Tailwind CSS

Framework: Professional

(Integrative-Generative AI Edition)

Student Price Book Center

คำนำ

ในยุคที่เทคโนโลยีเว็บพัฒนาไปอย่างรวดเร็ว การออกแบบและพัฒนา UI ที่สวยงาม ยืดหยุ่น และบำรุงรักษาได้ในระยะยาวกลายเป็นความท้าทายที่นักพัฒนาทุกระดับต้องเผชิญ Tailwind CSS ได้เข้ามาเปลี่ยนแนวทางการเขียน CSS แบบดั้งเดิม ด้วยการนำเสนอแนวคิด utility-first ที่ทำให้การสร้างอินเทอร์เฟซมีความยืดหยุ่นสูงและมีความสอดคล้องกับหลักการ component-based design อย่างแท้จริง หนังสือ *Tailwind CSS Framework: Professional* เล่มนี้จึงถูกเขียนขึ้นเพื่อตอบโจทย์มืออาชีพที่ต้องการยกระดับการใช้ Tailwind CSS จากระดับพื้นฐานไปสู่การพัฒนา Design System, UI Library และ Workflow ระดับทีมในองค์กร

ส่วนท้ายของหนังสือเล่มนี้ (บทที่ 16–20) ได้ถูกรวบรวมและเรียบเรียงขึ้นอย่างเข้มข้น เพื่อให้ผู้อ่านเข้าใจมิติของการใช้งาน Tailwind CSS ในระดับที่ลึกขึ้น เริ่มจากบทที่ 16 "สร้าง Design System" ที่ชี้ให้เห็นถึงความสำคัญของการวางรากฐานการออกแบบ UI ผ่านการกำหนดขนาด (scale) ของ spacing, font, line height และระบบสีอย่างเป็นระบบ พร้อมทั้งใช้ความสามารถของ Tailwind CSS เช่น variants, screens, และ extend เพื่อสร้างการออกแบบที่สอดคล้องกันในระดับ global โดยผ่านการกำหนด theme และ design token อย่างมีประสิทธิภาพ

ต่อเนื่องในบทที่ 17 "การสร้าง UI Library" ผู้อ่านจะได้เรียนรู้การนำ Tailwind CSS มาผนวกกับเครื่องมือยอดนิยมอย่าง Storybook เพื่อพัฒนาและทดสอบ component แบบแยกส่วน พร้อมแนวคิด Atomic Design ที่ช่วยจัดโครงสร้าง UI ให้ยืดหยุ่นและปรับขนาดได้ง่าย การใช้งาน @layer components และ @apply เพื่อสร้าง mixin และ reusable component ทำให้เกิด workflow การพัฒนา UI ที่สอดคล้องกันในระดับทีมและสามารถนำไปใช้งานในหลาย framework ได้อย่างคล่องตัว

ในบทที่ 18 "Production Optimization" หนังสือจะพาผู้อ่านเจาะลึกแนวทางการปรับปรุงประสิทธิภาพของ CSS ในการใช้งานจริง ตั้งแต่เทคนิคการ purge class ที่ไม่ได้ใช้งานด้วย Tailwind CLI หรือ PurgeCSS ไปจนถึงการลดขนาดไฟล์ CSS อย่างมีประสิทธิภาพผ่านการ minify และการจัดการ version ของไฟล์ เพื่อให้เหมาะกับระบบ production ที่ต้องการความรวดเร็วและเบาโหลด

เมื่อทีมพัฒนาเติบโตขึ้น การจัดการเวิร์กโฟลว์ระหว่างการพัฒนาและการ deploy จึงเป็นสิ่งสำคัญ บทที่ 19 "Workflow ทีมและ CI/CD" จะช่วยให้ผู้อ่านเข้าใจการจัดการ version ของ Tailwind ระหว่าง dev/prod การตั้งค่า lint rule เช่น Tailwind CSS IntelliSense และ Prettier ตลอดจนการผสาน Tailwind CSS เข้ากับระบบ Continuous Integration และ Continuous Deployment (CI/CD) เพื่อให้การพัฒนาและการนำขึ้นใช้งานเกิดขึ้นโดยอัตโนมัติและเป็นระบบ

สุดท้ายในบทที่ 20 "Best Practices & Anti-patterns" หนังสือจะสรุปแนวทางการใช้งาน Tailwind CSS อย่างมืออาชีพ โดยเน้นจุดที่ควรปฏิบัติ เช่น การจัด class อย่างเป็นระบบ, ใช้ @apply อย่างเหมาะสม, ยึดหลัก semantic HTML และการวางแผนโครงสร้างให้พร้อมสำหรับการ scale ในระยะยาว พร้อมทั้งเตือนถึงจุดที่ควรหลีกเลี่ยง เช่น การ override utility class โดยตรง หรือการซ้อน class มากเกินไป ซึ่งจะส่งผลให้โค้ดดูแลยากและเสี่ยงต่อความผิดพลาดในอนาคต

