วิชา การเขียนโปรแกรมบนเว็บ
Web Programming : 4122306

บทที่ 5 : Front-End Development
การพัฒนาส่วนหน้า/ส่วนติดต่อผู้ใช้ด้วย VUE.JS Part-1

Asst. Prof. Dr. Nattapong Songneam

**Department of Computer Science,
Faculty of Science and Technology,
Phranakhon Rajabhat University**

แก้ไขล่าสุด : 05.08.2568

Agenda

Vue.js คืออะไร

1. แนะนำ Vue.js
   - ✅ Vue.js คืออะไร และทำไมถึงควรใช้
   - ✅ เปรียบเทียบ Vue.js กับ React และ Angular
   - ✅ ข้อดีและข้อเสียของ Vue.js

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js
   - ✅ ติดตั้ง Vue.js ผ่าน CDN
   - ✅ ติดตั้ง Vue.js ผ่าน npm (npm create vue@latest)
   - ✅ โครงสร้างโปรเจกต์ Vue.js
   - ✅ สร้างแอป Vue.js แรกของคุณ

Agenda

Vue.js คืออะไร

3. พื้นฐาน Vue.js
- ✅ การสร้าง Vue Instance (createApp)
- ✅ Data Binding ({{ message }})
- ✅ Methods & Event Handling (@click)
- ✅ Computed Properties & Watchers
- ✅ Class และ Style Binding (:class, :style)

4. Vue Directives (คำสั่งพื้นฐาน)
- ✅ v-bind - ผูกค่าให้ attribute
- ✅ v-model - Two-way data binding
- ✅ v-if / v-else / v-show - ควบคุมการแสดงผล
- ✅ v-for - Loop รายการ
- ✅ v-on หรือ @event - การจัดการ event (@click, @keyup)
- ✅ v-html - แสดง HTML ภายใน Vue
- ✅ v-slot - Slot และการใช้งานใน Component

Agenda

Vue.js คืออะไร

5. Component-Based Development
- ✅ การสร้าง Vue Component
- ✅ การใช้ Props และ Emit (props, $emit)
- ✅ Slots - ส่งเนื้อหาภายใน Component
- ✅ การสื่อสารระหว่าง Component (Parent ↔ Child)

6. Vue Router (SPA Navigation)
- ✅ ติดตั้งและตั้งค่า Vue Router
- ✅ การสร้างหน้า (Pages) และ Routes
- ✅ การใช้ router-link และ router-view
- ✅ Dynamic Routes & Route Parameters
- ✅ Navigation Guards

Agenda

Vue.js คืออะไร

7. การจัดการสถานะ (State Management)
- ✅ ref และ reactive ใน Vue 3
- ✅ Vuex (สำหรับ Vue 2) และ Pinia (สำหรับ Vue 3)
- ✅ การใช้ Provide / Inject

8. การเรียก API ด้วย Vue.js
- ✅ ใช้ fetch() และ axios เพื่อดึงข้อมูล
- ✅ การจัดการ State จาก API
- ✅ แสดงข้อมูลแบบ Dynamic

9. Lifecycle Hooks ใน Vue.js
- ✅ onMounted - ทำงานหลังจาก Component สร้างเสร็จ
- ✅ onUpdated - เรียกใช้เมื่อมีการอัปเดต DOM
- ✅ onUnmounted - ทำงานเมื่อ Component ถูกลบออก

Agenda

Vue.js คืออะไร

10. Vue.js + CSS & Animation
- ✅ การใช้ Scoped CSS
- ✅ การใช้ Transition และ Animation
- ✅ Vue Animation Hooks (enter, leave, appear)

11. Vue 3 Composition API
- ✅ setup() - การใช้งานเบื้องต้น
- ✅ reactive() vs ref()
- ✅ computed() และ watchEffect()
- ✅ provide() และ inject()
- ✅ การสร้าง Reusable Functions (Composables)

12. การ Deploy Vue.js
- ✅ การ Build แอป Vue (npm run build)
- ✅ Deploy ไปยัง Netlify, Vercel หรือ Firebase
- ✅ การ Optimize Vue App สำหรับ Production

1. แนะนำ Vue.js

Vue.js คืออะไร

Vue.js คือ JavaScript framework ที่ใช้สำหรับสร้าง User Interface (UI) และ Single Page Applications (SPA) อย่างมีประสิทธิภาพ พัฒนาโดย Evan You และเปิดตัวครั้งแรกในปี 2014

Vue อ่านว่า วิว ออกเสียงแบบ View ในภาษาอังกฤษ จุดเริ่มต้นของ Vue คือทำหน้าที่เป็น View ใน MVC (Model View Controller) ซึ่งเป็น JavaScript Framework สำหรับการพัฒนา UI (User Interface) ปัจจุบัน คือ Vue เวอร์ชัน 3

1. แนะนำ Vue.js

Vue.js คืออะไร

Vue.js คืออะไร?

- **Vue.js (อ่านว่า "วิว") เป็น JavaScript Framework ที่ใช้ในการพัฒนา User Interface (UI) และ Single Page Applications (SPA) ได้อย่างมีประสิทธิภาพ**
- **Vue ถูกออกแบบให้ เรียนรู้ง่าย และ ปรับแต่งได้ง่าย ทำให้เป็นตัวเลือกที่ยอดเยี่ยมสำหรับนักพัฒนาทั้งมือใหม่และมืออาชีพ**

Vue.js คืออะไร

ทำไมถึงควรใช้ Vue.js?

1. ใช้งานง่ายและเรียนรู้ได้เร็ว
   - Vue มีโครงสร้างที่เข้าใจง่าย
   - ใช้ HTML, CSS และ JavaScript พื้นฐาน
   - Syntax คล้าย JavaScript ธรรมดา ไม่ซับซ้อน
2. รองรับการพัฒนาแบบ Component-Based
   - Vue ใช้แนวคิด Component-Based Development
   - ช่วยให้การพัฒนาเว็บ ง่ายขึ้น และ ดูแลรักษาง่าย
3. ประสิทธิภาพสูงและเบากว่า Framework อื่นๆ
   - ขนาดเล็ก (~20KB Gzipped) โหลดเร็ว
   - ใช้ Virtual DOM เพื่อเรนเดอร์ UI อย่างมีประสิทธิภาพ
   - เร็วกว่า Angular และมีประสิทธิภาพใกล้เคียง React

Vue.js คืออะไร

ทำไมถึงควรใช้ Vue.js?

4. มี Two-Way Data Binding
   - Vue มี v-model ซึ่งทำให้ UI อัปเดตข้อมูลแบบเรียลไทม์
5. รองรับการทำงานร่วมกับโปรเจกต์อื่นได้ง่าย
   - Vue สามารถใช้ในโปรเจกต์ที่มีอยู่แล้วได้ โดยไม่ต้องเปลี่ยนโครงสร้างทั้งหมด
   - รองรับการใช้งานกับ JavaScript Library อื่นๆ เช่น jQuery, Axios, Firebase
6. มี Ecosystem และ Community ใหญ่
   - มี Vue Router สำหรับการทำ Routing
   - มี Pinia/Vuex สำหรับการจัดการ State
   - มีเครื่องมือช่วยพัฒนา เช่น Vue DevTools
   - มี Community ใหญ่ และเอกสารที่เข้าใจง่าย

1. แนะนำ Vue.js

Vue.js คืออะไร

สรุป

✓ Vue.js เป็น JavaScript Framework ที่ง่ายต่อการเรียนรู้ และมีประสิทธิภาพสูง

✓ ใช้ Component-Based Development ช่วยให้การพัฒนาแอปพลิเคชันเป็นระเบียบ

✓ มี Two-Way Data Binding ทำให้ UI อัปเดตข้อมูลได้อัตโนมัติ

✓ เบา เร็ว และสามารถใช้ร่วมกับโปรเจกต์ที่มีอยู่ได้ง่าย

✓ มี Ecosystem ครบเครื่อง พร้อมใช้งาน

🎯 Vue.js เหมาะสำหรับใคร?

✅ นักพัฒนาที่ต้องการ Framework ที่ใช้งานง่าย

✅ ผู้เริ่มต้นที่ต้องการพัฒนาเว็บแบบ Interactive

✅ นักพัฒนาที่ต้องการพัฒนา Single Page Applications (SPA)

1. แนะนำ Vue.js

Vue.js คืออะไร

ใช้ Vue.js เมื่อไหร่?

- เมื่อพัฒนาเว็บแอปพลิเคชันที่ต้องการความ Interactive
- เมื่อสร้าง Single Page Applications (SPA)
- เมื่อทำ Progressive Web Applications (PWA)
- เมื่ออยากใช้ Frontend Framework ที่เบาและใช้งานง่ายกว่า React หรือ Angular

1. แนะนำ Vue.js

Vue.js คืออะไร

คุณสมบัติหลักของ Vue.js

1. ง่ายต่อการเรียนรู้ – ใช้ HTML, CSS และ JavaScript เป็นหลัก
2. Reactivity (การตอบสนองแบบเรียลไทม์) – ใช้ reactive data binding ทำให้ UI อัปเดตตามข้อมูลโดยอัตโนมัติ
3. Component-based (โครงสร้างแบบคอมโพเนนต์) – แบ่งโค้ดออกเป็นส่วนเล็ก ๆ ทำให้จัดการได้ง่าย
4. Virtual DOM – ทำให้การเรนเดอร์ UI มีประสิทธิภาพและเร็วขึ้น
5. Directives (ไดเรกทีฟ) – เช่น v-bind, v-if, v-for, v-model ใช้ควบคุมพฤติกรรมของ HTML
6. Vue Router – สำหรับจัดการเส้นทาง (Routing) ใน Single Page Application
7. Vuex หรือ Pinia – ใช้จัดการ State Management ในแอปพลิเคชันที่ซับซ้อน

วิธีในการติดตั้ง Vue.js

1. ติดตั้ง Vue.js

มีหลายวิธีในการติดตั้ง Vue.js เช่น

- **วิธีที่ 1: ติดตั้งแบบง่ายผ่าน CDN (เหมาะสำหรับมือใหม่/ทดลอง)**
- **วิธีที่ 2: ติดตั้งด้วย Vue CLI (สำหรับโปรเจกต์ที่จริงจัง)**
- **วิธีที่ 3: ใช้ Vite (ทางเลือกใหม่ เร็วและเบากว่า CLI)**

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

วิธีในการติดตั้ง Vue.js

การติดตั้ง Vue.js วิธีที่ 1 : โดยใช้ CDN (เหมาะสำหรับการทดสอบ)

ขั้นตอนที่ 1: สร้างไฟล์ HTML เปล่า ๆ
สร้างไฟล์ชื่อ index.html แล้วใส่โค้ดพื้นฐานตามนี้:
Vue01.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Vue.js CDN Example</title>
  <!-- ใส่ CDN ของ Vue.js -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14"></script>
  <!-- หรือ Vue 3 -->
  <!-- <script src="https://unpkg.com/vue@3.3.4/dist/vue.global.js"></script> -->
</head>
<body>
  <div id="app">
    <h1>{{ message }}</h1>
  </div>
  <script>
    new Vue({
      el: '#app',
      data: {
        message: 'สวัสดีจาก Vue.js (CDN)!'
      }
    });
  </script>
</body>
</html>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

วิธีในการติดตั้ง Vue.js

การติดตั้ง Vue.js วิธีที่ 1 : โดยใช้ CDN (เหมาะสำหรับการทดสอบ)

ขั้นตอนที่ 1: สร้างไฟล์ HTML เปล่า ๆ
สร้างไฟล์ชื่อ index.html แล้วใส่โค้ดพื้นฐานตามนี้:
Vue01.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>My first Vue page</title>
</head>
<body>

</body>
</html>
```

**Step 2: Add a **

Vue needs an HTML element on your page to connect to.
Put a tag inside the <body> tag and give it an id:

```
<body>
  <div id="app"></div>
</body>
```

Step 3: Add a link to Vue

To help our browser to interpret our Vue code, add this <script> tag:

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"
></script>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

วิธีในการติดตั้ง Vue.js

การติดตั้ง Vue.js วิธีที่ 1 : โดยใช้ CDN (เหมาะสำหรับการทดสอบ)

Step 4: Add the Vue instance

Now we need to add our Vue code.
This is called the Vue instance and can contain data and methods and other things, but now it just contains a message.
On the last line in this <script> tag our Vue instance is connected to the tag:

```
<div id="app"></div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        message: "Hello World!"
      }
    }
  })
 app.mount('#app')
</script>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

วิธีในการติดตั้ง Vue.js

การติดตั้ง Vue.js วิธีที่ 1 : โดยใช้ CDN (เหมาะสำหรับการทดสอบ)

Step 5: Display 'message' with Text Interpolation

Finally, we can use text interpolation, a Vue syntax with double curly braces {{ }} as a placeholder for data.

```
<div id="app"> {{ message }} </div>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

วิธีในการติดตั้ง Vue.js

Vue01_bg.html

การติดตั้ง Vue.js วิธีที่ 1 : โดยใช้ CDN (เหมาะสำหรับการทดสอบ)

```
<!DOCTYPE html>
<html>
<head>
  <title>My first Vue page</title>
  <style>
    #app {
      display: inline-block;
      padding: 10px;
      font-size: x-large;
      background-color: Orange;
    }
  </style>
</head>
<body>
<h1>Vue Example</h1>
<p>The message is taken from 'data' inside the Vue
instance by writing {{ message }} inside the div with
id="app".</p>
```

```
<div id="app">
  {{ message }}
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
   data() {
    return {
     message: "Hello World!"
    }
   }
  })
  app.mount('#app')
</script>
</body>
</html>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

วิธีในการติดตั้ง Vue.js

การติดตั้ง Vue.js วิธีที่ 1 : โดยใช้ CDN (เหมาะสำหรับการทดสอบ)

ขั้นตอนที่ 1: สร้างไฟล์ HTML เปล่า ๆ
สร้างไฟล์ชื่อ index.html แล้วใส่โค้ดพื้นฐานตามนี้:
Vue01.html

✅ **หมายเหตุ**
📝 **Vue 2 และ Vue 3 มีโครงสร้างแตกต่างกันเล็กน้อยในการเขียน**
Vue 2 ใช้ new Vue({...})
Vue 3 ใช้ Vue.createApp({...}).mount('#app')

Vue Version	ลิงก์ CDN
Vue 2.x	https://cdn.jsdelivr.net/npm/vue@2.6.14
Vue 3.x	https://unpkg.com/vue@3.3.4/dist/vue.global.js

