DAISYUI WEB PROGRAMMING

ADVANCE

(Integrative-Generative AI Edition)

DaisyUI + React *1

DaisyUI + Vue / Nuxt *77

DaisyUI + Backend Integration *141

Theme Switching (Dynamic) 218

Accessibility & Best Practices 3321

Bibliggattry 371

AUTHOR: STUDENT PRICE BOOK CENTER

คำนำ

โลกของการพัฒนาเว็บสมัยใหม่มีการเปลี่ยนแปลงอย่างรวดเร็ว ความต้องการของผู้ใช้เพิ่มขึ้นในด้าน **ประสบการณ์ใช้งานที่สวยงาม โต้ตอบได้ง่าย และเข้าถึงได้ทุกกลุ่ม** นักพัฒนาจึงต้องเรียนรู้เครื่องมือและเทคนิคที่ช่วยให้การสร้างเว็บแอปมีประสิทธิภาพและรวดเร็วมากขึ้น หนึ่งในเครื่องมือที่โดดเด่นคือ **DaisyUI** ซึ่งเป็น plugin ของ Tailwind CSS ที่ช่วยให้สามารถสร้าง UI ที่สวยงาม ใช้งานง่าย และปรับแต่งได้อย่างรวดเร็ว

หนังสือเล่มนี้มุ่งเน้นการสอน **การใช้ DaisyUI ในระดับ Advance** ครอบคลุมการทำงานร่วมกับเฟรมเวิร์กยอดนิยม เช่น **React, Vue, Nuxt** พร้อมการผสานเข้ากับ **backend integration** การจัดการ **dynamic theme** และแนวทาง **best practices ด้าน accessibility** ซึ่งทั้งหมดนี้ช่วยให้เว็บแอปของคุณมีคุณภาพสูงทั้งด้าน UI, UX และความเสถียร

ใน **บทที่ 11: DaisyUI + React** ผู้อ่านจะได้เรียนรู้ตั้งแต่การติดตั้ง DaisyUI ในโปรเจกต์ React ทั้ง **Next.js, CRA, และ Vite** ไปจนถึงการใช้ **component ของ DaisyUI ใน JSX** พร้อมทั้งการจัดการ **state ของ React** และตัวอย่างโครงการแบบบูรณาการที่ช่วยให้สามารถนำไปใช้งานจริงได้

ต่อด้วย **บทที่ 12: DaisyUI + Vue / Nuxt** ซึ่งจะเจาะลึกการใช้ DaisyUI กับ Vue 3 และ Nuxt ครอบคลุมการติดตั้ง การใช้ component ใน **Single File Component (SFC)** การจัดการ **reactive state** และตัวอย่างโปรเจกต์บูรณาการ ทำให้ผู้อ่านเข้าใจวิธีสร้างเว็บแอป Vue ที่มี UI สวยงามและตอบสนองต่อผู้ใช้ได้อย่างครบถ้วน

บทที่ 13: DaisyUI + Backend Integration จะพาผู้อ่านเรียนรู้การเชื่อมต่อเว็บแอปกับ backend จริง ผ่านการสร้างฟอร์ม **Login และ Register** การจัดการ **loading และ error state** และการแสดงผลข้อมูลด้วย **Table พร้อม Pagination** นอกจากนี้ยังมีตัวอย่าง Ultimate Dashboard Project ที่รวมทุกองค์ประกอบของ DaisyUI และ backend integration ให้เห็นภาพรวมการสร้างเว็บแอปแบบมืออาชีพ

ใน **บทที่ 14: Theme Switching (Dynamic)** ผู้อ่านจะได้เรียนรู้การทำ **dynamic theme switching** โดยใช้ JavaScript เพื่อเปลี่ยนธีมของเว็บแบบ runtime พร้อมการเก็บค่าธีมที่ผู้ใช้เลือกใน **LocalStorage** และการสร้าง **theme switcher button** ทั้งยังมีตัวอย่างโครงการ Ultimate และ Super Ultimate ที่ผสานการสลับธีมเข้ากับเว็บแอปจริง

สุดท้าย **บทที่ 15: Accessibility & Best Practices** จะเน้นแนวทางการสร้างเว็บแอปที่ **เข้าถึงได้ทุกคน (a11y-friendly)** ครอบคลุมการทำ **keyboard navigation, screen reader compatibility** และการปรับ **contrast กับ readable color** รวมทั้งตัวอย่างโครงการ Ultimate และ Super Ultimate Dashboard Project ที่ช่วยให้ผู้อ่านเห็นภาพรวมการพัฒนาเว็บแอปที่สวยงาม ใช้งานง่าย และเข้าถึงได้จริง

หนังสือเล่มนี้ออกแบบมาเพื่อให้ผู้อ่านสามารถ **เรียนรู้ทั้งพื้นฐานและแนวทางเชิงลึก** ของ DaisyUI ในการพัฒนาเว็บแอประดับมืออาชีพ ไม่ว่าคุณจะเป็นนักพัฒนาที่เริ่มใช้งาน DaisyUI หรือผู้ที่ต้องการยกระดับเว็บแอปให้มีคุณภาพสูง หนังสือเล่มนี้จะเป็นคู่มือที่ครบถ้วนและนำไปปฏิบัติได้จริง

ขอให้ผู้อ่านใช้หนังสือเล่มนี้เป็นแรงบันดาลใจในการสร้างเว็บแอปที่ **สวยงาม, ตอบสนองผู้ใช้, และมีประสิทธิภาพสูง** พร้อมทั้งเรียนรู้แนวทางปฏิบัติที่ดีที่สุดในโลกของการพัฒนาเว็บสมัยใหม่

ด้วยรักและปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 11 DaisyUI + React 1

- DaisyUI + React
- DaisyUI + React (Next.js / Vite / CRA
- ติดตั้ง DaisyUI ใน Next.js / CRA / Vite + React
- การใช้ Component DaisyUI เป็น JSX
- การจัดการ state ใน React กับ DaisyUI
- ตัวอย่างบูรณาการ
- Ultimate Examples

บทที่ 12 DaisyUI + Vue / Nuxt 77

- DaisyUI + Vue / Nuxt
- DaisyUI + Vue / Nuxt (เชิงลึก)
- การติดตั้ง DaisyUI บน Vue
- การใช้ DaisyUI component ใน Vue SFC (Single File Component)
- โปรแกรมบูรณาการ
- การจัดการ reactive state ใน Vue 3
- Ultimate Examples

บทที่ 13 DaisyUI + Backend Integration 141

- DaisyUI + Backend Integration
- DaisyUI + Backend Integration แบบเชิงลึก
- การทำฟอร์ม Login / Register เชื่อม API จริง
- ตัวอย่างแนวประยุกต์
- การจัดการ Loading + Error state ด้วย DaisyUI
- Ultimate Dashboard Project
- รายละเอียดเชิงลึกการแสดงผล Data ด้วย Table + Pagination
- ตัวอย่างบูรณาการ
- ตัวอย่าง Ultimate