ตลอดทั้งห้าบท ผู้อ่านจะได้รับทั้งแนวคิด หลักการ และตัวอย่างการประยุกต์ใช้จริงที่สามารถนำไปใช้ได้ในโครงการจริงทันที ทั้งในระดับบุคคลและระดับทีมองค์กร ไม่ว่าจะเป็นผู้ที่ทำงานด้าน front-end, UX/UI designer หรือ full-stack developer การเข้าใจการจัดระบบ Tailwind CSS ให้เป็น Design System และ UI Library จะช่วยเพิ่มความคล่องตัวและความแม่นยำในการพัฒนาได้อย่างมีประสิทธิภาพ

หวังว่าเนื้อหาในส่วนท้ายของหนังสือ *Tailwind CSS Framework: Professional* นี้จะเป็นเข็มทิศให้กับนักพัฒนาทุกท่านในการสร้างระบบ UI ที่ยั่งยืน ขยายได้ และดูแลต่อได้ง่ายในระยะยาว ทั้งยังสามารถเป็นรากฐานสำคัญในการพัฒนาระบบ Design และ DevOps ที่ทันสมัยในองค์กรของคุณ

ขอให้ทุกท่านสนุกกับการเรียนรู้ และนำ Tailwind CSS ไปใช้สร้างประสบการณ์ที่ยอดเยี่ยมทั้งสำหรับทีมพัฒนาและผู้ใช้ปลายทาง

ด้วยรักและปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 16
สร้าง Design System
(Design System)

เนื้อหา

- สร้าง Design System ด้วย Tailwind CSS
- สร้าง Design System ด้วย Tailwind CSS (รายละเอียดเชิงลึก)
- กำหนด scale ของ Spacing, Font size, Line height, Color
- การกำหนด Variants, Screens และ Extend ใน Tailwind CSS
- กำหนด Global Design Token ผ่าน theme ใน Tailwind CSS
- ใช้ระบบ Spacing / Typography ที่สอดคล้องกันใน Tailwind CSS
- ตัวอย่างบูรณาการ

บทนำบทที่ 16: สร้าง Design System

ในยุคที่การพัฒนาเว็บแอปพลิเคชันมีความซับซ้อนมากขึ้น และประสบการณ์ของผู้ใช้กลายเป็นศูนย์กลางของการออกแบบ การมีระบบการออกแบบ (Design System) ที่แข็งแรงและสามารถปรับใช้ซ้ำได้ (reusable) ถือเป็นองค์ประกอบสำคัญที่จะช่วยให้ทีมพัฒนาและออกแบบสามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ ลดความซ้ำซ้อนของโค้ด และสร้างความสม่ำเสมอในการแสดงผลทุกส่วนของแอปพลิเคชัน ทั้งในด้าน UI, UX และแบรนด์ขององค์กร

Tailwind CSS ในฐานะ CSS Framework แบบ Utility-first ถูกออกแบบมาเพื่อรองรับการสร้าง Design System อย่างยืดหยุ่นและมีประสิทธิภาพ ผ่านการกำหนดค่าในไฟล์ tailwind.config.js ที่สามารถควบคุมทุกอย่างตั้งแต่สเกลของ spacing, ขนาดตัวอักษร (font size), ความสูงบรรทัด (line height) ไปจนถึงชุดสี (color palette) ซึ่งสามารถกำหนดให้สอดคล้องกับแนวคิดของ Global Design Token ได้อย่างสมบูรณ์

ในบทนี้ ผู้อ่านจะได้เรียนรู้วิธีการกำหนด **scale** ที่เป็นระบบ เช่น spacing (ระยะห่าง), fontSize (ขนาดตัวอักษร), lineHeight (ความสูงของบรรทัด) และ color (ชุดสีหลัก-รอง) เพื่อสร้างพื้นฐานของ UI ที่คงเส้นคงวา การกำหนดสเกลที่เหมาะสมจะช่วยให้สามารถขยายหรือปรับลดขนาดของ UI ได้โดยไม่กระทบต่อความกลมกลืนของการออกแบบโดยรวม

นอกจากนี้ยังกล่าวถึงการกำหนด **variants**, **responsive screens** และการ **extend** class หรือ utility ที่มีอยู่ เพื่อให้ระบบสามารถตอบสนองกับสถานะและขนาดหน้าจอที่แตกต่างกันได้อย่าง

ยืดหยุ่น โดยไม่ต้องพึ่งพา CSS ภายนอกหรือเขียน class เพิ่มจำนวนมาก ซึ่งจะช่วยลดภาระของนักพัฒนาและเพิ่มความเร็วในการออกแบบและพัฒนา

อีกหนึ่งหัวใจของบทนี้คือแนวทางการกำหนด **Global Design Token** ผ่าน theme ซึ่งเป็นระบบที่ช่วยให้การควบคุมค่าสำคัญต่าง ๆ เช่น สี ขนาด ระยะห่าง และฟอนต์ เป็นไปอย่างมีโครงสร้าง ทำให้สามารถบำรุงรักษาและเปลี่ยนแปลงได้ง่ายในระดับโปรเจกต์ทั้งหมด โดยไม่ต้องแก้โค้ดในหลายจุด