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 3 : Vue01_message.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <title>Vue.js Example</title>
</head>
<body>
    <div id="app">
        <h1>{{ message }}</h1>
        <button @click="updateMessage">คลิกฉัน</button>
    </div>
    <script>
        const { createApp } = Vue;
        createApp({
            data() {
                return {
                    message: "Hello Vue!"
                };
            },
            methods: {
                updateMessage() {
                    this.message = "ข้อความเปลี่ยนแล้ว!";
                }
            }
        }).mount("#app");
    </script>

    <style>
        h1 {
            color: blue;
        }
    </style>
</body>
</html>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 3 : Vue01_message.html

ผลลัพธ์

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 4 : Vue02_Hello_Vue.html

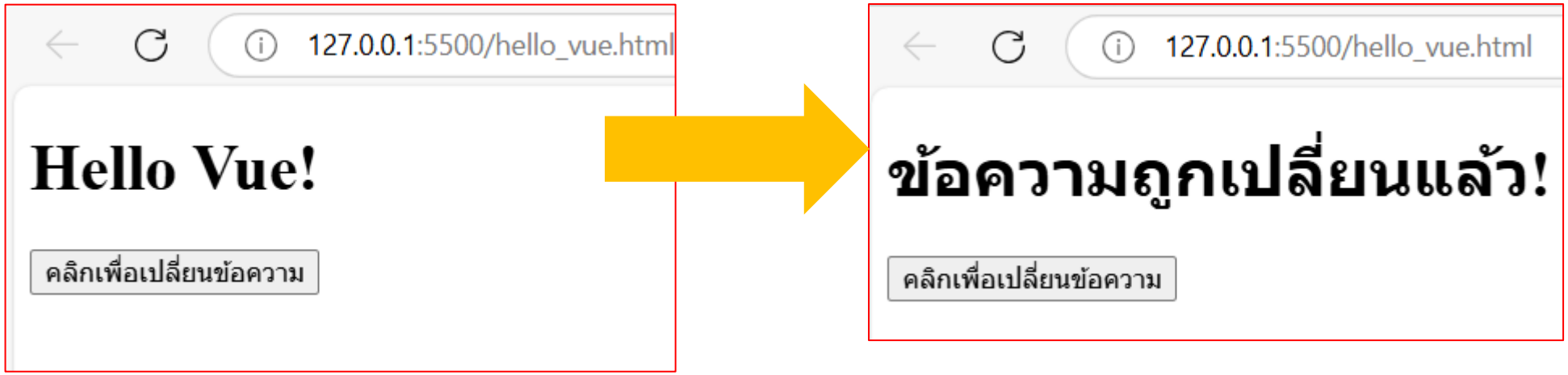
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <title>Vue.js Example</title>
</head>
<body>
    <div id="app">
        <h1>{{ message }}</h1>
        <button @click="updateMessage">คลิกเพื่อเปลี่ยนข้อความ</button>
    </div>
    <script>
        const { createApp } = Vue;
        createApp({
            data() {
                return {
                    message: "Hello Vue!"
                };
            },
            methods: {
                updateMessage() {
                    this.message = "ข้อความถูกเปลี่ยน
แล้ว!";
                }
            }
        }).mount("#app");
    </script>
</body>
</html>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 4 : Vue02_Hello_Vue.html

ผลลัพธ์



✅ อธิบาย

- ใช้ {{ message }} แสดงค่าข้อความ
- ใช้ @click="updateMessage" เพื่อเปลี่ยนค่า message เมื่อคลิกปุ่ม

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 5 : Input กับ v-model (Two-Way Binding)

Vue03_Two_Way_Binding.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <title>Vue.js Example</title>
</head>
<body>
    <div id="app">
        <input v-model="name" placeholder="ป้อนชื่อของคุณ">
        <p>สวัสดี, {{ name }}!</p>
    </div>
    <script>
```

```
const { createApp } = Vue;

        createApp({
            data() {
                return {
                    name: ""
                };
            }
        }).mount("#app");
    </script>
</body>
</html>
```

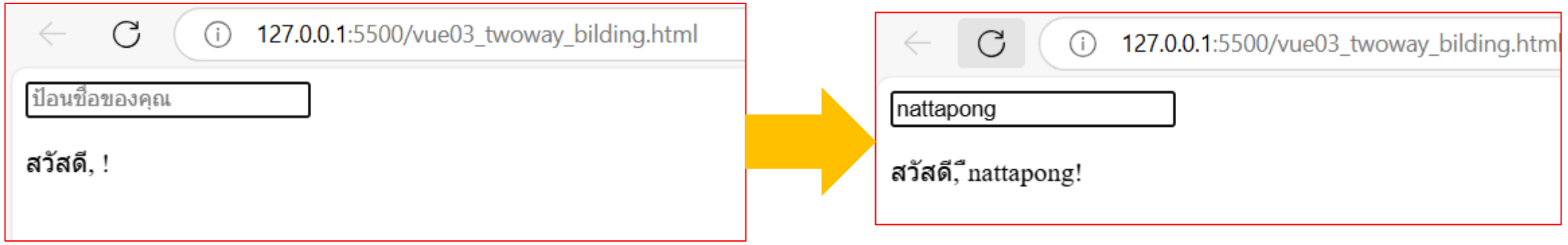
Agenda

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 3 : Input กับ v-model (Two-Way Binding)

Vue03_Two_Way_Binding.html

ผลลัพธ์



✅ อธิบาย

- v-model="name" ทำให้ค่า name เปลี่ยนไปตาม input ที่ป้อน

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 6 : เพิ่มปุ่ม และ Event Handling (Methods)

Vue04_Event_Handling.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Vue Event</title>
  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14"></script>
</head>
<body>
  <div id="app">
    <h1>{{ count }}</h1>
    <button @click="add">เพิ่ม</button>
  </div>
 <script>
    new Vue({
      el: '#app',
      data: {
        count: 0
      },
      methods: {
        add() {
          this.count++;
        }
      }
    });
  </script>
</body>
</html>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 6 : เพิ่มปุ่ม และ Event Handling (Methods)

Vue04_Event_Handling.html

ผลลัพธ์การทำงานการใช้ method add() และ event @click

ผลลัพธ์

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 7 : ใช้ v-model กับ Input

Vue05_V_model_Input.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Vue v-model</title>
  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14"></script>
</head>
<body>
  <div id="app">
    <input v-model="name" placeholder="กรอกชื่อ">
    <p>สวัสดี, {{ name }}</p>
  </div>
```

```
  <script>
    new Vue({
      el: '#app',
      data: {
        name: ''
      }
    });
  </script>
</body>
</html>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

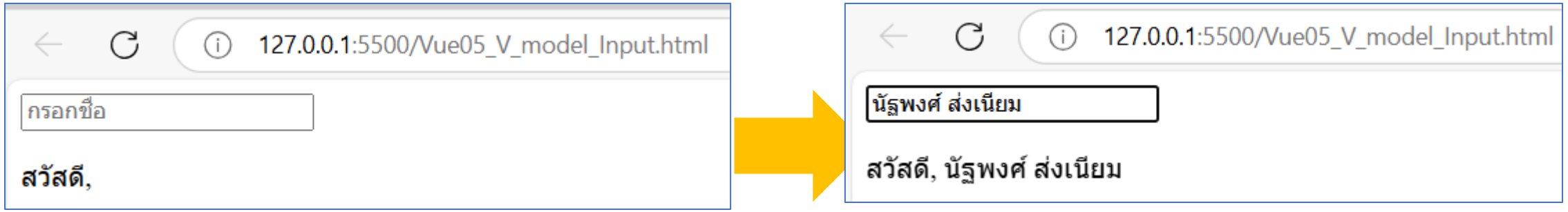
การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 7 : ใช้ v-model กับ Input

Vue05_V_model_Input.html

สิ่งที่เกิดขึ้น: เมื่อพิมพ์ใน input, ค่า name จะเปลี่ยนและแสดงผลทันที

ผลลัพธ์



2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 8 : ใช้ v-if, v-show, และ v-for

Vue06_v_if_v_show_v_for.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Vue Condition & Loop</title>
  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14"></script>
</head>
<body>
  <div id="app">
    <button @click="show = !show">แสดง/ซ่อน</button>
    <p v-if="show">นี่คือข้อความที่ซ่อน/แสดงได้</p>
    <h3>รายชื่อ:</h3>
 <ul>
      <li v-for="(person, index) in people" :key="index">
        {{ index + 1 }}. {{ person }}
      </li>
</ul>
  </div>
  <script>
    new Vue({
      el: '#app',
      data: {
        show: true,
        people: ['สมชาย', 'สมหญิง', 'สมปอง']
      }
    });
  </script>
</body>
</html>
```

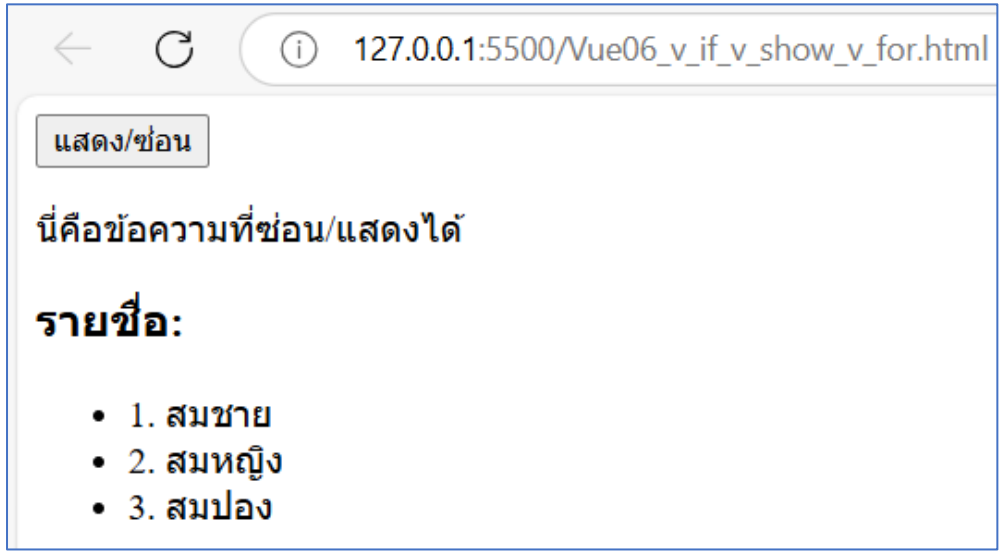
2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 8 : ใช้ v-if, v-show, และ v-for

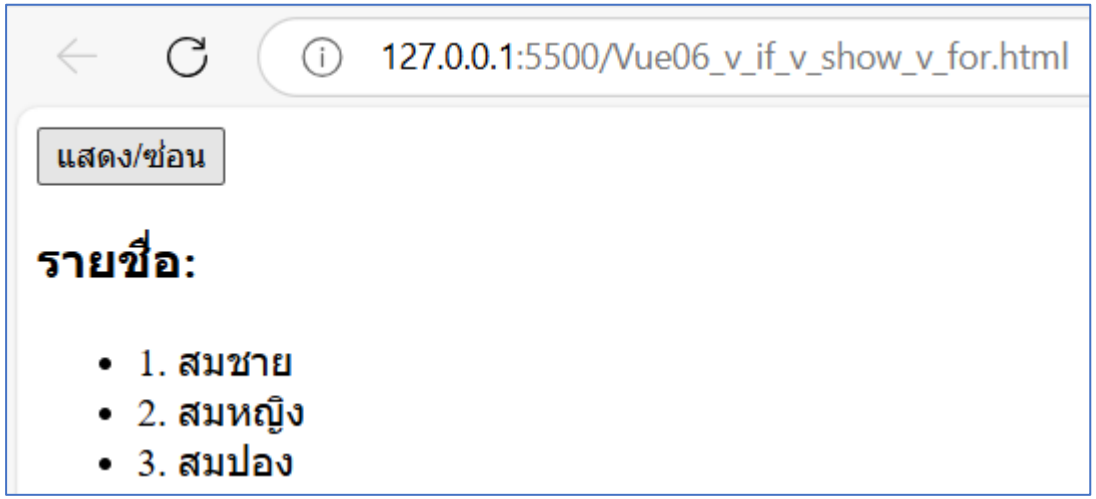
Vue06_v_if_v_show_v_for.html

ผลลัพธ์



สิ่งที่เกิดขึ้น:

- v-if ใช้ซ่อน/แสดงข้อความ
- v-for ใช้แสดงรายชื่อจาก array



2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

ตัวอย่างที่ 9 : ใช้ computed และ watch

Vue07_computed_watch.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Vue Computed & Watch</title>
  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14"></script>
</head>
<body>
  <div id="app">
    <input v-model="num1" type="number" placeholder="ตัวเลขที่ 1">
    <input v-model="num2" type="number" placeholder="ตัวเลขที่ 2">
    <p>ผลรวม: {{ sum }}</p>
  </div>
  <script>
  new Vue({
      el: '#app',
        data: {
          num1: 0,
          num2: 0
        },
        computed: {
          sum() {
            return parseFloat(this.num1) + parseFloat(this.num2);
          }
        },
        watch: {
          sum(newVal, oldVal) {
            console.log(`ค่าผลรวมเปลี่ยนจาก ${oldVal} เป็น ${newVal}`);
          }
        }
      });
  </script>
</body>
</html>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