บทที่ 14 Theme Switching (Dynamic).....218
- Theme Switching (Dynamic)
- Theme Switching (Dynamic) — Deep Dive
- การสลับธีม runtime ด้วย JavaScript
- การเก็บธีมที่เลือกใน LocalStorage
- การสร้าง Theme Switcher Button แบบพื้นฐาน
- ตัวอย่างบูรณาการ
- Ultimate
- Super Ultimate

บทที่ 15 Accessibility & Best Practices 312
- Accessibility & Best Practices
- Accessibility & Best Practices (เชิงลึก)
- การทำให้ DaisyUI a11y-friendly (Keyboard Navigation)
- Screen Reader Compatibility
- ปรับปรุง Contrast และ Readable Color
- ตัวอย่างบูรณาการ
- Ultimate
- Super Ultimate Dashboard Project
- ตัวอย่างบูรณาการ
- React Super Ultimate

บรรณานุกรม 371

บทที่ 11

DaisyUI + React

(DaisyUI + React)

เนื้อหา

- DaisyUI + React
- DaisyUI + React (Next.js / Vite / CRA
- ติดตั้ง DaisyUI ใน Next.js / CRA / Vite + React
- การใช้ Component DaisyUI เป็น JSX
- การจัดการ state ใน React กับ DaisyUI
- ตัวอย่างบูรณาการ
- Ultimate Examples

บทนำ บทที่ 11: DaisyUI + React

ในยุคปัจจุบัน การพัฒนาเว็บแอปด้วย React เป็นหนึ่งในเทคโนโลยีหลักที่ได้รับความนิยมอย่างมาก แต่การสร้าง UI ที่สวยงามและตอบสนองต่อผู้ใช้ได้ดีนั้นยังคงเป็นความท้าทายสำหรับนักพัฒนา ด้วยเหตุนี้ **DaisyUI** ซึ่งเป็น plugin ของ Tailwind CSS จึงเข้ามาช่วยให้การสร้างส่วนประกอบ UI ของเว็บเป็นเรื่องง่ายและรวดเร็วขึ้น

บทนี้จะพาผู้อ่านตั้งแต่ขั้นตอนการติดตั้ง DaisyUI ในโปรเจกต์ React ทั้งสามรูปแบบหลัก ได้แก่ **Next.js, Create React App (CRA), และ Vite** จากนั้นเราจะเรียนรู้การใช้งาน **component ของ DaisyUI ในรูปแบบ JSX** ซึ่งเป็นพื้นฐานสำคัญสำหรับการนำ UI components มาใช้งานใน React

นอกจากนี้บทนี้ยังครอบคลุมการ **จัดการ state** ของ UI components เช่น การเปิด-ปิด modal, dropdown, และองค์ประกอบ interactive อื่น ๆ ผ่าน React state และ event handling เพื่อให้สามารถสร้างเว็บแอปที่ตอบสนองต่อผู้ใช้ได้อย่างเป็นธรรมชาติ

ท้ายที่สุดแล้ว หลังจากศึกษาบทนี้ ผู้อ่านจะมีความเข้าใจทั้งด้านการติดตั้ง การใช้งาน component ของ DaisyUI ใน React และการจัดการ interaction ของ UI ซึ่งเป็นพื้นฐานสำคัญในการพัฒนาเว็บแอปสมัยใหม่ที่สวยงามและใช้งานง่าย

DaisyUI + React

- ติดตั้ง DaisyUI ใน Next.js / CRA / Vite + React
- การใช้ component DaisyUI เป็น JSX
- การจัดการ state (เปิดปิด modal, dropdown)

ยอด—มาดู **บทที่ 11: DaisyUI + React** แบบละเอียดสุด ๆ (รวมคำสั่งติดตั้งแบบทีละขั้นตอน, ตัวอย่างโค้ด JSX/TSX, แนวทางควบคุม state สำหรับ modal/dropdown, ข้อควรระวัง + accessibility/SSR tips) — ผมจะแบ่งเป็นหัวข้อย่อยเพื่อให้ตามได้ง่ายครับ □

บทนำสั้น ๆ

DaisyUI คือชุดคอมโพเนนต์ที่ต่อยอดจาก Tailwind CSS — ในโปรเจกต์ React (Next.js / Vite / CRA) เราจะติดตั้ง Tailwind ก่อน แล้วต่อด้วย daisyUI เป็น plugin หรือใช้ @plugin "daisyui" ในไฟล์ CSS แล้วเรียก class ของ daisyUI ใน JSX ได้เลย (docs แนะนำวิธีติดตั้งแบบ framework-specific). (daisyui.com)

---

1) การติดตั้ง (Step-by-step) — Next.js / Vite / Create React App (CRA)

หลักการเดียวกัน: ติดตั้ง Tailwind ⟶ สร้าง/แก้ไฟล์ CSS ให้มี directives ของ Tailwind ⟶ ติดตั้ง daisyUI ⟶ เพิ่ม daisyUI เป็น plugin / หรือ @plugin ในไฟล์ CSS ⟶ ตั้ง content ใน tailwind.config ให้ครอบคลุมไฟล์ React ของคุณ

A. Next.js (แนะนำ Next.js app-router / TypeScript ตัวอย่าง)

1. สร้างโปรเจกต์:

```
npx create-next-app@latest my-app
cd my-app
# ถ้า create-next-app ให้เลือก Tailwind ให้เลือก (หรือรันขั้นตอน Tailwind ด้านล่าง)
```

2. ถ้ายังไม่มี Tailwind:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

3. ติดตั้ง daisyUI:

```
npm i -D daisyui@latest
```

4. ตัวอย่าง tailwind.config.ts (Next + TS):

```
import type { Config } from 'tailwindcss'
import daisyui from 'daisyui'

const config: Config = {
  content: [
```

```
    './app/**/*.{js,ts,jsx,tsx,mdx}',
    './pages/**/*.{js,ts,jsx,tsx,mdx}',
    './components/**/*.{js,ts,jsx,tsx,mdx}',
  ],
  theme: { extend: {} },
  plugins: [daisyui],
}
export default config
```

5. ในไฟล์ CSS (เช่น app/globals.css) ให้มี:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* ถ้าชอบแบบ @plugin ก็ใช้ได้ (daisyUI docs) */
@import "tailwindcss";
@plugin "daisyui";
```

6. รัน npm run dev — ใช้ class เช่น <button className="btn btn-primary">Hi</button> ได้เลย. (daisyui.com)

B. Vite + React (แนะนำสำหรับ dev experience เร็ว)

1. สร้าง Vite React:

```
npm create vite@latest my-vite-app -- --template react
cd my-vite-app
```

2. ติดตั้ง Tailwind + daisyUI (ตาม daisyUI guide for Vite):

```
npm install tailwindcss@latest @tailwindcss/vite@latest daisyui@latest
```

3. ตัวอย่าง vite.config.js (เพิ่ม plugin ของ tailwind):

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

export default defineConfig({
  plugins: [tailwindcss(), react()],
})
```

4. ใน src/index.css / src/style.css:

```
@import "tailwindcss";
@plugin "daisyui";
```

