-3">Thông tin:</h1>
    <div class="card">
      <img
        :src="student.avatar"
        class="card-img-top w-50 mx-auto"
        alt="{{ student.name }}"
      />
      <ul class="list-group">
        <li class="list-group-item">Tên khóa học: {{ courseName }}</li>
        <li class="list-group-item">Họ tên: {{ student.name }}</li>
        <li class="list-group-item">Giới tính: {{ student.gender }}</li>
        <li class="list-group-item">Tuổi: {{ student.age }}</li>
        <li class="list-group-item">GPA: {{ student.GPA }}</li>
        <li class="list-group-item">
          Trạng thái: {{ student.isActive ? 'Đang học' : 'Ra trường' }}
        </li>
      </ul>
    </div>
  </div>
</template>
```

```
<script setup>
import { ref, reactive } from "vue";
// Khai báo trạng thái với ref: sử dụng cho number, string, boolean
const courseName = ref("Framework Vue 3");

// Khai báo trạng thái với reactive
const student = reactive({
    name: "Nguyễn Văn A",
    gender: "Nam",
    age: 20,
    GPA: 8.5,
     // Đường dẫn trực tiếp đến ảnh trong thư mục public
    avatar: "/ongvang.png",
    isActive: true,
});
</script>
```

Thông tin:

- Tên khóa học: Framework Vue 3
- Họ tên: Nguyễn Văn A
- Giới tính: Nam giờ
- Tuổi: 20 giờ
- GPA: 8.5
- Trạng thái: Đang học

➢ ***Xem ví dụ***

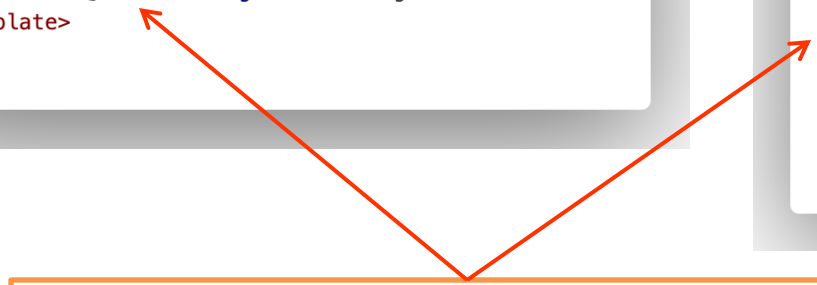
EVENT BINDING

View

```
<template>
    <!-- Hiển thị giá trị của myName -->
    <p>{{ myName }}</p>

    <!-- Nút thay đổi tên -->
    <button @click="changeName">Change name</button>
</template>
```

model

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

// Khai báo biến phản ứng
const myName = ref("Trần Văn A");

// Phương thức thay đổi giá trị của myName
function changeName() {
    myName.value = "Nguyễn Văn B";
// Cập nhật giá trị mới cho myName
}
</script>
```



khai báo sự kiện “click” và phương thức

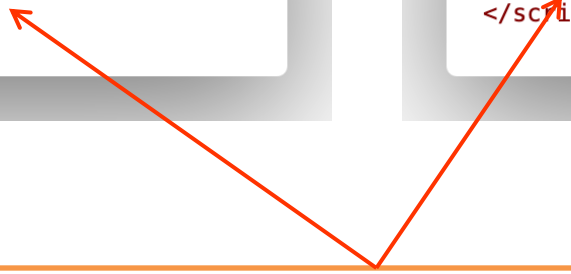
BINDING 2 CHIỀU

View

```
<template>
    {{ myName }}
    <input type="text" v-model="myName" />
</template>
```

model

```
<script setup>
import { ref } from "vue";
const myName = ref("Trần Văn A");
</script>
```



Sử dụng v-model để đồng bộ 2 chiều

CLASS VÀ STYLE BINDINGS

- Vue.js cung cấp hai directive chính để thực hiện binding cho class và style:
- **v-bind:class**: Dùng để binding các lớp CSS cho một phần tử.
  - **cú pháp**: <element **:class**="'active'">
- **v-bind:style**: Dùng để binding các kiểu CSS trực tiếp cho một phần tử.
  - **cú pháp** : <elelement **:style**="{ color: 'red'}">

CLASS BINDINGS - CHUỖI

- Sử dụng class binding dành cho một chuỗi

View

```
<template>
    <div v-bind:class="'active'">{{ content }}</div>
</template>
<style scoped>
/* CSS classes */
.active {
    background-color: green;
    color: white;
    padding: 10px;
}
</style>
```

model

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

const content = ref("Nội dung");
</script>
```

- ***Xem ví dụ***

CLASS BINDINGS - ĐỐI TƯỢNG

- Sử dụng class binding dành cho đối tượng

View