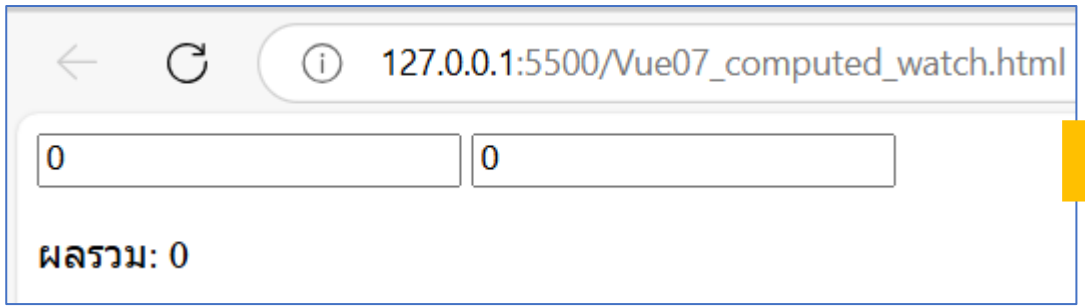
ตัวอย่างที่ 9 : ใช้ computed และ watch

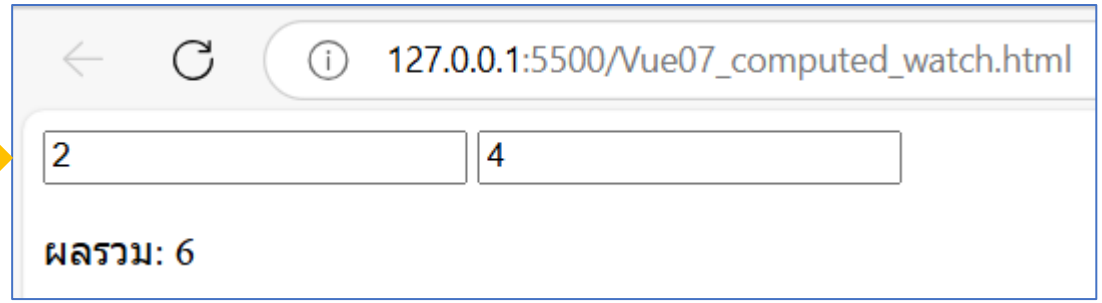
Vue07_computed_watch.html

ผลลัพธ์

สิ่งที่เกิดขึ้น:

- computed คำนวณผลรวมอัตโนมัติ
- watch ดูการเปลี่ยนแปลงของผลรวมและพิมพ์ใน console





2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

✅ ตัวอย่าง Todo App (Vue 3 แบบ CDN)

Vue08_TodoApp.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Vue Todo App</title>
  <script
src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  <style>
    body { font-family: Arial; padding: 2rem; }
    .done { text-decoration: line-through; color: gray; }
    input[type="text"] { padding: 6px; margin-right: 10px;
}
    button { padding: 6px 10px; margin-left: 5px; }
  </style>
</head>
```

```
<body>
  <div id="app">
    <h2>📝 Todo App</h2>
    <input v-model="newTodo" @keyup.enter="addTodo"
placeholder="เพิ่มรายการใหม่...">
  <button @click="addTodo">เพิ่ม</button>
    <ul>
      <li v-for="(todo, index) in todos" :key="index">
        <span :class="{ done: todo.done }"
@click="toggleDone(index)">
          {{ todo.text }}
        </span>
        <button @click="removeTodo(index)">ลบ</button>
      </li>
    </ul>
  </div>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

✅ ตัวอย่าง Todo App (Vue 3 แบบ CDN)

Vue08_TodoApp.html

```
<script>
  const app = Vue.createApp({
    data() {
      return {
        newTodo: '',
        todos: []
      };
    },
    methods: {
      addTodo() {
        if (this.newTodo.trim() !== '') {
          this.todos.push({ text: this.newTodo, done: false });
          this.newTodo = '';
        }
      },
      removeTodo(index) {
        this.todos.splice(index, 1);
      },
      toggleDone(index) {
        this.todos[index].done = !this.todos[index].done;
      }
    }
  });
  app.mount('#app');
</script>
</body>
</html>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

✅ ตัวอย่าง Todo App (Vue 3 แบบ CDN)

Vue08_TodoApp.html

✅ สิ่งที่ใช้ในโปรเจกต์นี้:

ฟีเจอร์	ตัวอย่าง
Data Binding	{{ todo.text }}, v-model="newTodo"
Event Handling	@click="addTodo", @click="removeTodo"
v-for loop	วนรายการ todos
v-model	ผูก input กับตัวแปร
Class Binding	:class="{ done: todo.done }"
Method	addTodo, removeTodo, toggleDone

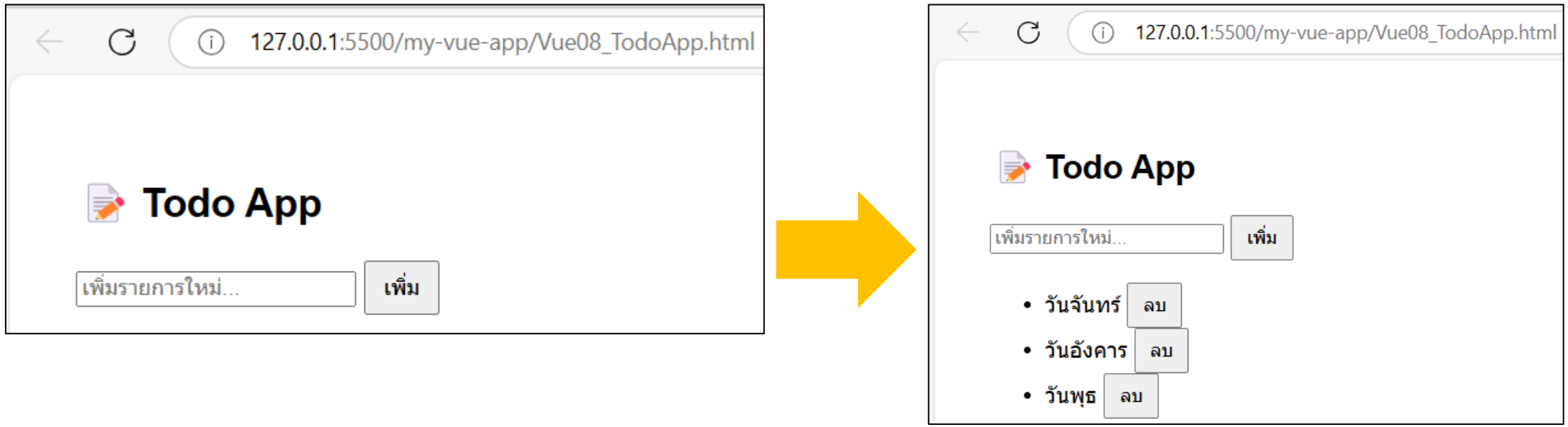
2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

การใช้งาน Vue3 ด้วยวิธี CDN

✅ ตัวอย่าง Todo App (Vue 3 แบบ CDN)

Vue08_TodoApp.html

ผลลัพธ์



2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

2 การติดตั้ง Vue.js วิธีที่ 2: ด้วย Vue CLI

🔧 วิธีติดตั้ง Vue 3 ด้วย Vue CLI (Step-by-step)

1

✅ ขั้นตอนที่ 1: ติดตั้ง Node.js

Vue CLI ต้องการ Node.js ในเครื่องก่อน

- ดาวน์โหลด Node.js ได้ที่: https://nodejs.org/
- แนะนำให้ใช้ LTS (Long-Term Support)
- เมื่อติดตั้งเสร็จ ให้เช็กว่า Node และ npm ติดตั้งแล้วหรือยัง โดยรันใน terminal:

```
node -v
npm -v
```

```
PS D:\ComInBuss> npm -v
10.9.2
PS D:\ComInBuss> node -v
v22.14.0
```

2

✅ ขั้นตอนที่ 2: ติดตั้ง Vue CLI

เปิด Terminal หรือ Command Prompt แล้วพิมพ์คำสั่งนี้:

```
npm install -g @vue/cli
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

2 การติดตั้ง Vue.js วิธีที่ 2: ด้วย Vue CLI

3 ✅ **ขั้นตอนที่ 3: สร้างโปรเจกต์ใหม่**

จากนั้น Vue CLI จะถาม:

- Default preset (Vue 3, Babel, ESLint) – ให้กด Enter
- หรือเลือก manually แล้วเลือก Vue 3 + Options ที่คุณต้องการ (เช่น TypeScript, Router, Vuex เป็นต้น)

4 ✅ **ขั้นตอนที่ 4: เข้าไปยังโฟลเดอร์โปรเจกต์**

```
cd my-vue-app
```

5 ✅ **ขั้นตอนที่ 5: เริ่มรันโปรเจกต์ (start development server)**

```
npm run serve
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

2 การติดตั้ง Vue.js วิธีที่ 2: ด้วย Vue CLI

6

หากสำเร็จ คุณจะเห็นข้อความประมาณนี้:

```
App running at:
  - Local:   http://localhost:8080/

เปิด browser แล้วเข้าไปที่ http://localhost:8080 เพื่อดูแอปของคุณ 🎉
```

7

✅ โครงสร้างโปรเจกต์เบื้องต้นจะเป็นแบบนี้:

```
my-vue-app/
├── node_modules/
├── public/
├── src/
│   ├── assets/
│   ├── components/
│   └── App.vue
│   └── main.js
├── package.json
└── README.md
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

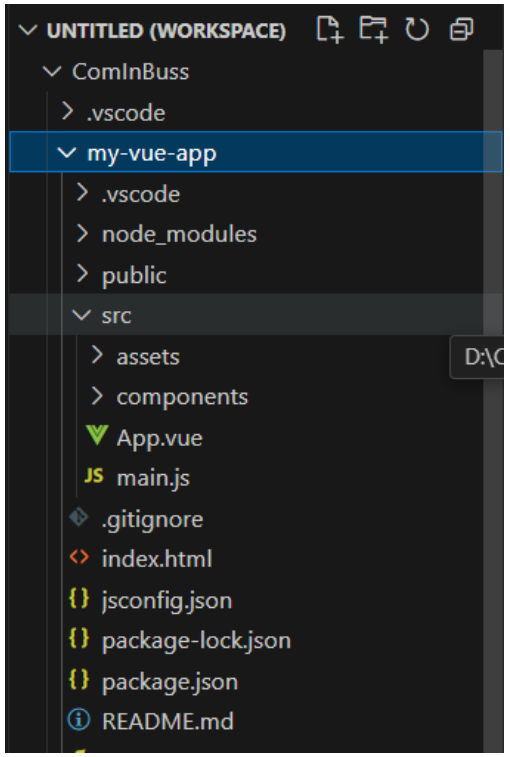
ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

npm create vue@latest

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน




```
my-vue-app/
├── node_modules/          # ไฟล์ dependencies จาก npm
├── public/                # ไฟล์ที่เข้าถึงได้จากเว็บ เช่น index.html
│   ├── favicon.ico        # ไอคอนของเว็บ
│   └── index.html         # จุดเริ่มต้นของเว็บ
├── src/                   # ไฟล์ซอร์สโค้ดหลัก
│   ├── assets/            # เก็บไฟล์รูปภาพ, CSS, ไอคอน ฯลฯ
│   ├── components/        # ไฟล์ Vue Components
│   ├── views/             # ไฟล์ของแต่ละหน้า (ใช้กับ Vue Router)
│   ├── App.vue            # Component หลักของแอป
│   ├── main.js            # จุดเริ่มต้นของแอป Vue
│   └── router/            # (ถ้ามี) ใช้สำหรับจัดการ Routing
├── .gitignore             # รายการไฟล์ที่ไม่ต้องการให้ git track
├── package.json           # รายละเอียดโปรเจกต์ และ dependencies
├── vite.config.js         # การตั้งค่า Vite (Build Tool)
└── README.md              # คำอธิบายโปรเจกต์
```

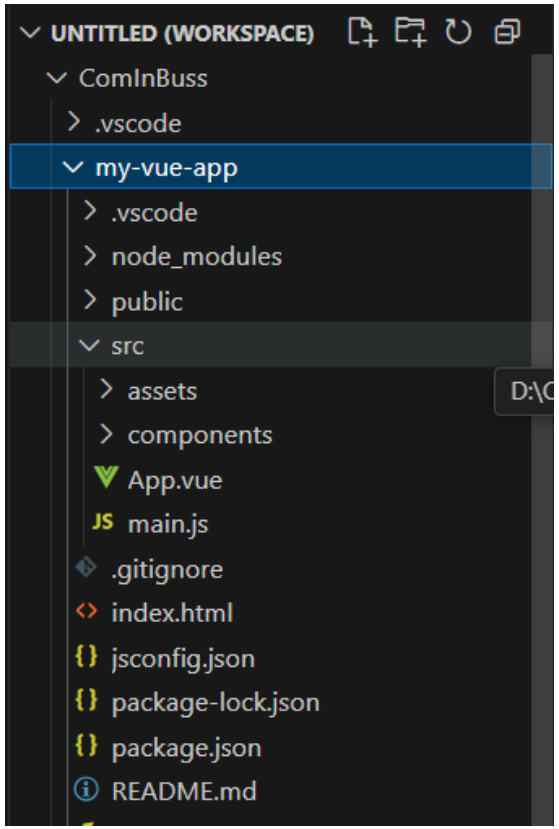
2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

```
npm create vue@latest
```



Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

1. public/ – ไฟล์ที่เข้าถึงจากเบราว์เซอร์ได้โดยตรง

- index.html: เป็นไฟล์หลักของแอป Vue.js
- favicon.ico: ไอคอนของเว็บ
  - หมายเหตุ: Vue จะ Inject แอปลงใน ใน index.html

2. src/ – ไฟล์โค้ดหลักของ Vue

- ✅ App.vue – Component หลักของแอป
- ✅ main.js – จุดเริ่มต้นของ Vue App
- ✅ assets/ – เก็บไฟล์รูปภาพ, CSS
- ✅ components/ – เก็บ Vue Components
- ✅ views/ – เก็บหน้า (ใช้กับ Vue Router)
- ✅ router/ – (ถ้ามี) ใช้สำหรับ Vue Router

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

```
npm create vue@latest
```

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

ตัวอย่างไฟล์ main.js

```
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

ตัวอย่างไฟล์ App.vue

```
<template>
  <h1>Welcome to Vue.js</h1>
</template>
<script>
export default {
  name: "App"
};
</script>
<style>
h1 {
  color: blue;
}
</style>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

```
npm create vue@latest
```

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

📌 3. components/ – เก็บ Vue Components ที่นำมาใช้ซ้ำได้

✅ ตัวอย่าง Component HelloWorld.vue

```
<template>
  <h1>สวัสดี {{ name }}!</h1>
</template>

<script>
export default {
  props: ["name"]
};
</script>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

```
npm create vue@latest
```

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

📌 3. components/ – เก็บ Vue Components ที่นำมาใช้ซ้ำได้

✅ ตัวอย่าง Component HelloWorld.vue

```
<template>
  <h1>สวัสดี {{ name }}!</h1>
</template>

<script>
export default {
  props: ["name"]
};
</script>
```

✅ นำไปใช้ใน App.vue

```
<HelloWorld name="Vue.js" />
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

```
npm create vue@latest
```

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

📌 4. views/ – เก็บไฟล์ของแต่ละหน้า (ใช้กับ Vue Router)