5. เพิ่ม tailwind.config.js (content paths) ⟶ เริ่มเขียน JSX ได้ทันที. (daisyui.com)

C. Create React App (CRA) — ถ้าจำเป็น (แต่ Tailwind แนะนำ Vite/Next แทน)

- Tailwind มี guide สำหรับ CRA (แต่ CRA อาจมีข้อจำกัดเรื่อง PostCSS หรือเวอร์ชัน) — ถ้าใช้ CRA ให้ทำตาม Tailwind's CRA guide (init postcss, npx tailwindcss init -p, เช็ก content path ./src/**/*.{js,jsx,ts,tsx}). ถ้าเป็นไปได้ ให้ย้ายไป Vite/Next เพื่อ UX ที่ดีกว่า. (v3.tailwindcss.com)

---

2) การใช้ DaisyUI component เป็น JSX (หลักการและตัวอย่าง)

DaisyUI เป็น CSS class-based library — ใน React คุณใช้ className ตามปกติ เช่น btn, card, modal ฯลฯ (docs บางหน้าใช้ $$btn ในตัวอย่างเป็น shorthand ในเอกสาร — แต่ **ในโค้ดจริงใช้ btn**).

ตัวอย่างปุ่ม & Card (JSX)

```
export default function Sample() {
  return (
    <div className="p-6">
      <button className="btn btn-primary">Primary</button>

      <div className="card w-96 bg-base-100 shadow-xl mt-6">
        <div className="card-body">
          <h2 className="card-title">Card Title</h2>
          <p>Some content</p>
          <div className="card-actions justify-end">
            <button className="btn btn-sm">Action</button>
          </div>
        </div>
      </div>
    </div>
  )
}
```

หมายเหตุสำคัญ

- ใช้ className เช่นปกติ

- หลาย component (modal, dropdown) มี *หลายวิธี* ให้เปิด/ปิด — docs อธิบาย method ต่าง ๆ (dialog, checkbox, anchor, CSS focus, popover API). จะแนะนำวิธีที่เหมาะกับ React ต่อไป. (daisyui.com)

---

3) การจัดการ State (เปิด/ปิด modal, dropdown) — Patterns + ตัวอย่างโค้ดจริง

ผมจะแสดง 3 pattern ที่ใช้บ่อยใน React (และข้อดี/ข้อจำกัดของแต่ละแบบ)

A — Modal: แนะนำ (preferred) — ใช้ HTML <dialog> + ref (accessible)

ข้อดี: เข้าถึงได้ดี (ESC ทำงาน), semantic, ไม่ต้องจัด class ซับซ้อน

ข้อควรระวัง: ต้องเรียก .showModal() / .close() จาก client (ไม่บน server), ดังนั้นใน Next.js ให้ใช้ 'use client' หรือเรียกใน useEffect / event handler เท่านั้น

ตัวอย่าง (React, TSX)

```
'use client' // ถ้า Next.js app router

import { useRef } from 'react'

export default function DialogModal() {
  const dialogRef = useRef<HTMLDialogElement | null>(null)

  const open = () => dialogRef.current?.showModal()
  const close = () => dialogRef.current?.close()

  return (
    <>
      <button className="btn" onClick={open}>Open modal</button>

      <dialog ref={dialogRef} className="modal">
        <div className="modal-box">
          <h3 className="font-bold text-lg">Modal Title</h3>
          <p className="py-4">Content here</p>
          <div className="modal-action">
            <button className="btn" onClick={close}>Close</button>
          </div>
        </div>
```

```
      </dialog>
    </>
  )
}
```

- ใน Next.js/SSR: ให้แน่ใจว่าโค้ดที่เรียก showModal() รันฝั่ง client เท่านั้น (use client หรือ ตรวจ typeof window !== 'undefined'). (daisyui.com)

B — Modal: ใช้ state + modal-open class (โปรแกรมเมอร์คุ้นเคย)

แนวคิด: toggle modal-open class บนองค์ประกอบ .modal เพื่อเปิด/ปิด

ตัวอย่าง (JSX)

```
import { useState } from 'react'

export default function ControlledModal() {
  const [open, setOpen] = useState(false)
  return (
    <>
      <button className="btn" onClick={() => setOpen(true)}>Open</button>

      <div className={`modal ${open ? 'modal-open' : ''}`} role="dialog" aria-modal={open}>
        <div className="modal-box">
          <h3 className="font-bold text-lg">Title</h3>
          <p className="py-4">Some text</p>
          <div className="modal-action">
            <button onClick={() => setOpen(false)} className="btn">Close</button>
          </div>
        </div>
      </div>
    </>
  )
}
```

ข้อดี: สมเหตุสมผลเมื่อคุณต้องการควบคุม state ด้วย React

ข้อควรระวัง: ตรวจ accessibility (focus trap/ESC handling) เองถ้าต้องการ behavior ครบถ้วน (ทางเลือก: ใช้ dialog method ถ้าต้องการ built-in accessibility) (daisyui.com)

C — Dropdown: 3 วิธี (ตาม docs) — ตัวอย่างวิธีที่ใช้ใน React

DaisyUI docs อธิบาย 3 วิธี: details/summary, **popover API (ใหม่)**, CSS focus และยังมี dropdown-open class เพื่อบังคับสถานะ

C.1 — Controlled dropdown (toggle class dropdown-open)

```
import { useState, useRef, useEffect } from 'react'

export default function ControlledDropdown() {
  const [open, setOpen] = useState(false)
  const ref = useRef(null)

  useEffect(() => {
    function onDocClick(e) {
      if (ref.current && !ref.current.contains(e.target)) setOpen(false)
    }
    document.addEventListener('mousedown', onDocClick)
    return () => document.removeEventListener('mousedown', onDocClick)
  }, [])

  return (
    <div ref={ref} className={`dropdown ${open ? 'dropdown-open' : ''}`}>
      <button className="btn" onClick={() => setOpen(v => !v)}>Menu</button>
      <ul className="dropdown-content menu p-2 shadow bg-base-100 rounded-box w-52">
        <li><a>Item 1</a></li>
        <li><a>Item 2</a></li>
      </ul>
    </div>
  )
}
```

ข้อดี: ควบคุมด้วย React ได้เต็มที่ (ปิดด้วย outside click, ESC)

ข้อควรระวัง: เพิ่ม event listeners ให้แยกออก ทำ keyboard handling (Escape) เพิ่มเติมถ้าจำเป็น. (daisyui.com)

C.2 — details / summary native element (ง่าย + semantic)

```
<details className="dropdown">
  <summary className="btn">Open</summary>
```

```
  <ul className="dropdown-content menu p-2 shadow bg-base-100 rounded-box w-52">
    <li><a>Item 1</a></li>
  </ul>
</details>
```

ข้อดี: ไม่ต้องจัด state เอง (native open attribute) — แต่ถ้าต้องการควบคุมจาก React ให้แก้ attribute open ผ่าน ref/props. (daisyui.com)

C.3 — Popover API (modern, top-layer positioning)