```
<template>
    <div :class="{ active: state.isActive }">Nội dung</div>
    <button @click="toggleActive">Toggle Active State</button>
</template>
<style scoped>
/* CSS classes */
.active {
    background-color: green;
    color: white;
    padding: 10px;
}
</style>
```

model

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

// Khai báo biến reactive để quản lý lớp CSS
const state = reactive({
    isActive: true,
});

// Hàm để chuyển đổi trạng thái lớp CSS
function toggleActive() {
    state.isActive = !state.isActive;
}
</script>
```

- ***Xem ví dụ***

CLASS BINDINGS - MẢNG

- Sử dụng class binding dành cho mảng

View

```
<template>
    <div :class="[activeClass, errorClass]">{{ message }}
</div>
</template>
<style scoped>
/* CSS classes */
.active {
    background-color: green;
    color: white;
    padding: 10px;
}

.text-danger {
    color: red;
}
</style>
```

model

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

// Khai báo state chứa nội dung
const message = ref("Hello Vue 3");
// Khai báo state chứa tên lớp CSS
const activeClass = ref("active");
const errorClass = ref("text-danger");
</script>
```

- ***Xem ví dụ***

STYLE BINDINGS

cho phép tự động hóa việc áp dụng các kiểu CSS (style) vào các phần tử trong ứng dụng.

Áp dụng chuỗi

```
<template>
    <div :style="styleString">This is a styled div.</div>
</template>

<script setup>
// Khai báo chuỗi chứa các thuộc tính CSS
const styleString = "color: blue; background-color: lightgray; font-size: 20px;";
</script>
```

➤ ***Xem ví dụ***

Áp dụng đối tượng

```
<template>
    <div :style="styleObject">This is a styled div.</div>
</template>

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

// Khai báo một đối tượng chứa các thuộc tính CSS
const styleObject = ref({
    color: "blue",
    backgroundColor: "lightgray",
    fontSize: "20px",
});
</script>
```

➤ ***Xem ví dụ***

Áp dụng điều kiện

```
<template>
    <div :style="dynamicStyle">This is a styled div.</div>
</template>

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

// Trạng thái để điều chỉnh kiểu CSS
const isActive = ref(true);

// Tạo một thuộc tính CSS động dựa trên trạng thái
const dynamicStyle = computed(() => ({
    color: isActive.value ? "green" : "red",
    fontSize: isActive.value ? "24px" : "16px",
}));
</script>
```

➤ ***Xem ví dụ***

Ví dụ

- Kết hợp class binding và style binding

View

```
<template>
    <div
        :class="['card', student.isActive ? 'active-card' : 'inactive-card']"
        :style="{
            backgroundColor: student.isActive ? 'lightgreen' : 'lightcoral',
        }"
    >
        <img :src="student.avatar"
            class="card-img-top w-50 mx-auto"
            :alt="student.name"
        />
        <ul class="list-group">
            <li class="list-group-item">Tên khóa học: {{ courseName }}</li>
            <li class="list-group-item">Họ tên: {{ student.name }}</li>
            <li class="list-group-item">Giới tính: {{ student.gender }}</li>
            <li class="list-group-item">Tuổi: {{ student.age }}</li>
            <li class="list-group-item">GPA: {{ student.GPA }}</li>
            <li class="list-group-item">
                Trạng thái: {{ student.isActive ? "Đang học" : "Ra trường" }}
            </li>
        </ul>
    </div>
</template>
<style scoped>
/* CSS classes for dynamic binding */
.active-card {
    border: 2px solid green;
}

.inactive-card {
    border: 2px solid red;
}
</style>
```

model

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

// Sử dụng reactive để quản lý trạng thái của sinh viên
const student = reactive({
    name: "Nguyễn Văn A",
    gender: "Nam",
    age: 20,
    GPA: 8.5,
    avatar: "/ongvang.png",
    isActive: true, // Trạng thái sinh viên đang học hay đã ra trường
});

// Sử dụng ref cho tên khóa học
const courseName = ref("Framework Vue 3");
</script>
```

- ***Xem ví dụ***


demo

SUMARRY

- ☑ Giới thiệu về Reactivity trong vue
- ☑ Giải thích về Reactive State
- ☑ Dùng Data(), Ref() và Reactive() để khai báo state
- ☑ Giải thích về mối liên kết hai chiều giữa model (logic) và view (HTML).
- ☑ Nhấn mạnh tầm quan trọng của việc giao diện tự động cập nhật khi dữ liệu thay đổi.


thank
you!