เช่น

- Home.vue
- About.vue

ตัวอย่าง Home.vue

```
<template>
  <h1>หน้าแรก</h1>
</template>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

```
npm create vue@latest
```

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

📌 5. router/ – ไฟล์กำหนด Routing ของแอป
(ใช้เมื่อมี Vue Router)

✅ ตัวอย่าง router/index.js

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home.vue'
import About from '../views/About.vue'
const routes = [
  { path: '/', component: Home },
  { path: '/about', component: About }
]
const router = createRouter({
  history: createWebHistory(),
  routes
})
export default router
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

```
npm create vue@latest
```

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

📌 5. router/ – ไฟล์กำหนด Routing ของแอป
(ใช้เมื่อมี Vue Router)

✅ ตัวอย่าง router/index.js

✅ ใช้ใน main.js

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'

createApp(App).use(router).mount('#app')
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

```
npm create vue@latest
```

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

📌 **6. package.json – รายละเอียดโปรเจกต์และ dependencies**

เก็บข้อมูล dependencies และ scripts ที่ใช้รันโปรเจกต์

```
{
  "name": "my-vue-app",
  "version": "0.1.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "serve": "vite บทที่ 5"
  },
  "dependencies": {
    "vue": "^3.3.0"
  }
}
```

✅ คำสั่งที่ใช้บ่อย

```
npm install   # ติดตั้ง dependencies
npm run dev   # รันเซิร์ฟเวอร์
npm run build # สร้างไฟล์สำหรับ production
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js

```
npm create vue@latest
```

เมื่อเราสร้างโปรเจกต์ Vue.js โดยใช้คำสั่ง

Vue จะสร้างโครงสร้างไฟล์ที่เป็นมาตรฐานและพร้อมใช้งาน

📌 7. vite.config.js – การตั้งค่า Vite (Build Tool สำหรับ Vue 3)

ตัวอย่างการตั้งค่าพื้นฐานของ vite.config.js

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'

export default defineConfig({
  plugins: [vue()]
})
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

โครงสร้างโปรเจกต์ Vue.js 🏗️

🔷 **สรุป**

- ✅ public/ – เก็บไฟล์ที่ต้องเข้าถึงโดยตรง
- ✅ src/ – เก็บไฟล์โค้ด Vue
- ✅ components/ – เก็บ Vue Components
- ✅ views/ – หน้าเว็บแต่ละหน้า
- ✅ router/ – (ถ้ามี) ไฟล์กำหนด Route
- ✅ package.json – ข้อมูล dependencies และ script
- ✅ vite.config.js – ตั้งค่า Vite

💡 Vue มีโครงสร้างที่ยืดหยุ่นและสามารถปรับเปลี่ยนได้ 🔥

ใช้ npm สำหรับการพัฒนาแอปจริง

3. พื้นฐาน Vue.js

3. พื้นฐาน Vue.js

- ✅ การสร้าง Vue Instance (createApp)
- ✅ Data Binding ({{ message }})
- ✅ Methods & Event Handling (@click)
- ✅ Computed Properties & Watchers
- ✅ Class และ Style Binding (:class, :style)

3. พื้นฐาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

การสร้าง Vue Instance (createApp)

พื้นฐานสำคัญของ Vue.js ที่ควรรู้ไว้ก่อนลงลึกไปยังคอมโพเนนต์หรือการจัดการสถานะอื่น ๆ

✅ 1. การสร้าง Vue Instance (createApp)

👉 ใช้ Vue.createApp({...}).mount('#app') เพื่อเริ่มต้น Vue และผูกกับ HTML

```
<div id="app">
  <p>{{ message }}</p>
</div>

<script>
  const app = Vue.createApp({
    data() {
      return {
        message: 'สวัสดี Vue!'
      };
    }
  });

  app.mount('#app');
</script>
```

3. พื้นฐาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

การสร้าง Vue Instance (createApp)

```
<div id="app">
  <p>{{ message }}</p>
</div>
```

✅ 2. Data Binding ({{ message }})

- {{ message }} เรียกว่า Mustache Syntax
- ดึงค่าจาก data() มาแสดงบนหน้าเว็บ
- ใช้ได้กับข้อความ, number, array ฯลฯ

✅ 3. Methods & Event Handling (@click)

- @click คือ shorthand ของ v-on:click
- ใช้เพื่อเรียก method เวลากดปุ่ม

```
<div id="app">
  <button @click="sayHi">กดฉันสิ</button>
</div>
<script>
  const app = Vue.createApp({
    methods: {
      sayHi() {
        alert('สวัสดี!');
      }
    }
  });
  app.mount('#app');
</script>
```

3. พื้นฐาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

การสร้าง Vue Instance (createApp)

```
<div id="app">
  <input v-model="firstName">
  <input v-model="lastName">
  <p>ชื่อเต็ม: {{ fullName }}</p>
</div>
<script>
  const app = Vue.createApp({
    data() {
      return {
        firstName: 'สมชาย',
     lastName: 'ใจดี'
    };
  },
  computed: {
      fullName() {
        return this.firstName + ' ' + this.lastName;
      }
    }
  });
  app.mount('#app');
</script>
```

✅ 4. Computed Properties & Watchers

- Computed

computed จะคำนวณค่าตาม reactive data และ cache ค่าไว้

3. พื้นฐาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

- ✅ 4. Computed Properties & Watchers
  - 🔹 Watchers

ใช้สำหรับดูค่าที่เปลี่ยน แล้ว “ทำอะไรบางอย่าง”
เช่น fetch data, log, validate

การสร้าง Vue Instance (createApp)

```
<script>
  const app = Vue.createApp({
    data() {
      return {
        age: 20
      };
    },
    watch: {
      age(newValue, oldValue) {
        console.log(`อายุเปลี่ยนจาก ${oldValue} เป็น
${newValue}`);
      }
    }
  });

  app.mount('#app');
</script>
```

3. พื้นฐาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

✅ 5. Class และ Style Binding (:class, :style)

- Binding class

การสร้าง Vue Instance (createApp)

```
<div id="app">
  <p :class="{ active: isActive }">ฉันเปลี่ยนสีได้</p>
  <button @click="isActive = !isActive">Toggle Class</button>
</div>
<script>
  const app = Vue.createApp({
    data() {
      return {
        isActive: false
      };
    }
  });
  app.mount('#app');
</script>
<style>
  .active {
    color: red;
    font-weight: bold;
  }
</style>
```

ใช้ npm สำหรับการพัฒนาแอปจริง

การสร้าง Vue Instance (createApp)

✅ 5. Class และ Style Binding (:class, :style)

- Binding style

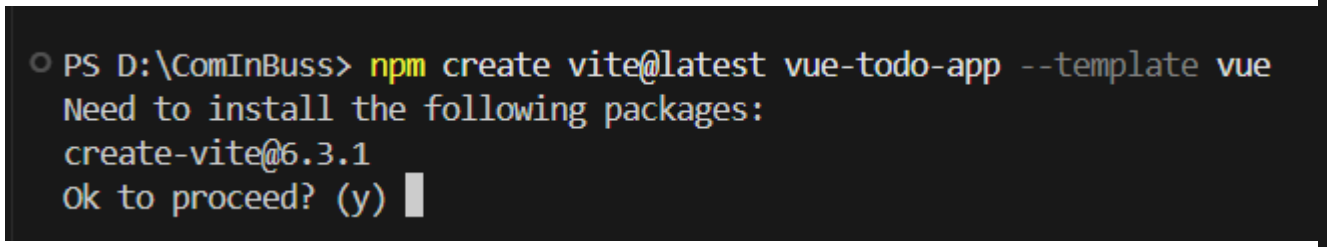
```
<p :style="{ color: textColor, fontSize: fontSize + 'px' }">
  ตัวอย่าง style binding
</p>
```

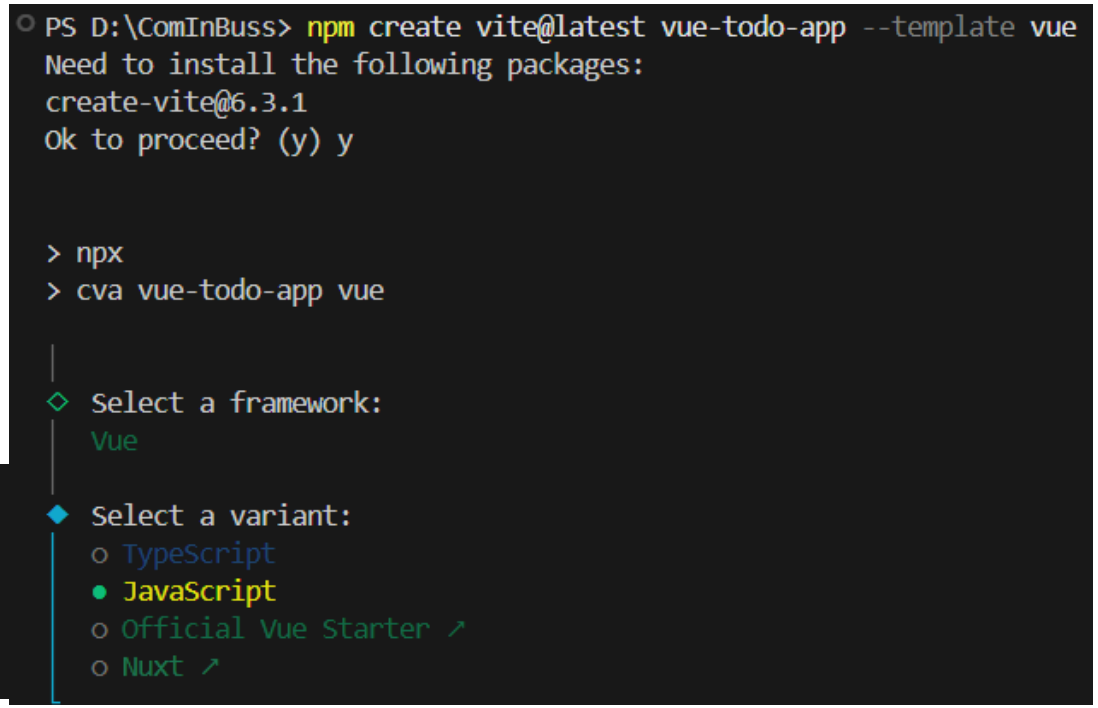
2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

3 การติดตั้ง Vue.js วิธีที่ 3: แบบ Vite

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก
🔧 ขั้นตอนเตรียมโปรเจกต์:

```
npm create vite@latest vue-todo-app --template vue
cd vue-todo-app
npm install
npm run dev
```





2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก
🔧 ขั้นตอนเตรียมโปรเจกต์:

```
npm create vite@latest vue-todo-app --template vue
cd vue-todo-app
npm install
npm run dev
```

```
PS D:\ComInBuss> npm create vite@latest vue-todo-app --template vue
Need to install the following packages:
create-vite@6.3.1
Ok to proceed? (y)
```

```
PS D:\ComInBuss> npm create vite@latest vue-todo-app --template vue
Need to install the following packages:
create-vite@6.3.1
Ok to proceed? (y) y

> npx
> cva vue-todo-app vue

◇ Select a framework:
  Vue

◆ Select a variant:
  ○ TypeScript
  ● JavaScript
  ○ Official Vue Starter ↗
  ○ Nuxt ↗
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก
🔧 ขั้นตอนเตรียมโปรเจกต์:

```
npm create vite@latest vue-todo-app --template vue
cd vue-todo-app
npm install
npm run dev
```

สรุปขั้นตอนตอนนี้:
1.เลือก Vue ✅
2.เลือก JavaScript ✅
3.กด Enter ✅

```
PS D:\ComInBuss> npm create vite@latest vue-todo-app --template vue
Need to install the following packages:
create-vite@6.3.1
Ok to proceed? (y) y


> npx
> cva vue-todo-app vue


◇  Select a framework:
│  Vue
│
◆  Select a variant:
│  ○ TypeScript
│  ● JavaScript
│  ○ Official Vue Starter ↗
│  ○ Nuxt ↗
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini

Vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก
🔧 ขั้นตอนเตรียมโปรเจกต์:

```
npm create vite@latest vue-todo-app --template vue
cd vue-todo-app
npm install
npm run dev
```

สรุปขั้นตอนตอนนี้:
1.เลือก Vue ✅
2.เลือก JavaScript ✅
3.กด Enter ✅

```
PS D:\ComInBuss> npm create vite@latest vue-todo-app --template vue
Need to install the following packages:
create-vite@6.3.1
Ok to proceed? (y) y


> npx
> cva vue-todo-app vue

│
◇  Select a framework:
│  Vue
│
◇  Select a variant:
│  JavaScript
│
◇  Scaffolding project in D:\ComInBuss\vue-todo-app...
│
└  Done. Now run:

  cd vue-todo-app
  npm install
  npm run dev

PS D:\ComInBuss>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก
🔧 ขั้นตอนเตรียมโปรเจกต์:

✅ ขั้นตอนหลังจาก npm create vite@latest vue-todo-app --template vue

1. เมื่อสร้างเสร็จแล้ว จะได้ข้อความประมาณนี้:

```
cd vue-todo-app
npm install
npm run dev

PS D:\ComInBuss>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก
🔧 ขั้นตอนเตรียมโปรเจกต์:

✅ ขั้นตอนหลังจาก npm create vite@latest vue-todo-app --template vue

1. เมื่อสร้างเสร็จแล้ว จะได้ข้อความประมาณนี้:
2. PS D:\WebProgramming\vue-todo-app> npm install

```
  cd vue-todo-app
  npm install
  npm run dev

PS D:\ComInBuss>
```

```
◇  Scaffolding project in D:\ComInBuss\vue-todo-app...
│
└  Done. Now run:

  cd vue-todo-app
  npm install
  npm run dev

PS D:\ComInBuss> cd vue-todo-app
PS D:\ComInBuss\vue-todo-app> npm install

added 30 packages, and audited 31 packages in 8s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\ComInBuss\vue-todo-app>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ **วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก**
🔧 **ขั้นตอนเตรียมโปรเจกต์:**

✅ ขั้นตอนหลังจาก npm create vite@latest vue-todo-app --template vue

1. เมื่อสร้างเสร็จแล้ว จะได้ข้อความประมาณนี้:
2. PS D:\WebProgramming\vue-todo-app> npm install
3. PS D:\WebProgramming\vue-todo-app> npm run dev