- ใช้ attributes popovertarget / popover / anchor positioning — docs แสดงตัวอย่าง TSX (popoverTarget, positionAnchor) สำหรับ React. ให้พิจารณาเป็น progressive enhancement (บางเบราว์เซอร์อาจยังไม่รองรับ fully). (daisyui.com)

---

4) Accessibility & SSR best practices (สำคัญ)

- **Modal** — prefer <dialog> (recommended by daisyUI) เพราะรองรับ ESC, semantic และสามารถใช้ .showModal() / .close() ได้. ถ้าใช้ modal-open ต้องจัดการ focus trapping / keyboard closing เอง. (daisyui.com)
- **Dropdown** — details มีพฤติกรรม accessible พื้นฐาน; ถ้าทำ controlled dropdown ให้เพิ่ม aria-expanded, keyboard navigation (Arrow keys/ESC) และ focus management. (daisyui.com)
- **SSR (Next.js)** — หลีกเลี่ยงเรียก document/window ใน render phase; ทำงาน DOM ops ใน event handlers หรือ useEffect หรือใช้ไฟล์ client component ('use client') ใน app-router. (ถ้าใช้ .showModal() ให้แน่ใจว่า DOM พร้อม)
- **Tailwind content paths** — ถ้า class ของ daisyUI ไม่ขึ้น ให้ตรวจ content ใน tailwind.config ว่าครอบคลุมไฟล์ JSX/TSX/MDX/Pages/Components ทั้งหมด

---

5) ข้อปัญหายอดนิยม (และวิธีแก้)

- **Dropdown ถูกตัด (overflow/z-index)** ⟶ ใช้ popover API หรือย้าย dropdown ออกนอก container ที่ overflow หรือจัด z-index/position ให้ถูก. (daisyui.com)
- **Class ของ daisyUI ไม่ทำงาน** ⟶ มักเพราะยังไม่ได้เพิ่ม plugin หรือ tailwind.config.content ไม่ครอบคลุมไฟล์ React. ตรวจ plugins: [require('daisyui')] (หรือ import daisyui ใน ESM) และ @tailwind directives ใน CSS. (daisyui.com)
- **Compatibility issues (Tailwind/DaisyUI versions)** ⟶ ตรวจเวอร์ชัน (docs มี release notes/version links) ถ้าใช้ Tailwind เวอร์ชันใหม่มาก อาจต้องเลือกเวอร์ชัน daisyUI ที่รองรับ. (เช็ก docs/CHANGELOG). (daisyui.com)

---

6) Quick reference — โค้ดสั้น ๆ ที่ใช้บ่อย

tailwind.config.js minimal

```
module.exports = {
  content: ['./src/**/*.{js,jsx,ts,tsx}', './app/**/*.{ts,tsx,js,jsx}'],
  theme: { extend: {} },
  plugins: [require('daisyui')],
}
```

เปิด modal ด้วย dialog (JSX quick)

```
<button onClick={()=>document.getElementById('m').showModal()}
className="btn">Open</button>
<dialog id="m" className="modal">
  <div className="modal-box">...</div>
</dialog>
```

ปิดด้วย state (class toggle)

```
<div className={`modal ${open ? 'modal-open' : ''}`}>...</div>
```

dropdown controlled

```
<div className={`dropdown ${open ? 'dropdown-open' : ''}`}>
  <button onClick={()=>setOpen(v=>!v)} className="btn">Menu</button>
  <ul className="dropdown-content menu">...</ul>
</div>
```

(ใช้ตัวอย่างเต็มด้านบนเมื่อทำงานจริง)

---

7) แหล่งอ้างอิงสำคัญ (อ่านเพิ่ม / official docs)

- daisyUI — Install / How to use (general). (daisyui.com)
- daisyUI — How to install & Tailwind in Next.js (step by step). (daisyui.com)
- daisyUI — Vite install guide + examples. (daisyui.com)
- daisyUI — Modal component docs (dialog, checkbox, modal-open, accessibility notes). (daisyui.com)
- daisyUI — Dropdown component docs (details/summary, popover API, CSS focus, dropdown-open). (daisyui.com)

DaisyUI + React (Next.js / Vite / CRA

---

1) ติดตั้ง & ตั้งค่า (ละเอียด — Next.js / Vite / CRA)

แพ็กเกจที่ต้องติดตั้ง (พื้นฐาน):

```
# Tailwind + daisyUI (ตัวอย่าง npm)
npm install -D tailwindcss postcss autoprefixer
npm i -D daisyui@latest
# (สำหรับ modal accessible) optional
npm i focus-trap-react
# สำหรับจัดการ class แบบสะดวก
npm i classnames
```

การติดตั้งและวิธีใช้งาน daisyUI เป็น Tailwind plugin อธิบายไว้ในคู่มืออย่างเป็นทางการ — ติดตั้งเป็น Node package แล้วเพิ่มเป็น plugin / หรือใช้ @plugin "daisyui" ในไฟล์ CSS. (daisyui.com)

ตัวอย่างไฟล์สำคัญ (tailwind.config.js):

```
// tailwind.config.js
module.exports = {
  content: [
    "./app/**/*.{js,ts,jsx,tsx,mdx}", // Next app dir
    "./pages/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}",
    "./src/**/*.{js,ts,jsx,tsx}"       // Vite/CRA
  ],
  theme: { extend: {} },
  plugins: [require("daisyui")],
}
```

ตรวจ content ให้ครอบคลุมไฟล์ React/TSX ของคุณ — ถ้าคลาสถูกสร้างแบบไดนามิก ให้ใช้ safelist ใน Tailwind config (ไม่ให้ถูก purge). (tailwindcss.com)

ไฟล์ CSS (ตัวอย่าง src/index.css หรือ app/globals.css)

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* แบบปกติ */
@import "tailwindcss";
@plugin "daisyui";

/* หรือ: กำหนด config ของ daisyUI ในไฟล์ CSS ได้ (ดูหัวข้อ Theming ข้างล่าง) */
```

```
@plugin "daisyui" {
  /* custom config here */
}
```

(ดูวิธีและทางเลือกการติดตั้ง React-specific guide ของ daisyUI). (daisyui.com)

---

2) แนวคิดสำคัญ: DaisyUI เป็น CSS classes — React ใช้ className

- ทุกคอมโพเนนต์ daisyUI ถูกใช้งานโดยการใส่ className="btn btn-primary", className="modal", className="dropdown" ฯลฯ — ไม่มี JavaScript เฉพาะจาก daisyUI ที่ต้องเรียก (มันเป็น Tailwind plugin ที่ผลิต CSS).
- หากต้องการ "สถานะ" (เปิด/ปิด) ให้ทำผ่าน **(a)** DOM API (<dialog>.showModal()), **(b)** controlled state + modal-open / dropdown-open class, หรือ **(c)** native elements (<details>/<summary>). เอกสารอธิบายวิธีต่าง ๆ. (daisyui.com)

---

3) Modal — เชิงลึก (3 แบบ + ตัวอย่าง production-ready)

daisyUI แนะนำ 3 วิธีหลักสำหรับ modal: **HTML <dialog> (แนะนำ)**, **checkbox/anchor trick**, **toggle class modal-open** — แต่ใน React/Next คุณมักอยากได้ behaviour ที่ accessible, focus-trapped, และ SSR-safe ⟶ ผมแสดง **(A) dialog + showModal**, และ **(B) controlled portal + focus-trap** (ทั้งสองเวอร์ชันมีข้อดีข้อเสีย) — อ้างอิง: daisyUI modal docs. (daisyui.com)

A — Recommended: <dialog> + showModal (ง่าย + native accessibility)

ข้อดี: native ESC support, semantic. **ข้อจำกัด:** ต้องเรียก DOM API (ต้องรันฝั่ง client) — ใน Next.js ต้องเป็น Client Component ("use client"). (daisyui.com)

ตัวอย่าง (Next.js / React, TypeScript):