```
PS D:\ComInBuss\vue-todo-app> npm run dev

> vue-todo-app@0.0.0 dev
> vite


  VITE v6.2.5  ready in 295 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก

🔧 ขั้นตอนเตรียมโปรเจกต์:

✅ ถัดไป: แก้ไขให้เป็น Todo App

1. เปิดโฟลเดอร์นี้ใน VS Code หรือ editor ที่คุณใช้
2. ไปที่ไฟล์ src/App.vue
3. ลบเนื้อหาทั้งหมด แล้วแทนด้วย โค้ด Todo App ด้านล่างนี้:

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

```
<template>
  <div class="app">
    <h1>📝 Vue Todo App</h1>
    <div class="input-area">
      <input
        v-model="newTodo"
        @keyup.enter="addTodo"
        placeholder="เพิ่มรายการใหม่..."
      />
      <button @click="addTodo">เพิ่ม</button>
    </div>
    <ul class="todo-list">
      <li v-for="(todo, index) in todos" :key="index">
        <span
          :class="{ done: todo.done }"
          @click="toggleDone(index)"
        >
          {{ todo.text }}
        </span>
        <button @click="removeTodo(index)">ลบ</button>
      </li>
    </ul>
  </div>
</template>
<script setup>
import { ref } from 'vue'
const newTodo = ref('')
const todos = ref([])
function addTodo() {
  if (newTodo.value.trim()) {
    todos.value.push({ text: newTodo.value, done: false })
    newTodo.value = ''
  }
}
function removeTodo(index) {
  todos.value.splice(index, 1)
}
function toggleDone(index) {
  todos.value[index].done = !todos.value[index].done
}
</script>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

```
<style scoped>
.app {
  font-family: Arial;
  padding: 2rem;
  max-width: 600px;
  margin: auto;
}
.input-area input {
  padding: 8px;
  width: 250px;
  margin-right: 10px;
}
button {
  padding: 6px 10px;
}
.done {
  text-decoration: line-through;
  color: gray;
}
.todo-list li {
  margin: 8px 0;
}
</style>
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก

🔧 ขั้นตอนเตรียมโปรเจกต์:

ผลลัพธ์

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก

🔧 ขั้นตอนเตรียมโปรเจกต์:

✅ วิธีเริ่มต้น (แบบ Vite) คืออะไร

การเริ่มต้นโปรเจกต์ด้วย Vite คือการใช้เครื่องมือที่ช่วยให้การพัฒนาเว็บแอปพลิเคชันทำได้เร็วและมีประสิทธิภาพมากขึ้น โดยเฉพาะอย่างยิ่งเมื่อใช้กับ JavaScript หรือ TypeScript นั่นเอง Vite จะช่วยให้การจัดการกับการแสดงผลในเวลาพัฒนา (development server) ทำได้เร็วและไม่ยุ่งยาก

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก
🔧 ขั้นตอนเตรียมโปรเจกต์:

✅ วิธีเริ่มต้น (แบบ Vite) คืออะไร

```
npm create vite@latest vue-todo-app --template vue
```

ขั้นตอนถัดไปจะเป็นการตั้งค่าและเริ่มต้นโปรเจกต์ Vue ของคุณ ด้วยขั้นตอนง่ายๆ ตามนี้:
1. เข้าไปในโฟลเดอร์โปรเจกต์ที่สร้างขึ้น:

```
cd vue-todo-app
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ **วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก**

🔧 **ขั้นตอนเตรียมโปรเจกต์:**

2. ติดตั้ง dependencies:

หลังจากที่เข้าไปในโฟลเดอร์โปรเจกต์แล้ว ให้ติดตั้ง dependencies ที่จำเป็นสำหรับโปรเจกต์:

bash

```
npm install
```

คำสั่งนี้จะทำการดาวน์โหลดและติดตั้ง dependencies ที่โปรเจกต์ Vue ของคุณต้องการ เช่น Vue.js, Vite และอื่นๆ ที่ระบุไว้ในไฟล์ package.json

3. รันโปรเจกต์ในโหมดพัฒนา:

หลังจากติดตั้งเสร็จแล้ว คุณสามารถรันโปรเจกต์ในโหมดพัฒนา (development mode) ด้วยคำสั่ง:

```
npm run dev
```

2. การติดตั้งและเริ่มต้นใช้งาน Vue.js

ใช้ npm สำหรับการพัฒนาแอปจริง

Mini Project: Todo App ด้วย Vue.js แบบ "ติดตั้ง" (ไม่ใช้ CDN)

Vue09_TodoApp.vue

✅ วิธีเริ่มต้น (แบบ Vite) – แนะนำเพราะเบาและเร็วมาก

🔧 ขั้นตอนเตรียมโปรเจกต์:

คำสั่งนี้จะเริ่มต้นการรันเซิร์ฟเวอร์ในโหมดพัฒนา และคุณสามารถเข้าถึงแอปของคุณที่ http://localhost:5173/ (ตามค่าเริ่มต้นของ Vite) ในเบราว์เซอร์

4. เริ่มพัฒนาแอป:
·ตอนนี้คุณสามารถเริ่มพัฒนาแอปได้แล้ว โดยแก้ไขไฟล์ในโฟลเดอร์ src/ เช่น src/App.vue, src/components/, หรือ src/main.js
·ทุกครั้งที่คุณบันทึกการเปลี่ยนแปลงในไฟล์ ระบบจะทำการรีเฟรช (Hot Module Replacement) เพื่อแสดงผลการเปลี่ยนแปลงในทันที

5. สร้างโปรเจกต์สำหรับการผลิต (Production Build):
เมื่อพัฒนาแอปเสร็จและพร้อมที่จะ deploy ไปยัง production, คุณสามารถสร้าง build ที่เหมาะสมสำหรับการใช้งานจริงโดยใช้คำสั่ง:

```
npm run build
```

ไฟล์ที่สร้างขึ้นจะอยู่ในโฟลเดอร์ dist/ ซึ่งคุณสามารถนำไป deploy บนเซิร์ฟเวอร์ต่างๆ เช่น Netlify, Vercel หรืออื่นๆ

ใช้ npm สำหรับการพัฒนาแอปจริง

4. Vue Directives (คำสั่งพื้นฐาน)

4. Vue Directives (คำสั่งพื้นฐาน)

- ✅ v-bind – ผูกค่าให้ attribute
- ✅ v-model – Two-way data binding
- ✅ v-if / v-else / v-show – ควบคุมการแสดงผล
- ✅ v-for – Loop รายการ
- ✅ v-on หรือ @event – การจัดการ event (@click, @keyup)
- ✅ v-html – แสดง HTML ภายใน Vue
- ✅ v-slot – Slot และการใช้งานใน Component

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.1 v-bind – ผูกค่าให้กับ attribute

1. v-bind – ผูกค่าให้กับ attribute
v-bind ใช้เพื่อผูกค่าให้กับ attribute ของ HTML element ใน Vue ทำให้เราสามารถใช้ค่าจากตัวแปรหรือ computed properties มาเชื่อมโยงกับ attribute ของ element นั้นๆ
ตัวอย่าง:

```
<img v-bind:src="imageUrl" alt="Vue logo">
```

ในตัวอย่างนี้, imageUrl จะถูกผูกให้กับ src attribute ของ <img>

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.1 v-bind – ผูกค่าให้กับ attribute

1. v-bind – ผูกค่าให้กับ attribute

v-bind ใช้เพื่อผูกค่าให้กับ attribute ของ HTML element ใน Vue ทำให้เราสามารถใช้ค่าจากตัวแปรหรือ computed properties มาเชื่อมโยงกับ attribute ของ element นั้นๆ

ตัวอย่าง:

✅ 1. v-bind กับ href ของ <a>

```
<template>
  <a v-bind:href="url">ไปยังเว็บไซต์</a>
</template>

<script>
export default {
  data() {
    return {
      url: 'https://www.example.com'
    }
  }
}
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.1 v-bind – ผูกค่าให้กับ attribute

1. v-bind – ผูกค่าให้กับ attribute

v-bind ใช้เพื่อผูกค่าให้กับ attribute ของ HTML element ใน Vue ทำให้เราสามารถใช้ค่าจากตัวแปรหรือ computed properties มาเชื่อมโยงกับ attribute ของ element นั้นๆ

ตัวอย่าง:

✅ 2. v-bind กับ src ของ <img>

```
<template>
  <img v-bind:src="imageSrc" alt="ภาพตัวอย่าง">
</template>

<script>
export default {
  data() {
    return {
      imageSrc: 'https://via.placeholder.com/150'
    }
  }
}
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.1 v-bind – ผูกค่าให้กับ attribute

1. v-bind – ผูกค่าให้กับ attribute

v-bind ใช้เพื่อผูกค่าให้กับ attribute ของ HTML element ใน Vue ทำให้เราสามารถใช้ค่าจากตัวแปรหรือ computed properties มาเชื่อมโยงกับ attribute ของ element นั้นๆ

ตัวอย่าง:

✅ 3. v-bind กับ class แบบ dynamic

```
<template>
  <div v-bind:class="isActive ? 'active' : 'inactive'">
สถานะ</div>
</template>

<script>
export default {
  data() {
    return {
      isActive: true
    }
  }
}
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.1 v-bind – ผูกค่าให้กับ attribute

1. v-bind – ผูกค่าให้กับ attribute

v-bind ใช้เพื่อผูกค่าให้กับ attribute ของ HTML element ใน Vue ทำให้เราสามารถใช้ค่าจากตัวแปรหรือ computed properties มาเชื่อมโยงกับ attribute ของ element นั้นๆ

ตัวอย่าง:

✅ 4. v-bind กับ style

```
<template>
  <div v-bind:style="{ color: textColor, fontSize: fontSize + 'px' }">
    ข้อความที่มีสไตล์
  </div>
</template>
<script>
export default {
  data() {
    return {
      textColor: 'blue',
      fontSize: 18
    }
  }
}
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.1 v-bind – ผูกค่าให้กับ attribute

1. v-bind – ผูกค่าให้กับ attribute

v-bind ใช้เพื่อผูกค่าให้กับ attribute ของ HTML element ใน Vue ทำให้เราสามารถใช้ค่าจากตัวแปรหรือ computed properties มาเชื่อมโยงกับ attribute ของ element นั้นๆ

ตัวอย่าง:

✅ 5. v-bind แบบ shorthand (:) กับ disabled ของ <button>

```
<template>
  <div v-bind:style="{ color: textColor, fontSize: fontSize + 'px' }">
    ข้อความที่มีสไตล์
  </div>
</template>
<script>
export default {
  data() {
    return {
      textColor: 'blue',
      fontSize: 18
    }
  }
}
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

🧩 วิธีรวมในโปรเจกต์เดียวกัน (2 วิธีหลัก):

✅ **วิธีที่ 2: รวมทุกตัวอย่างในไฟล์เดียว (เช่น App.vue)**
ถ้าแค่อยากทดลองหรือฝึกเฉย ๆ ก็สามารถเขียนทั้งหมดใน App.vue ตัวเดียวได้

1 `npm create vite@latest vue-vbind-demo -- --template vue`

vue-vbind-demo = ชื่อโฟลเดอร์โปรเจกต์ (เปลี่ยนได้ตามต้องการ)

- ใช้ --template vue เพื่อเลือก Vue 3 แบบพร้อมใช้งาน

```
PS D:\ComInBuss> npm create vite@latest vue-vbind-demo -- --template vue

> npx
> cva vue-vbind-demo vue

◆  Select a framework:
│  ○ Vanilla
│  ● Vue
│  ○ React
│  ○ Preact
│  ○ Lit
│  ○ Svelte
│  ○ Solid
│  ○ Qwik
│  ○ Angular
│  ○ Others
```

2 เลือก javaScript

```
◇  Select a framework:
│  Vue
│
◆  Select a variant:
│  ○ TypeScript
│  ● JavaScript
│  ○ Official Vue Starter ↗
│  ○ Nuxt ↗
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

วิธีรวม

วิธีรวมในโปรเจกต์เดียวกัน (2 วิธีหลัก):

✅ วิธีที่ 2: รวมทุกตัวอย่างในไฟล์เดียว (เช่น App.vue)
ถ้าแค่อยากทดลองหรือฝึกเฉย ๆ ก็สามารถเขียนทั้งหมดใน App.vue ตัวเดียวได้

2. เข้าไปในโฟลเดอร์โปรเจกต์

3 `cd vue-vbind-demo`

4 `npm install`

5 `npm run dev`

```
cd vue-vbind-demo
npm install
npm run dev

PS D:\ComInBuss> cd vue-vbind-demo
PS D:\ComInBuss\vue-vbind-demo> npm install

added 30 packages, and audited 31 packages in 10s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\ComInBuss\vue-vbind-demo> npm run dev

> vue-vbind-demo@0.0.0 dev
> vite


  VITE v6.2.5  ready in 299 ms

  → Local:   http://localhost:5173/
  → Network: use --host to expose
  → press h + enter to show help
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

🧩 วิธีรวมในโปรเจกต์เดียวกัน (2 วิธีหลัก):

✅ วิธีที่ 1: แสดงแต่ละตัวอย่างในคนละ <component> เหมาะกับโปรเจกต์แบบ SFC (Vue CLI หรือ Vite)

1

```
src/
├── components/
│   ├── BindHref.vue
│   ├── BindImg.vue
│   ├── BindClass.vue
│   ├── BindStyle.vue
│   └── BindDisabled.vue
└── App.vue
```

2 แล้วใน App.vue:

```vue
<template>
  <BindHref />
  <BindImg />
  <BindClass />
  <BindStyle />
  <BindDisabled />
</template>
<script>
import BindHref from './components/BindHref.vue'
import BindImg from './components/BindImg.vue'
import BindClass from
'./components/BindClass.vue'
import BindStyle from
'./components/BindStyle.vue'
import BindDisabled from
'./components/BindDisabled.vue'
export default {
  components: {
    BindHref,
    BindImg,
    BindClass,
    BindStyle,
    BindDisabled
  }
}
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

🧩 วิธีรวมในโปรเจกต์เดียวกัน (2 วิธีหลัก):

✅ **วิธีที่ 2: รวมทุกตัวอย่างในไฟล์เดียว (เช่น App.vue)**
ถ้าแค่อยากทดลองหรือฝึกเฉย ๆ ก็สามารถเขียนทั้งหมดใน App.vue ตัวเดียวได้

5. เปิด src/App.vue แล้วแทนที่เนื้อหาในไฟล์ด้วยตัวอย่างทั้งหมด

6