```
// components/DialogModal.tsx
'use client' // Next.js app-router: this component uses browser APIs
import { useEffect, useRef } from 'react'

type Props = {
  open: boolean
  onClose: () => void
  title?: string
  children?: React.ReactNode
}
```

```
export default function DialogModal({ open, onClose, title, children }: Props) {
  const dialogRef = useRef<HTMLDialogElement | null>(null)

  useEffect(() => {
    const d = dialogRef.current
    if (!d) return
    function onCloseEvent() { onClose() } // when user closes with Esc or dialog.close()
    d.addEventListener('close', onCloseEvent)
    return () => d.removeEventListener('close', onCloseEvent)
  }, [onClose])

  useEffect(() => {
    const d = dialogRef.current
    if (!d) return
    if (open) {
      if (typeof d.showModal === 'function') d.showModal()
      else d.setAttribute('open', '')
    } else {
      if (d.hasAttribute('open')) d.close()
    }
  }, [open])

  return (
    <dialog ref={dialogRef} className="modal">
      <form method="dialog" className="modal-box">
        {title && <h3 className="font-bold text-lg">{title}</h3>}
        <div className="py-4">{children}</div>
        <div className="modal-action">
          <button className="btn" onClick={() => { dialogRef.current?.close(); onClose();
}}>Close</button>
        </div>
      </form>
    </dialog>
  )
```

```
}
```

ข้อควรระวัง: ถ้าใช้ Next.js app-router อย่าลืม use client เพราะ .showModal() ต้องรันฝั่ง client. (nextjs.org)

B — Production-ready Controlled Modal (Portal + Focus Trap + Restore focus)

เมื่อต้องการ: full control, animation, portal (avoid overflow/z-index issues), และต้องการ focus trapping (recommended for accessibility). ใช้ focus-trap-react เพื่อทำ focus trap อย่างปลอดภัย. (library ref). (GitHub)

ตัวอย่าง (TypeScript + React):

```
// components/PortalModal.tsx
'use client'
import { useEffect, useRef } from 'react'
import { createPortal } from 'react-dom'
import FocusTrap from 'focus-trap-react'
import clsx from 'classnames'

type Props = {
  open: boolean
  onClose: () => void
  children: React.ReactNode
  title?: string
}

export default function PortalModal({ open, onClose, children, title }: Props) {
  const rootRef = useRef<HTMLElement | null>(null)
  const triggerRef = useRef<HTMLElement | null>(null) // for restore focus
  useEffect(() => { rootRef.current = document.body }, [])

  useEffect(() => {
    if (open) {
      // save active element
      triggerRef.current = document.activeElement as HTMLElement | null
      document.body.style.overflow = 'hidden' // lock scroll
    } else {
```

```
      document.body.style.overflow = ''
      // restore focus
      triggerRef.current?.focus?.()
    }
  }, [open])

  useEffect(() => {
    function onKey(e: KeyboardEvent) {
      if (e.key === 'Escape') onClose()
    }
    if (open) document.addEventListener('keydown', onKey)
    return () => document.removeEventListener('keydown', onKey)
  }, [open, onClose])

  if (!rootRef.current) return null
  return createPortal(
    open ? (
      <div className="modal modal-open" role="dialog" aria-modal="true" aria-
labelledby="modal-title">
        <FocusTrap>
          <div className="modal-box">
            {title && <h3 id="modal-title" className="font-bold text-lg">{title}</h3>}
            <div className="py-4">{children}</div>
            <div className="modal-action">
              <button className="btn" onClick={onClose}>Close</button>
            </div>
          </div>
        </FocusTrap>
        {/* backdrop click */}
        <div className="fixed inset-0" onClick={onClose} aria-hidden />
      </div>
    ) : null,
    rootRef.current
  )
```

```
}
```

ประเด็นสำคัญที่โค้ดแก้ให้:

- ป้องกัน scroll ของ body ขณะเปิด modal
- restore focus ให้ trigger element หลังปิด
- trap focus ภายใน modal ด้วย focus-trap-react (ดู docs ของ lib). (focus-trap.github.io)

---

4) Dropdown — เชิงลึก (วิธี + keyboard nav + outside click)

daisyUI ระบุ 3 วิธีหลัก: details/summary (native), popover API (ใหม่), CSS focus และ dropdown-open class (manual control). เลือกวิธีตามความต้องการของ accessibility และ control ที่ต้องการ. (daisyui.com)

A — details/summary (ง่าย + semantic)

```
<details className="dropdown">
  <summary className="btn">Menu</summary>
  <ul className="dropdown-content menu p-2 shadow bg-base-100 rounded-box w-52">
    <li><a href="#">Item 1</a></li>
    <li><a href="#">Item 2</a></li>
  </ul>
</details>
```

- ข้อดี: browser handles toggle; keyboard focus basic behaviors present.
- ข้อจำกัด: การ customize keyboard navigation (arrow keys) ต้องเพิ่มโค้ดเอง.

B — Controlled dropdown (dropdown-open) + keyboard navigation (ตัวอย่าง)

```
// ControlledDropdown.tsx
import { useEffect, useRef, useState } from 'react'

export default function ControlledDropdown() {
  const [open, setOpen] = useState(false)
  const ref = useRef<HTMLDivElement | null>(null)
  const itemsRef = useRef<Array<HTMLAnchorElement | null>>([])

  useEffect(() => {
    function onDoc(e: MouseEvent) {
      if (ref.current && !ref.current.contains(e.target as Node)) setOpen(false)
    }
    function onKey(e: KeyboardEvent) {
```

```
      if (!open) return
      if (e.key === 'Escape') setOpen(false)
      // Arrow nav
      const focused = document.activeElement
      const idx = itemsRef.current.findIndex(el => el === focused)
      if (e.key === 'ArrowDown') {
        e.preventDefault()
        const next = itemsRef.current[(idx + 1) % itemsRef.current.length]
        next?.focus()
      }
      if (e.key === 'ArrowUp') {
        e.preventDefault()
        const prev = itemsRef.current[(idx - 1 + itemsRef.current.length) %
itemsRef.current.length]
        prev?.focus()
      }
    }
    document.addEventListener('mousedown', onDoc)
    document.addEventListener('keydown', onKey)
    return () => {
      document.removeEventListener('mousedown', onDoc)
      document.removeEventListener('keydown', onKey)
    }
  }, [open])

  return (
    <div ref={ref} className={`dropdown ${open ? 'dropdown-open' : ''}`}>
      <button className="btn" onClick={() => setOpen(v => !v)}>Open</button>
      <ul className="dropdown-content menu p-2 shadow bg-base-100 rounded-box w-52"
role="menu">
        {['One','Two','Three'].map((t,i) => (
          <li key={t}><a ref={el=>itemsRef.current[i]=el} role="menuitem" tabIndex={-1}
href="#">{t}</a></li>
        ))}
```

```
      </ul>
    </div>
  )
}
```

จุดสำคัญ: add role="menu" / role="menuitem", manage tabIndex and arrow keys, close on outside click & Escape.

C — Popover API (daisyUI): ใช้เมื่อต้อง positioning ชั้นบน / anchor positioning — docs อธิบาย API ใหม่. (daisyui.com)

---

5) Theming & Customization (daisyUI themes)

daisyUI มาพร้อมธีมสำเร็จรูปหลายธีม และอนุญาตให้ **สร้างธีมของเราเอง** ผ่าน @plugin "daisyui" { themes: [...] } หรือผ่าน tailwind.config.js plugin—รายละเอียดอยู่ใน docs. ตัวอย่าง config และการสลับธีม: (daisyui.com)

ตัวอย่างการเพิ่ม custom theme ใน CSS:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@plugin "daisyui" {
  themes: [
    {
      mytheme: {
        "primary": "#1FB2A6",
        "secondary": "#F000B8",
        "accent": "#37CDBE",
        "neutral": "#3D4451",
        "base-100": "#FFFFFF"
      }
    },
    "dark" // include builtin dark theme
  ],
  logs: false
}
```

สลับธีมจาก React: (data-theme attribute)

```
function ThemeToggle() {
  function setTheme(name: string) {
    document.documentElement.setAttribute('data-theme', name)
  }
  return (
    <div>
      <button className="btn" onClick={() => setTheme('mytheme')}>MyTheme</button>
      <button className="btn" onClick={() => setTheme('dark')}>Dark</button>
    </div>
  )
}
```

ข้อควรระวัง: ถ้าต้องการไม่มี FOUC (flash of wrong theme) บน SSR ให้ใส่ initial data-theme ใน HTML server-render (หรือใส่ inline script เล็ก ๆ เพื่ออ่าน prefered-theme จาก localStorage ก่อน CSS โหลด). (daisyui.com)

---

6) SSR / Hydration / Tailwind Purge (ปัญหาและการป้องกัน)

- **DOM API ใน Server Components:** ไม่เรียก document/window ใน server render — ถ้าคอมโพเนนต์เรียก .showModal() หรือ document.body ให้ทำเป็น Client Component ("use client") หรือเรียกใน useEffect. (Next.js docs). (nextjs.org)
- **Tailwind Purge & dynamic classes:** ถ้าคลาสสร้างแบบไดนามิก (เช่น btn-${color}) ให้ใช้ safelist ใน tailwind.config.js เพื่อไม่ให้ถูกลบเมื่อ build production. ตัวอย่างและหลักการที่ Tailwind ใช้ในการสแกนไฟล์อยู่ใน docs. (tailwindcss.com)

ตัวอย่าง safelist:

```
module.exports = {
  content: ['./src/**/*.{js,jsx,ts,tsx}'],
  safelist: [
    'btn-primary','btn-secondary','btn-accent',
    { pattern: /^bg-(red|green|blue)-\d{3}$/ } // ตัวอย่าง pattern
  ],
}
```

---

7) Accessibility checklist (สรุปให้ใช้จริง)

- ใช้ <dialog> ถ้าเป็นไปได้ (native ESC support) — daisyUI แนะนำ. (daisyui.com)

- ถ้าใช้ controlled modal: trap focus (ex. focus-trap-react), restore focus to trigger, handle Escape key, set role="dialog" + aria-modal="true" + aria-labelledby/aria-describedby. (GitHub)
- Dropdown: role="menu" / role="menuitem", keyboard navigation (Up/Down/Escape), and outside-click handling. (daisyui.com)

---

8) Testing & Debugging tips

- **Unit / integration tests:** ใช้ @testing-library/react + user-event เพื่อทดสอบการเปิด/ปิด modal, Escape key, focus restore.
- **Accessibility testing:** ใช้ axe-core / jest-axe เพื่อตรวจ violation ใน modal/dropdown.
- **Common bugs & fixes:**
  - **Dropdown ถูกตัด** ⟶ ตรวจ z-index และ render dropdown เป็น portal (mount บน body) หรือใช้ popover API.
  - **Class ไม่ขึ้น** ⟶ ตรวจ tailwind.config.content + @plugin 'daisyui' ถูกโหลดหรือไม่. (daisyui.com)

---

9) Packages / tools ที่แนะนำเพิ่มเติม

- focus-trap-react — focus trapping สำหรับ modal. (GitHub)
- classnames หรือ clsx — จัดการ className แบบมีเงื่อนไข
- @testing-library/react, user-event, jest-axe — testing & accessibility

---

10) ตัวอย่างสรุปคำสั่งติดตั้ง (Next.js + DaisyUI + FocusTrap)

npx create-next-app@latest my-app

cd my-app

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

npm i -D daisyui@latest

npm i focus-trap-react classnames

เพิ่ม plugin ใน tailwind.config.js และ @plugin "daisyui" ใน global css

(คู่มือการติดตั้งและตัวอย่าง React-specific มีใน docs). (daisyui.com)

สรุป — คำแนะนำเชิงปฏิบัติ (short checklist)

1. ติดตั้ง Tailwind ⟶ ติดตั้ง daisyUI (plugin). (daisyui.com)

2. เลือก modal approach: <dialog> (ถ้าต้องการ native accessibility) หรือ portal+focus-trap (ถ้าต้องการ full control). (daisyui.com)
3. สำหรับ dropdown: details/summary ถ้าต้องการง่าย & semantic; controlled dropdown ถ้าต้องการ custom keyboard behaviors. (daisyui.com)
4. จัดการ theme ผ่าน @plugin "daisyui" { themes: [...] } และสลับด้วย data-theme attribute. (daisyui.com)
5. ป้องกันปัญหา SSR / Purge: use client สำหรับ DOM API, และใช้ Tailwind safelist สำหรับ dynamic classes. (nextjs.org)

ติดตั้ง DaisyUI ใน Next.js / CRA / Vite + React

DaisyUI เป็น **Tailwind CSS Plugin** เพราะฉะนั้นทุกโปรเจกต์ React ที่ใช้ DaisyUI ต้องทำขั้นตอนหลัก 3 อย่าง:

1. ติดตั้ง **Tailwind CSS**
2. ติดตั้ง **DaisyUI**
3. ตั้งค่า tailwind.config.js ให้โหลด plugin DaisyUI

☐ 1) ติดตั้งใน Next.js (เช่น Next.js 13/14 App Router)

```
npx create-next-app my-daisyui-app
cd my-daisyui-app
npm install -D tailwindcss postcss autoprefixer daisyui
npx tailwindcss init -p
```