```
<template>
  <div>
    <!-- ตัวอย่าง 1 -->
    <a :href="url">ไปยังเว็บไซต์</a><br>
    <!-- ตัวอย่าง 2 -->
    <img :src="imageSrc" alt="ภาพตัวอย่าง"><br>
    <!-- ตัวอย่าง 3 -->
    <div :class="isActive ? 'active' : 'inactive'">สถานะ</div><br>
    <!-- ตัวอย่าง 4 -->
    <div :style="{ color: textColor, fontSize: fontSize + 'px' }">ข้อความมีสไตล์
</div><br>
    <!-- ตัวอย่าง 5 -->
    <button :disabled="isDisabled">คลิกไม่ได้</button><br>
  </div>
</template>
```

7

```
<script>
export default {
  data() {
    return {
      url: 'https://www.example.com',
      imageSrc: 'https://via.placeholder.com/150',
      isActive: true,
      textColor: 'blue',
      fontSize: 18,
      isDisabled: true
    }
  }
}
</script>
```

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

2. v-model – Two-way data binding

v-model ใช้สำหรับเชื่อมโยงค่า input ของ form กับตัวแปรใน data ของ Vue ทำให้สามารถทำการปรับค่าของตัวแปรใน data และค่าใน input ได้ทั้งสองทิศทาง

ตัวอย่าง:

```
<input v-model="message" placeholder="Enter a message">
<p>{{ message }}</p>
```

ในตัวอย่างนี้, ค่าใน input จะถูกเก็บไว้ในตัวแปร message และทุกครั้งที่ message เปลี่ยนแปลง ค่าใน input ก็จะเปลี่ยนไปตาม

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

ตัวอย่าง v-model – Two-way data binding แบบติดตั้ง จำนวน 5 ตัวอย่างโดยสร้างให้อยู่ในโปรเจ็กต์เดียวกัน

- ✅ ใช้กับ Vue 3
- ✅ อยู่ใน โปรเจกต์เดียวกัน
- ✅ ใช้ในไฟล์ App.vue ไฟล์เดียว เพื่อความสะดวก
- ✅ เหมาะสำหรับผู้ติดตั้ง Vue แล้วผ่าน Vite หรือ Vue CLI

📁 **โครงสร้างโปรเจกต์**

```
vue-vmodel-demo/
└── src/
    └── App.vue ← ใส่ตัวอย่างทั้งหมดที่นี่
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

ตัวอย่าง v-model – Two-way data binding แบบติดตั้ง จำนวน 5 ตัวอย่างโดยสร้างให้อยู่ในโปรเจ็กต์เดียวกัน

✅ คำสั่งสร้างโปรเจกต์ Vue ด้วย Vite

1. สร้างโปรเจกต์ใหม่

```
npm create vite@latest vue-vmodel-demo -- --template vue
```

- vue-vmodel-demo = ชื่อโฟลเดอร์โปรเจกต์ (เปลี่ยนชื่อได้ตามต้องการ)
- --template vue = ใช้ template Vue 3 แบบเริ่มต้น

2. เข้าไปในโฟลเดอร์โปรเจกต์

```
cd vue-vmodel-demo
```

3. ติดตั้ง dependencies

```
npm install
```

4. เปิดโปรเจกต์ใน VS Code

```
PS D:\ComInBuss> npm create vite@latest vue-vmodel-demo -- --template vue
```

5. แก้ไขไฟล์ src/App.vue แล้วแทนที่ด้วยโค้ด v-model

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

ตัวอย่าง v-model – Two-way data binding แบบติดตั้ง จำนวน 5 ตัวอย่างโดยสร้างให้อยู่ในโปรเจ็กต์เดียวกัน

```
PS D:\ComInBuss> npm create vite@latest vue-vmodel-demo -- --template vue
```

```
PS D:\ComInBuss> npm create vite@latest vue-vmodel-demo -- --template vue

> npx
> cva vue-vmodel-demo vue

◆  Select a framework:
│  ○ Vanilla
│  ● Vue
│  ○ React
│  ○ Preact
│  ○ Lit
│  ○ Svelte
│  ○ Solid
│  ○ Qwik
│  ○ Angular
│  ○ Others
```

```
PS D:\ComInBuss> npm create vite@latest vue-vmodel-demo -- --template vue

> npx
> cva vue-vmodel-demo vue

◇  Select a framework:
│  Vue
│
◆  Select a variant:
│  ○ TypeScript
│  ● JavaScript
│  ○ Official Vue Starter ↗
│  ○ Nuxt ↗
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

ตัวอย่าง v-model – Two-way data binding แบบติดตั้ง จำนวน 5 ตัวอย่างโดยสร้างให้อยู่ในโปรเจ็กต์เดียวกัน

```
PS D:\ComInBuss> npm create vite@latest vue-vmodel-demo -- --template vue

> npx
> cva vue-vmodel-demo vue

◇  Select a framework:
│  Vue
│
◇  Select a variant:
│  JavaScript
│
◇  Scaffolding project in D:\ComInBuss\vue-vmodel-demo...
│
└  Done. Now run:

  cd vue-vmodel-demo
  npm install
  npm run dev

PS D:\ComInBuss>
```

```
  cd vue-vmodel-demo
  npm install
  npm run dev

PS D:\ComInBuss> cd vue-vmodel-demo
PS D:\ComInBuss\vue-vmodel-demo> npm install

added 30 packages, and audited 31 packages in 9s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\ComInBuss\vue-vmodel-demo> npm run dev

> vue-vmodel-demo@0.0.0 dev
> vite

Port 5173 is in use, trying another one...

  VITE v6.2.5  ready in 278 ms

  ➜  Local:   http://localhost:5174/
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

✅ โค้ดใน App.vue – รวมตัวอย่าง v-model ทั้ง 5 แบบ

```
<template>
  <div style="padding: 20px; max-width: 500px; margin: auto; font-family: sans-serif;">
    <h2>v-model – Two-way Data Binding</h2>
    <!-- ตัวอย่าง 1: v-model กับ <input type="text"> -->
    <div>
      <label>ชื่อผู้ใช้:</label>
      <input v-model="username" type="text" />
      <p>คุณพิมพ์: {{ username }}</p>
    </div>
    <!-- ตัวอย่าง 2: v-model กับ <textarea> -->
    <div>
      <label>ข้อความเพิ่มเติม:</label>
      <textarea v-model="message"></textarea>
      <p>ข้อความของคุณ: {{ message }}</p>
    </div>
```

```
    <!-- ตัวอย่าง 3: v-model กับ <input type="checkbox"> -->
    <div>
      <label>
        <input type="checkbox" v-model="isChecked" />
        ยอมรับเงื่อนไข
      </label>
      <p>สถานะ: {{ isChecked ? ' ยอมรับแล้ว' : '❌ ยังไม่ยอมรับ' }}</p>
    </div>
    <!-- ตัวอย่าง 4: v-model กับ <input type="radio"> -->
    <div>
      <label>เพศ:</label>
      <label><input type="radio" value="ชาย" v-model="gender" />
ชาย</label>
      <label><input type="radio" value="หญิง" v-model="gender" />
หญิง</label>
      <p>คุณเลือก: {{ gender }}</p>
    </div>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

✅ โค้ดใน App.vue – รวมตัวอย่าง v-model ทั้ง 5 แบบ

```
<!-- ตัวอย่าง 5: v-model กับ <select> -->
    <div>
      <label>จังหวัด:</label>
      <select v-model="province">
        <option disabled value="">-- เลือกจังหวัด --</option>
        <option>กรุงเทพฯ</option>
        <option>เชียงใหม่</option>
        <option>ขอนแก่น</option>
        <option>ภูเก็ต</option>
      </select>
      <p>จังหวัดที่เลือก: {{ province }}</p>
    </div>
  </div>
</template>
```

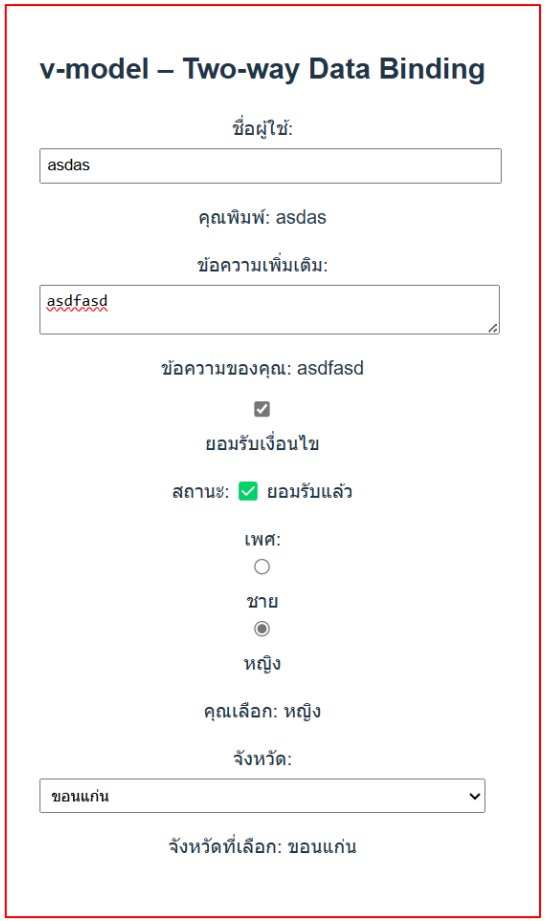
```
<script>
export default {
  data() {
    return {
      username: '',
      message: '',
      isChecked: false,
      gender: '',
      province: ''
    }
  }
}
</script>
<style>
input, textarea, select {
  display: block;
  margin: 5px 0 10px;
  padding: 5px;
  width: 100%;
}
</style>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

✅ โค้ดใน App.vue – รวมตัวอย่าง v-model ทั้ง 5 แบบ



4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.2 v-model – Two-way data binding

ตัวอย่าง v-model – Two-way data binding แบบติดตั้ง จำนวน 5 ตัวอย่างโดยสร้างให้อยู่ในโปรเจ็กต์เดียวกัน

🔧 สรุปตัวอย่าง:

ตัวอย่างที่	ใช้กับ	รูปแบบ
1	<input type="text">	พิมพ์ชื่อ
2	<textarea>	พิมพ์ข้อความ
3	<input type="checkbox">	ติ๊กยอมรับเงื่อนไข
4	<input type="radio">	เลือกเพศ
5	<select>	เลือกจังหวัด

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.3 v-if, v-else, v-show – ควบคุมการแสดงผล

3. v-if, v-else, v-show – ควบคุมการแสดงผล

·v-if ใช้ในการแสดงหรือซ่อน element ตามเงื่อนไข
·v-else ใช้ในการแสดง element อื่นๆ เมื่อเงื่อนไขของ v-if เป็นเท็จ
·v-show ใช้ในการควบคุมการแสดงหรือซ่อนโดยการเปลี่ยนแค่ display ของ element
ตัวอย่าง:

```
<p v-if="isVisible">This text is visible if isVisible is true</p>
<p v-else>This text is visible if isVisible is false</p>
```

หรือการใช้ v-show:

```
<p v-show="isVisible">This text will be shown/hidden based on
isVisible</p>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.3 v-if, v-else, v-show – ควบคุมการแสดงผล

✅ ตัวอย่างที่ 1: แสดงข้อความเมื่อผู้ใช้ล็อกอิน (v-if / v-else)

```
<template>
  <div>
    <p v-if="isLoggedIn">ยินดีต้อนรับกลับมา!</p>
    <p v-else>กรุณาเข้าสู่ระบบก่อน</p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      isLoggedIn: false
    };
  }
};
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.3 v-if, v-else, v-show – ควบคุมการแสดงผล

✅ ตัวอย่างที่ 2: ซ่อน/แสดงเมนูเพิ่มเติม (v-show)

1

```
<template>
  <div>
    <button @click="toggleMenu">แสดง/ซ่อน เมนู</button>
    <div v-show="showMenu">
      <ul>
        <li>หน้าแรก</li>
        <li>เกี่ยวกับ</li>
        <li>ติดต่อ</li>
      </ul>
    </div>
  </div>
</template>
```

2

```
<script>
export default {
  data() {
    return {
      showMenu: false
    };
  },
  methods: {
    toggleMenu() {
      this.showMenu = !this.showMenu;
    }
  }
};
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.3 v-if, v-else, v-show – ควบคุมการแสดงผล

✅ ตัวอย่างที่ 3: แสดงระดับคะแนน (v-if / v-else-if / v-else)

1

```
<template>
  <div>
    <p v-if="score >= 80">เกรด A</p>
    <p v-else-if="score >= 70">เกรด B</p>
    <p v-else-if="score >= 60">เกรด C</p>
    <p v-else>ตก</p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      score: 75
    };
  }
};
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.3 v-if, v-else, v-show – ควบคุมการแสดงผล

✅ ตัวอย่างที่ 4: โหลดข้อมูลแบบ Loading (v-if)

1

```
<template>
  <div>
    <p v-if="isLoading">กำลังโหลดข้อมูล...</p>
    <p v-else>ข้อมูลโหลดเสร็จแล้ว!</p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      isLoading: true
    };
  },
  mounted() {
    setTimeout(() => {
      this.isLoading = false;
    }, 2000);
  }
};
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.3 v-if, v-else, v-show – ควบคุมการแสดงผล

✅ ตัวอย่างที่ 5: ปุ่มซ่อน/แสดงรหัสผ่าน (v-show)

1

```
<template>
  <div>
    <input :type="showPassword ? 'text' : 'password"" placeholder="รหัสผ่าน">
    <button @click="showPassword = !showPassword">
      {{ showPassword ? 'ซ่อน' : 'แสดง' }}
    </button>
  </div>
</template>

<script>
export default {
  data() {
    return {
      showPassword: false
    };
  }
};
</script>
```

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

4. v-for – Loop รายการ
v-for ใช้สำหรับการวนลูปผ่าน array หรือ object และแสดงผลรายการนั้นๆ
ตัวอย่าง:

```
<ul> <li v-for="item in items" :key="item.id">{{ item.name }}</li> </ul>
```

**ในตัวอย่างนี้, items เป็น array ที่จะถูกวนลูปและแสดงผลแต่ละ item ใน **

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

ตัวอย่างการใช้ v-for ใน Vue.js

1. แสดงรายการของชื่อ:

```
<template>
  <ul>
    <li v-for="(name, index) in names" :key="index">{{ name }}</li>
  </ul>
</template>

<script>
export default {
  data() {
    return {
      names: ['John', 'Jane', 'Alice', 'Bob', 'Charlie']
    };
  }
}
</script>
```

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

ตัวอย่างการใช้ v-for ใน Vue.js

2. แสดงรายการของตัวเลข:

```
<template>
  <ul>
    <li v-for="number in numbers" :key="number">{{ number }}</li>
  </ul>
</template>

<script>
export default {
  data() {
    return {
      numbers: [1, 2, 3, 4, 5]
    };
  }
}
</script>
```

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

ตัวอย่างการใช้ v-for ใน Vue.js

3. แสดงรายการของวัตถุ (Object):

```
<template>
  <ul>
    <li v-for="(product, index) in products" :key="index">
      {{ product.name }} - ${{ product.price }}
    </li>
  </ul>
</template>
```

```
<script>
export default {
  data() {
    return {
      products: [
        { name: 'Apple', price: 1.2 },
        { name: 'Banana', price: 0.5 },
        { name: 'Cherry', price: 2.5 },
        { name: 'Date', price: 3.0 },
        { name: 'Elderberry', price: 1.8 }
      ]
    };
  }
}
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

ตัวอย่างการใช้ v-for ใน Vue.js

4. แสดงรายการที่ใช้ v-for กับคำนวณ (เช่น การสร้างเลข 1-5):

```
<template>
  <ul>
    <li v-for="(number, index) in computedNumbers"
:key="index">{{ number }}</li>
  </ul>
</template>

<script>
export default {
  computed: {
    computedNumbers() {
      return [1, 2, 3, 4, 5];
    }
  }
}
</script>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

ตัวอย่างการใช้ v-for ใน Vue.js

5.แสดงรายการพร้อมการเพิ่มข้อมูลใหม่:

html

```
<template>
  <div>
    <input v-model="newItem" @keyup.enter="addItem"
placeholder="Add item"/>
    <ul>
      <li v-for="(item, index) in items" :key="index">{{ item
}}</li>
    </ul>
  </div>
</template>
```

```
<script>
export default {
  data() {
    return {
      newItem: '',
      items: []
    };
  },
  methods: {
    addItem() {
      if (this.newItem) {
        this.items.push(this.newItem);
        this.newItem = '';
      }
    }
  }
}
</script>
```

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

ตัวอย่างการใช้ v-for ใน Vue.js

5.แสดงรายการพร้อมการเพิ่มข้อมูลใหม่: html

```
PS D:\ComInBuss> npm create vite@latest vue-for-app --template vue

> npx
> cva vue-for-app vue

◆ Select a framework:
  ○ Vanilla
  ● Vue
  ○ React
  ○ Preact
  ○ Lit
  ○ Svelte
  ○ Solid
  ○ Qwik
  ○ Angular
  ○ Others
```

```
PS D:\ComInBuss> npm create vite@latest vue-for-app --template vue

> npx
> cva vue-for-app vue

◇ Select a framework:
  Vue

◆ Select a variant:
  ○ TypeScript
  ● JavaScript
  ○ Official Vue Starter ↗
  ○ Nuxt ↗
```

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

ตัวอย่างการใช้ v-for ใน Vue.js

5.แสดงรายการพร้อมการเพิ่มข้อมูลใหม่:

html

```
◇  Select a framework:
   Vue

◇  Select a variant:
   JavaScript

◇  Scaffolding project in D:\ComInBuss\vue-for-app...

   Done. Now run:

  cd vue-for-app
  npm install
  npm run dev
```

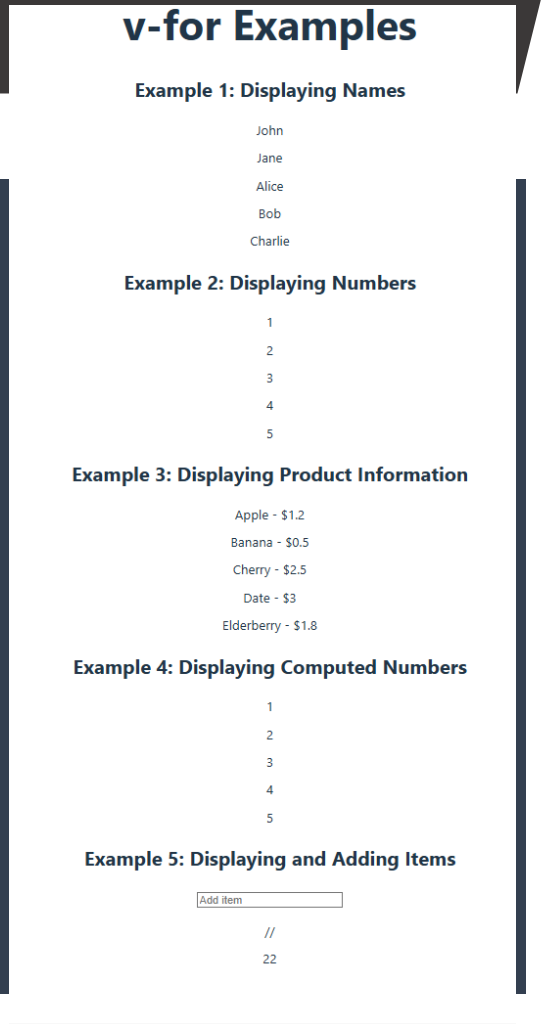
4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

App.vue

```
<template>
  <div class="container">
    <h1>v-for Examples</h1>
    <!-- Example 1: Displaying Names -->
    <h2>Example 1: Displaying Names</h2>
    <ul>
      <li v-for="(name, index) in names" :key="index">{{ name }}</li>
    </ul>
    <!-- Example 2: Displaying Numbers -->
    <h2>Example 2: Displaying Numbers</h2>
    <ul>
      <li v-for="number in numbers" :key="number">{{ number }}</li>
    </ul>
    <!-- Example 3: Displaying Product Information -->
    <h2>Example 3: Displaying Product Information</h2>
```



4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

App.vue

```
    <ul>
      <li v-for="(product, index) in products" :key="index">
        {{ product.name }} - ${{ product.price }}
      </li>
    </ul>
    <!-- Example 4: Displaying Computed Numbers -->
    <h2>Example 4: Displaying Computed Numbers</h2>
    <ul>
      <li v-for="(number, index) in computedNumbers" :key="index">{{ number }}</li>
    </ul>
    <!-- Example 5: Displaying and Adding Items -->
    <h2>Example 5: Displaying and Adding Items</h2>
    <input v-model="newItem" @keyup.enter="addItem" placeholder="Add item"/>
    <ul>
      <li v-for="(item, index) in items" :key="index">{{ item }}</li>
    </ul>
  </div>
</template>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

App.vue

```
<script>
export default {
  data() {
    return {
      // Example 1: List of Names
      names: ['John', 'Jane', 'Alice', 'Bob', 'Charlie'],
      // Example 2: List of Numbers
      numbers: [1, 2, 3, 4, 5],
      // Example 3: List of Products
      products: [
        { name: 'Apple', price: 1.2 },
        { name: 'Banana', price: 0.5 },
        { name: 'Cherry', price: 2.5 },
        { name: 'Date', price: 3.0 },
        { name: 'Elderberry', price: 1.8 }
      ],
      // Example 4: Computed Numbers
      newItem: '',
      items: [],
    };
  },
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.4 v-for – Loop รายการ

App.vue

```
  computed: {
    // Example 4: Generate Numbers 1 to 5
    computedNumbers() {
      return [1, 2, 3, 4, 5];
    }
  },
  methods: {
    // Example 5: Add new item to the list
    addItem() {
      if (this.newItem) {
        this.items.push(this.newItem);
        this.newItem = '';
      }
    }
  }
}
</script>
```

App.vue

```
<style scoped>
.container {
  max-width: 800px;
  margin: 0 auto;
  padding: 20px;
}

ul {
  list-style-type: none;
  padding-left: 0;
}

li {
  margin-bottom: 10px;
}
</style>
```

4. Vue Directives (คำสั่งพื้นฐาน)

ใช้ npm สำหรับการพัฒนาแอปจริง

4.5 v-on หรือ @event – การจัดการ event

5. v-on หรือ @event – การจัดการ event
v-on ใช้ในการฟัง event เช่น click, keyup, และสามารถใช้สัญลักษณ์ย่อ @ ได้เช่นกัน
ตัวอย่าง:

```
<button v-on:click="handleClick">Click Me</button>
```

หรือใช้ @click แทน v-on:click:

```
<button @click="handleClick">Click Me</button>
```

ในตัวอย่างนี้, เมื่อคลิกที่ปุ่มจะเรียกฟังก์ชัน handleClick

ใช้ npm สำหรับการพัฒนาแอปจริง

4.6 v-html – แสดง HTML ภายใน Vue

6. v-html – แสดง HTML ภายใน Vue
v-html ใช้เพื่อแสดง HTML ที่อยู่ใน string โดยไม่ต้อง escape หรือทำการ render ใหม่
ตัวอย่าง:

```
<div v-html="rawHtml"></div>
```

ในตัวอย่างนี้, rawHtml จะต้องเป็น string ที่ประกอบด้วย HTML markup ที่คุณต้องการให้แสดงผล

ใช้ npm สำหรับการพัฒนาแอปจริง

4.7 v-slot – Slot และการใช้งานใน Component

7. v-slot – Slot และการใช้งานใน Component
v-slot ใช้สำหรับการส่ง content ไปยัง slot ของ component โดย slot เป็นพื้นที่ที่กำหนดให้ component สามารถแสดงเนื้อหาที่ถูกส่งมาได้
ตัวอย่าง:

```
<template v-slot:header>
        <h1>Welcome to My Site</h1>
</template>
<MyComponent>
        <template v-slot:header>
                <h1>My Custom Header</h1>
        </template>
</MyComponent>
```

ในตัวอย่างนี้, slot header ถูกกำหนดให้มีเนื้อหาของ <h1> ที่ส่งมาใน MyComponent

ใช้ npm สำหรับการพัฒนาแอปจริง

4. Vue Directives (คำสั่งพื้นฐาน)

สรุป

ทุก directive ที่กล่าวมานี้ช่วยให้การพัฒนา Vue app เป็นไปอย่างสะดวกและง่ายขึ้น โดยไม่ต้องเขียนโค้ดเยอะ แค่ใช้คำสั่งพื้นฐานเหล่านี้ก็ช่วยจัดการการแสดงผล, การควบคุม event, และการผูกข้อมูลได้ง่ายๆ

PWA คืออะไร

Progressive Web Applications (PWA)

Progressive Web Applications (PWA) คือ เว็บแอปพลิเคชันที่สามารถทำงานได้เหมือนแอปพลิเคชันบนมือถือ โดยใช้เทคโนโลยีเว็บ เช่น HTML, CSS และ JavaScript แต่มีฟีเจอร์ที่ทำให้มันคล้ายกับแอปเนทีฟ เช่น การทำงานแบบออฟไลน์, การแจ้งเตือนแบบพุช (Push Notifications), และการติดตั้งลงในอุปกรณ์โดยไม่ต้องผ่าน App Store หรือ Play Store

Agenda

คุณสมบัติหลักของ PWA

คุณสมบัติหลักของ PWA

1. Responsive – ทำงานได้บนทุกอุปกรณ์ (มือถือ, แท็บเล็ต, คอมพิวเตอร์)
2. Offline Support – ใช้ Service Workers เพื่อให้แอปทำงานได้แม้ไม่มีอินเทอร์เน็ต
3. Installable – ผู้ใช้สามารถติดตั้งแอปลงในอุปกรณ์โดยไม่ต้องผ่าน Store
4. Push Notifications – ส่งการแจ้งเตือนให้ผู้ใช้ได้เหมือนแอปมือถือ
5. Fast & Secure – โหลดเร็ว และใช้ HTTPS เพื่อความปลอดภัย
6. Auto Update – อัปเดตอัตโนมัติเมื่อมีการเปลี่ยนแปลง

Agenda

ข้อดีของ PWA

ข้อดีของ PWA

- ✅ ไม่ต้องติดตั้งผ่าน App Store
- ✅ โหลดเร็วและทำงานได้แม้ไม่มีอินเทอร์เน็ต
- ✅ ใช้ทรัพยากรเครื่องน้อยกว่าการพัฒนา Native App
- ✅ ปรับปรุงและอัปเดตแอปได้ง่าย
- 🔹 ตัวอย่าง PWA ยอดนิยม

•Twitter Lite – โหลดเร็วกว่าแอปปกติ

•Pinterest – PWA ทำให้ engagement เพิ่มขึ้น 60%

•Starbucks – รองรับการสั่งกาแฟแม้ไม่มีอินเทอร์เน็ต

PWA เป็นเทคโนโลยีที่เหมาะกับธุรกิจที่ต้องการให้ผู้ใช้เข้าถึงแอปได้ง่าย โดยไม่ต้องดาวน์โหลดจาก Store 🎉

Agenda

ทดสอบ PWA และติดตั้งลงเครื่อง

5. ทดสอบ PWA และติดตั้งลงเครื่อง
    1. เปิดเว็บแอป http://127.0.0.1:5500/ ใน Google Chrome
    2. ไปที่ DevTools (F12) → Application → Manifest
    3. จะเห็นปุ่ม "Install App" หรือสามารถติดตั้ง PWA บนอุปกรณ์ได้โดยคลิกที่ไอคอนติดตั้งบนแถบ URL

การใช้งาน Live Server

Live Server คืออะไร

Live Server คือเครื่องมือที่ช่วยให้คุณสามารถ แสดงผลเว็บไซต์แบบเรียลไทม์ (real-time) ได้ทันทีขณะที่คุณกำลังเขียนโค้ด โดยไม่ต้องรีเฟรชหน้าเว็บด้วยตนเองทุกครั้งที่มีการเปลี่ยนแปลงไฟล์

ความสามารถของ Live Server:

- ✅ เปิดไฟล์ HTML แล้วแสดงในเว็บเบราว์เซอร์ทันที
- ✅ เมื่อมีการแก้ไขไฟล์ (เช่น HTML, CSS, JS) ตัวเว็บจะ รีเฟรชอัตโนมัติ
- ✅ ใช้งานง่าย แค่คลิกขวาที่ไฟล์ HTML แล้วเลือก "Open with Live Server"

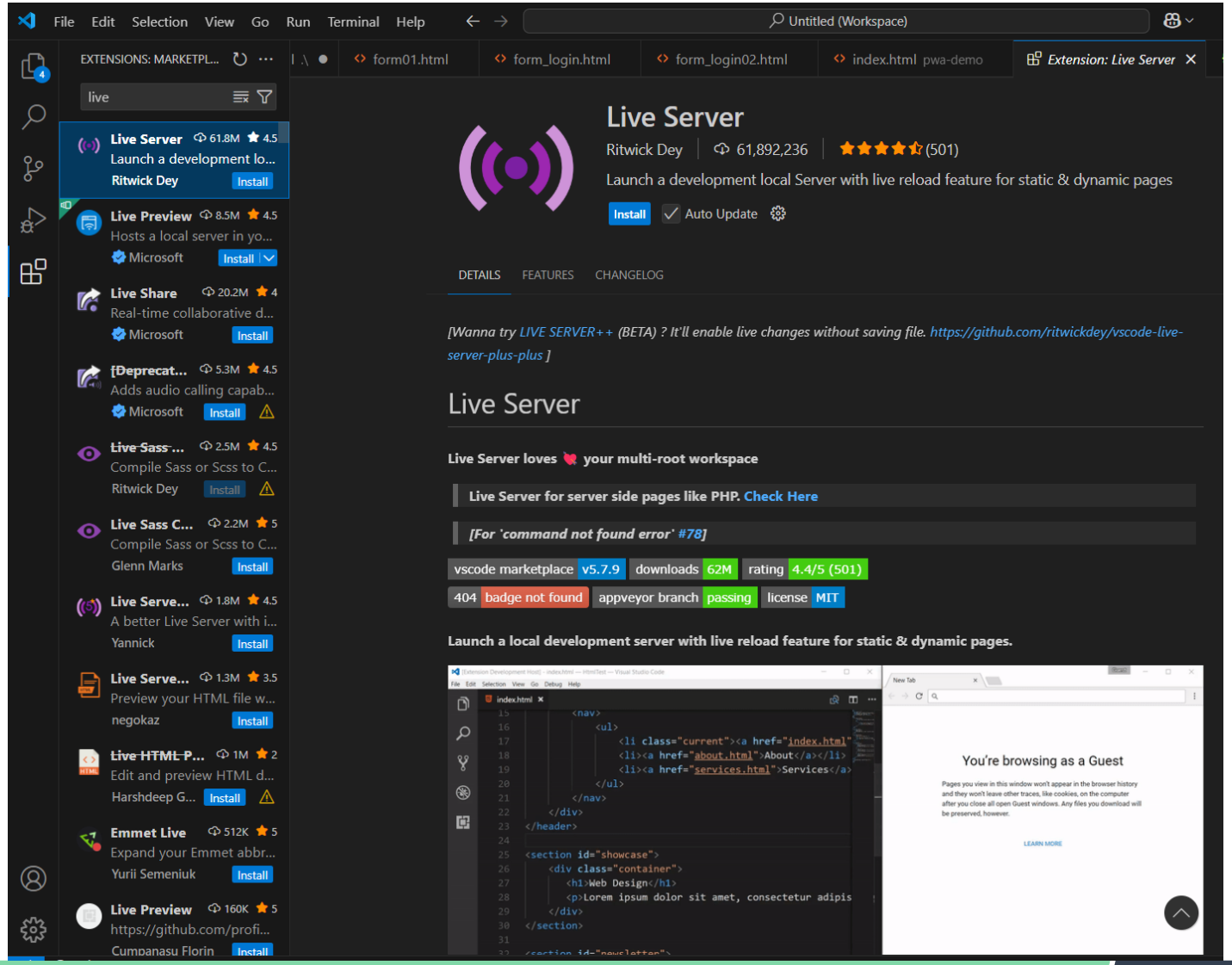
✅ วิธีแก้ไขปัญหา Live Server ไม่แสดงใน VS Code

- 1. ตรวจสอบว่าได้ติดตั้ง Live Server แล้วหรือยัง
    1. เปิด VS Code
    2. ไปที่ Extensions (Ctrl + Shift + X)
    3. พิมพ์ "Live Server" ในช่องค้นหา
    4. ถ้ายัง ไม่ได้ติดตั้ง → กด Install
    5. รีสตาร์ท VS Code แล้วลองใหม่

การใช้งาน Live Server

บทที่ 5

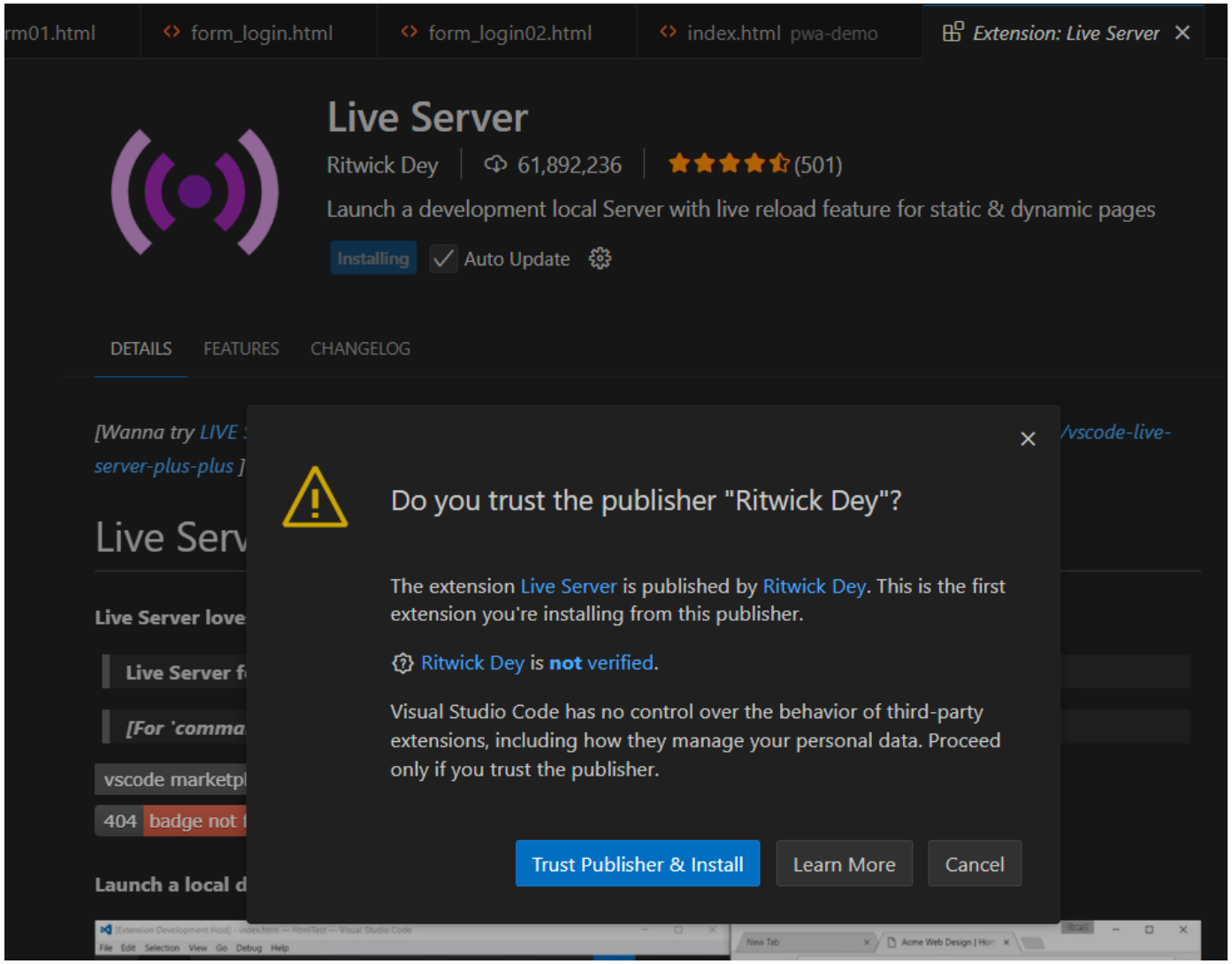
การติดตั้ง Live Server



การใช้งาน Live Server

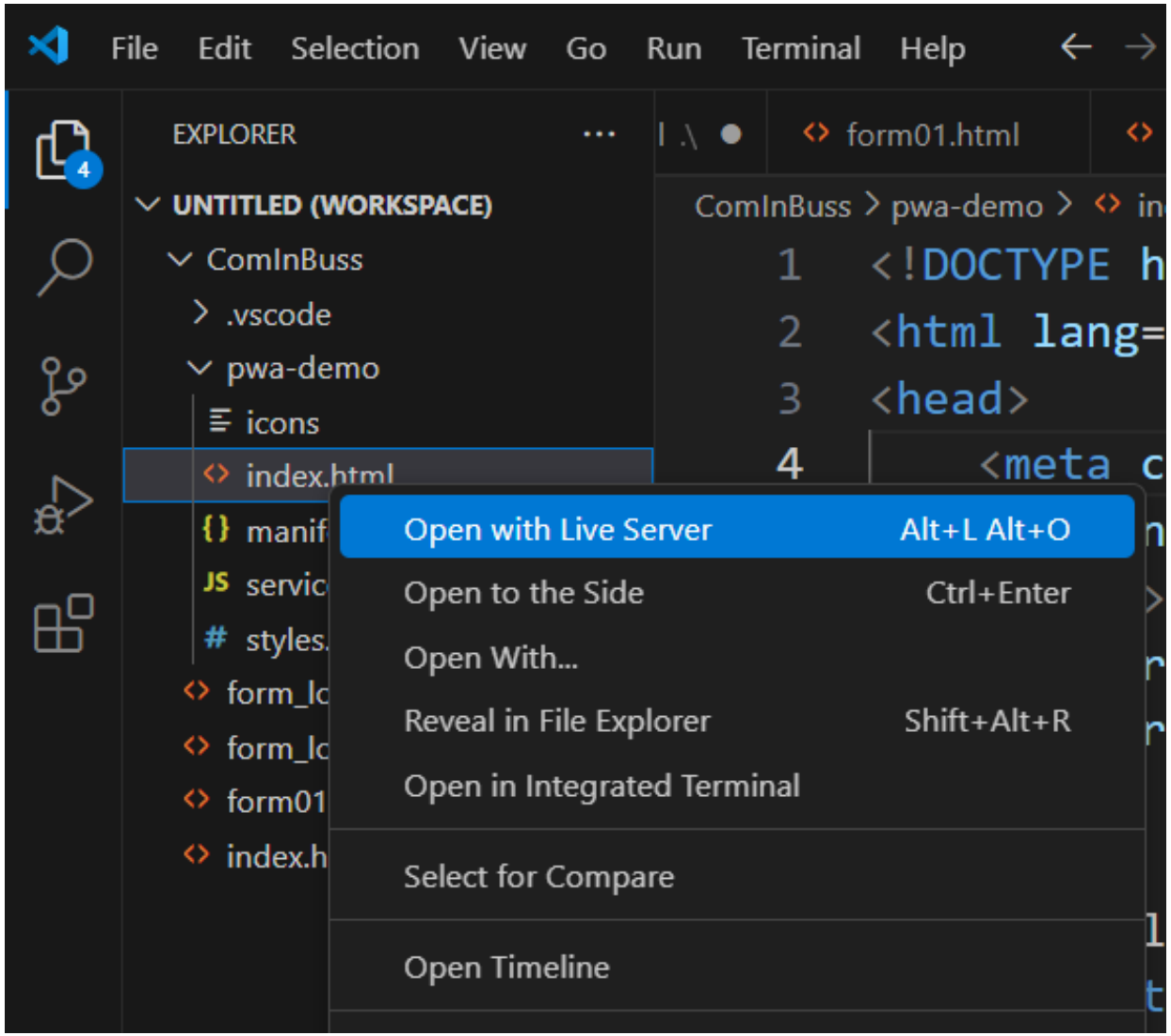
บทที่ 5

การติดตั้ง Live Server



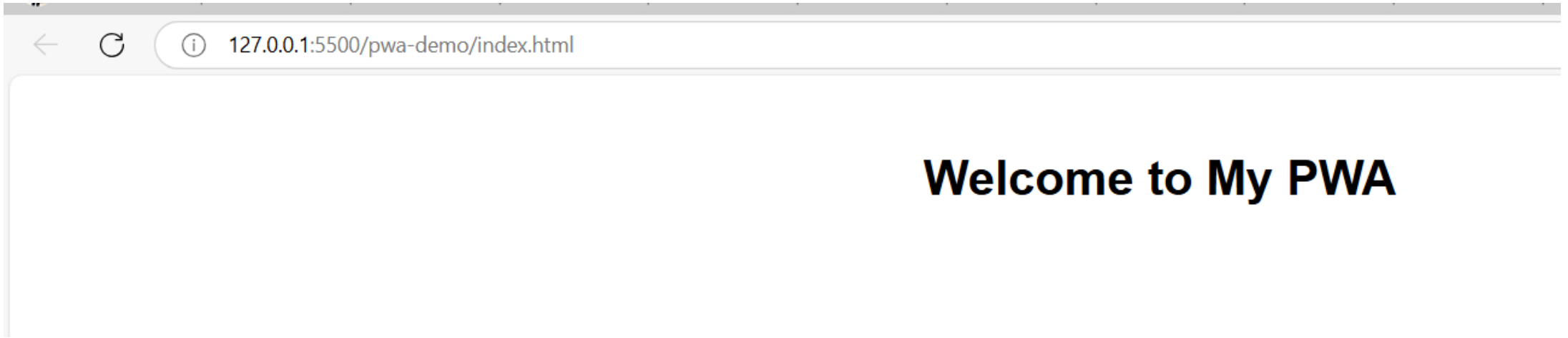
การใช้งาน Live Server

การติดตั้ง Live Server



การใช้งาน Live Server

การติดตั้ง Live Server



อ้างอิง

- https://monnapablog.wordpress.com/บทที่-6-การใช้คอมพิวเตอร์/
- Vue Crash Course" (Traversy Media, 2019) แนะนำการพัฒนาเว็บแอปพลิเคชันด้วย Vue.js สำหรับผู้เริ่มต้น

วิดีโอ
- Traversy Media. (2019, August 20). *Vue crash course* [Video]. YouTube. https://www.youtube.com/watch?v=qZXt1Aom3Cs