แก้ไข tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./app/**/*.{js,ts,jsx,tsx}",
    "./pages/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}"
  ],
  theme: {
    extend: {},
  },
  plugins: [require("daisyui")],
}
```

เพิ่ม globals.css

@tailwind base;

@tailwind components;

@tailwind utilities;

☐ เสร็จแล้ว คุณสามารถใช้ DaisyUI component ได้ทันที เช่น btn, card, modal

---

☐ **2) ติดตั้งใน CRA (Create React App)**

npx create-react-app my-daisyui-cra

cd my-daisyui-cra

npm install -D tailwindcss postcss autoprefixer daisyui

npx tailwindcss init -p

แก้ไข tailwind.config.js

```
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [require("daisyui")],
}
```

ใน src/index.css

@tailwind base;

@tailwind components;

@tailwind utilities;

แล้ว restart server (npm start) ⟶ ใช้ DaisyUI ได้เลย

☐ **3) ติดตั้งใน Vite + React**

npm create vite@latest my-daisyui-vite --template react

cd my-daisyui-vite

npm install -D tailwindcss postcss autoprefixer daisyui

npx tailwindcss init -p

แก้ไข tailwind.config.js

export default {

```
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [require("daisyui")],
}
```

ใน src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

แล้วรัน

```
npm run dev
```

☐ **สรุปความแตกต่าง**

- **Next.js** ⟶ ระบุ app/, pages/, components/ ใน content
- **CRA** ⟶ ระบุ src/ ทั้งหมด
- **Vite** ⟶ ระบุ index.html + src/

ทุกโปรเจกต์ React ใช้หลักเดียวกันคือ: **ติดตั้ง Tailwind ⟶ ติดตั้ง DaisyUI ⟶ config Tailwind ⟶ import CSS**

---

3 โปรแกรมพื้นฐาน และ **3 โปรแกรมแนวประยุกต์** แบบ **เต็มไฟล์ + โครงสร้างโปรเจกต์ + คำอธิบายโค้ด + ผลการรัน (UI ที่จะได้)**

☐ **โครงสร้างโปรเจกต์ (ใช้ Vite + React + DaisyUI)**

ทุกตัวอย่างจะมีโครงสร้างแบบเดียวกัน (ต่างกันที่ src/App.jsx)

```
my-daisyui-app/
├── node_modules/
├── public/
├── src/
│   ├── App.jsx       <-- ไฟล์หลักที่เปลี่ยนแต่ละตัวอย่าง
```

```
│   ├── main.jsx
│   ├── index.css
├── tailwind.config.js
├── package.json
```

ใน index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

---

3 โปรแกรมพื้นฐาน

1) ปุ่ม (Button) พื้นฐาน

ไฟล์: src/App.jsx

```
export default function App() {
  return (
    <div className="min-h-screen flex items-center justify-center bg-base-200">
      <button className="btn btn-primary">Click Me</button>
    </div>
  );
}
```

คำอธิบายโค้ด

- ใช้ DaisyUI btn btn-primary ⟶ ปุ่มสีน้ำเงิน
- Layout: ใช้ Tailwind flex items-center justify-center ให้ปุ่มอยู่กลางหน้าจอ

ผลการรัน

หน้าจอสีเทาอ่อน + ปุ่มน้ำเงิน “Click Me” อยู่ตรงกลาง

---

2) การ์ด (Card) พื้นฐาน

ไฟล์: src/App.jsx

```
export default function App() {
  return (
    <div className="min-h-screen flex items-center justify-center bg-base-100">
      <div className="card w-96 bg-base-200 shadow-xl">
        <div className="card-body">
          <h2 className="card-title">DaisyUI Card</h2>
          <p>This is a simple card using DaisyUI.</p>
```

```
          <div className="card-actions justify-end">
            <button className="btn btn-secondary">Action</button>
          </div>
        </div>
      </div>
    </div>
  );
}
```

คำอธิบายโค้ด

- ใช้ component card ของ DaisyUI
- card-body ⟶ เนื้อหาภายในการ์ด
- card-actions ⟶ ปุ่มอยู่ด้านล่างขวา

ผลการรัน

การ์ดสี่เหลี่ยมมีหัวข้อ “DaisyUI Card”, ข้อความ, และปุ่มสีม่วง “Action”

---

3) Modal พื้นฐาน

ไฟล์: src/App.jsx

```
import { useState } from "react";

export default function App() {
  const [open, setOpen] = useState(false);

  return (
    <div className="min-h-screen flex items-center justify-center bg-base-200">
      <button className="btn btn-accent" onClick={() => setOpen(true)}>
        Open Modal
      </button>

      {open && (
        <dialog className="modal" open>
          <div className="modal-box">
            <h3 className="font-bold text-lg">Hello!</h3>
            <p className="py-4">This is a simple DaisyUI modal.</p>
            <div className="modal-action">
```

```
            <button className="btn" onClick={() => setOpen(false)}>
              Close
            </button>
          </div>
        </div>
      </dialog>
    )}
  </div>
 );
}
```

คำอธิบายโค้ด

- ใช้ React state (useState) จัดการเปิด/ปิด modal
- DaisyUI modal + modal-box สร้าง modal UI

ผลการรัน

มีปุ่ม “Open Modal” ⟶ เมื่อกดจะเปิด popup modal ⟶ กด “Close” ปิดได้

□ 3 โปรแกรมแนวประยุกต์

4) Navbar + Dark Mode Toggle

ไฟล์: src/App.jsx

```
import { useState } from "react";

export default function App() {
 const [theme, setTheme] = useState("light");

 return (
  <div data-theme={theme} className="min-h-screen">
   <div className="navbar bg-base-300">
    <div className="flex-1">
     <a className="btn btn-ghost normal-case text-xl">MyApp</a>
    </div>
    <div className="flex-none">
     <button
      className="btn"
      onClick={() => setTheme(theme === "light" ? "dark" : "light")}
```

```
          >
            Toggle {theme === "light" ? "Dark" : "Light"}
          </button>
        </div>
      </div>
      <div className="p-6">
        <p>Current theme: {theme}</p>
      </div>
    </div>
  );
}
```

คำอธิบายโค้ด

- ใช้ navbar component ของ DaisyUI
- ใช้ React state จัดการ theme (light / dark)
- DaisyUI ใช้ data-theme="..." เพื่อเปลี่ยนธีม

ผลการรัน

Navbar ด้านบน + ปุ่ม Toggle theme ⟶ เปลี่ยน light/dark ได้ทันที

---

5) ฟอร์ม Login

ไฟล์: src/App.jsx

```
import { useState } from "react";

export default function App() {
  const [form, setForm] = useState({ email: "", password: "" });

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Login with ${form.email} / ${form.password}`);
  };

  return (
    <div className="min-h-screen flex items-center justify-center bg-base-200">
      <div className="card w-96 bg-base-100 shadow-xl">
        <div className="card-body">
```

```
          <h2 className="card-title">Login</h2>
          <form onSubmit={handleSubmit} className="form-control gap-4">
            <input
              type="email"
              placeholder="Email"
              className="input input-bordered"
              value={form.email}
              onChange={(e) => setForm({ ...form, email: e.target.value })}
            />
            <input
              type="password"
              placeholder="Password"
              className="input input-bordered"
              value={form.password}
              onChange={(e) => setForm({ ...form, password: e.target.value })}
            />
            <button className="btn btn-primary">Login</button>
          </form>
        </div>
      </div>
    </div>
  );
}
```

คำอธิบายโค้ด

- ใช้ input input-bordered ของ DaisyUI
- ใช้ React state เก็บ email/password
- กด Login ⟶ alert แสดงค่า

ผลการรัน

กล่อง Login form มี Email, Password, ปุ่ม Login

---

6) Dashboard + Card Stats

ไฟล์: src/App.jsx

```
export default function App() {
  return (
```

```
    <div className="min-h-screen bg-base-200 p-10">
      <h1 className="text-2xl font-bold mb-6">Dashboard</h1>
      <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
        <div className="stat bg-base-100 shadow">
          <div className="stat-title">Users</div>
          <div className="stat-value">1,200</div>
          <div className="stat-desc">↗ 15% from last month</div>
        </div>
        <div className="stat bg-base-100 shadow">
          <div className="stat-title">Sales</div>
          <div className="stat-value">$34K</div>
          <div className="stat-desc">↗ 22% from last month</div>
        </div>
        <div className="stat bg-base-100 shadow">
          <div className="stat-title">New Clients</div>
          <div className="stat-value">320</div>
          <div className="stat-desc">↘ 5% this month</div>
        </div>
      </div>
    </div>
  );
}
```

คำอธิบายโค้ด

- ใช้ DaisyUI stat component สำหรับการ์ดสถิติ
- Layout ใช้ grid grid-cols-3 ให้แสดงเป็น 3 คอลัมน์

ผลการรัน

Dashboard ที่มีการ์ดสถิติ Users, Sales, New Clients พร้อมตัวเลขและเปอร์เซ็นต์

สรุป

- **พื้นฐาน** ⟶ Button, Card, Modal (เรียนรู้ component และ state)
- **แนวประยุกต์** ⟶ Navbar + Dark Mode, Login Form, Dashboard Stats (ผสมหลาย component + React state)

---

6 ตัวอย่างก่อนหน้า (3 พื้นฐาน + 3 ประยุกต์) เข้าไว้ในเว็บเดียว โดยใช้ DaisyUI + React (ผ่าน Vite)

□ โครงสร้างโปรเจกต์

```
ultimate-daisyui-app/
├── src/
│   ├── components/
│   │   ├── BasicButton.jsx
│   │   ├── BasicCard.jsx
│   │   ├── BasicModal.jsx
│   │   ├── NavbarTheme.jsx
│   │   ├── LoginForm.jsx
│   │   └── DashboardStats.jsx
│   ├── App.jsx
│   ├── main.jsx
│   └── index.css
├── tailwind.config.js
└── package.json
```

⚡ ตัวอย่างไฟล์ทั้งหมด

1) src/components/BasicButton.jsx

```
export default function BasicButton() {
  return (
    <div className="p-6 flex justify-center">
      <button className="btn btn-primary">Click Me</button>
    </div>
  );
}
```

---

2) src/components/BasicCard.jsx

```
export default function BasicCard() {
  return (
    <div className="p-6 flex justify-center">
```

```
      <div className="card w-96 bg-base-200 shadow-xl">
        <div className="card-body">
          <h2 className="card-title">DaisyUI Card</h2>
          <p>This is a simple card using DaisyUI.</p>
          <div className="card-actions justify-end">
            <button className="btn btn-secondary">Action</button>
          </div>
        </div>
      </div>
    </div>
  );
}
```

---

3) src/components/BasicModal.jsx

```
import { useState } from "react";

export default function BasicModal() {
  const [open, setOpen] = useState(false);

  return (
    <div className="p-6 flex justify-center">
      <button className="btn btn-accent" onClick={() => setOpen(true)}>
        Open Modal
      </button>

      {open && (
        <dialog className="modal" open>
          <div className="modal-box">
            <h3 className="font-bold text-lg">Hello!</h3>
            <p className="py-4">This is a simple DaisyUI modal.</p>
            <div className="modal-action">
              <button className="btn" onClick={() => setOpen(false)}>
                Close
              </button>
```

```
            </div>
          </div>
        </dialog>
      )}
    </div>
  );
}
```

4) src/components/NavbarTheme.jsx

```
import { useState } from "react";

export default function NavbarTheme() {
  const [theme, setTheme] = useState("light");

  return (
    <div data-theme={theme} className="min-h-screen">
      <div className="navbar bg-base-300 shadow">
        <div className="flex-1">
          <a className="btn btn-ghost normal-case text-xl">UltimateApp</a>
        </div>
        <div className="flex-none">
          <button
            className="btn"
            onClick={() => setTheme(theme === "light" ? "dark" : "light")}
          >
            Toggle {theme === "light" ? "Dark" : "Light"}
          </button>
        </div>
      </div>
    </div>
  );
}
```

5) src/components/LoginForm.jsx

```
import { useState } from "react";

export default function LoginForm() {
  const [form, setForm] = useState({ email: "", password: "" });

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Login with ${form.email} / ${form.password}`);
  };

  return (
    <div className="p-6 flex justify-center">
      <div className="card w-96 bg-base-100 shadow-xl">
        <div className="card-body">
          <h2 className="card-title">Login</h2>
          <form onSubmit={handleSubmit} className="form-control gap-4">
            <input
              type="email"
              placeholder="Email"
              className="input input-bordered"
              value={form.email}
              onChange={(e) => setForm({ ...form, email: e.target.value })}
            />
            <input
              type="password"
              placeholder="Password"
              className="input input-bordered"
              value={form.password}
              onChange={(e) => setForm({ ...form, password: e.target.value })}
            />
            <button className="btn btn-primary">Login</button>
          </form>
        </div>
      </div>
```

```
    </div>
  );
}
```

6) src/components/DashboardStats.jsx

```
export default function DashboardStats() {
  return (
    <div className="p-6">
      <h1 className="text-2xl font-bold mb-6">Dashboard</h1>
      <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
        <div className="stat bg-base-100 shadow">
          <div className="stat-title">Users</div>
          <div className="stat-value">1,200</div>
          <div className="stat-desc">↗ 15% from last month</div>
        </div>
        <div className="stat bg-base-100 shadow">
          <div className="stat-title">Sales</div>
          <div className="stat-value">$34K</div>
          <div className="stat-desc">↗ 22% from last month</div>
        </div>
        <div className="stat bg-base-100 shadow">
          <div className="stat-title">New Clients</div>
          <div className="stat-value">320</div>
          <div className="stat-desc">↘ 5% this month</div>
        </div>
      </div>
    </div>
  );
}
```

7) src/App.jsx (รวมทุกตัวอย่าง)

```
import BasicButton from "./components/BasicButton";
import BasicCard from "./components/BasicCard";
import BasicModal from "./components/BasicModal";
```