เรายังจะสำรวจวิธีการวางโครงสร้างระบบ spacing และ typography ที่ **สอดคล้องกันทั้งระบบ** เพื่อให้การจัด layout และการจัดวางข้อความใน UI มีความชัดเจน สะอาดตา และมีความรู้สึกเป็นระบบเดียวกันอย่างแท้จริง ซึ่งถือเป็นแกนหลักของ Design System ที่ดี ไม่เพียงแต่ในด้านความงาม แต่ยังส่งผลต่อการใช้งานจริงของผู้ใช้

ท้ายที่สุด บทนี้จะช่วยให้ผู้อ่านเข้าใจว่า “Design System” ไม่ใช่เพียงเอกสารหรือ guideline ที่ใช้ดู แต่สามารถถูก **ฝังอยู่ในโค้ด** และกลายเป็นส่วนหนึ่งของกระบวนการพัฒนาได้อย่างแท้จริง ด้วยพลังของ Tailwind CSS ที่สามารถนำไปสู่การสร้าง UI ที่สวยงาม ยั่งยืน และปรับขยายได้อย่างง่ายดายในระยะยาว

สร้าง Design System ด้วย Tailwind CSS

---

1. กำหนด scale ของ Spacing, Font size, Line height, Color

- Tailwind CSS มาพร้อมกับ **scale** ที่ออกแบบมาให้ใช้ซ้ำได้ง่าย เช่น
  - **Spacing** (margin, padding, gap) มีค่าเป็น 0, 0.5, 1, 1.5, 2, 2.5, 3, ... (หน่วย rem) เช่น p-4 เท่ากับ 1rem (16px)
  - **Font size** กำหนดระดับตัวอักษร เช่น text-xs, text-sm, text-base, text-lg, text-xl, ...
  - **Line height** กำหนดความสูงบรรทัด เช่น leading-none, leading-tight, leading-normal, leading-loose
  - **Color** ใช้ระบบสีที่กำหนดเป็นชุด เช่น gray-100 ถึง gray-900, blue-500, red-700 ซึ่งเป็นแบบระบบสีที่มีความสม่ำเสมอ
- การกำหนด scale เหล่านี้ช่วยให้ UI มีความสอดคล้องและคาดเดาได้ง่ายในการออกแบบ

---

2. การกำหนด Variants, Screens, Extend

- ในไฟล์ tailwind.config.js เราสามารถ **เพิ่ม** หรือ **แก้ไข** ค่าพื้นฐาน Tailwind เช่น

```
module.exports = {
  theme: {
    extend: {
      colors: {
        brand: {
```

```
          light: '#3AB0FF',
          DEFAULT: '#0077FF',
          dark: '#005FCC',
        },
      },
      spacing: {
        '72': '18rem',
        '84': '21rem',
        '96': '24rem',
      },
      fontSize: {
        'xxs': '0.625rem',  // 10px
      },
    },
  },
  variants: {
    extend: {
      backgroundColor: ['active', 'group-hover'],
      textColor: ['visited'],
    },
  },
  screens: {
    'xs': '475px',
    ... // เพิ่ม breakpoint ใหม่หรือตั้งค่า breakpoint เดิมใหม่
  },
};
```

- **extend** ใช้เพิ่ม scale ใหม่โดยไม่ลบค่าพื้นฐาน
- **variants** คือการกำหนดให้ utility สามารถใช้กับสถานะต่าง ๆ เช่น hover, focus, active, visited
- **screens** กำหนด breakpoint สำหรับ responsive design เช่น sm, md, lg แต่เราสามารถเพิ่ม breakpoint ใหม่ เช่น xs ได้

---

3. กำหนด global design token ผ่าน theme

- การใช้ theme ใน tailwind.config.js เหมือนเป็น **central source of truth** หรือ **design token** สำหรับโปรเจกต์ เช่น

```
module.exports = {
  theme: {
    colors: {
      primary: '#2563EB',
      secondary: '#4B5563',
      accent: '#D97706',
      background: '#F3F4F6',
      text: '#1F2937',
    },
    fontSize: {
      base: ['16px', '24px'],
      lg: ['18px', '28px'],
    },
    spacing: {
      xs: '4px',
      sm: '8px',
      md: '16px',
      lg: '24px',
      xl: '32px',
    },
  },
};
```

- พัฒนาให้ทีมใช้สี ขนาด ระยะห่าง เดียวกัน
- สามารถเรียกใช้งานใน CSS/HTML ด้วย text-primary, bg-background, p-md, text-lg เป็นต้น

---

4. ใช้ระบบ spacing / typography ที่สอดคล้องกัน

- หลีกเลี่ยงการกำหนดค่าแบบตัวเลขสุ่ม ไม่สม่ำเสมอ
- กำหนด scale spacing ชัดเจน เช่น padding/margin ทุกระดับเท่ากับ unit เดียวกัน
- Typography ต้องใช้ font-size และ line-height ที่เหมาะสม และสอดคล้องกัน เช่น

```
<p class="text-base leading-relaxed mb-4">This is paragraph text with consistent spacing and line height.</p>
```

- สร้างความสมดุลใน UI และง่ายต่อการบำรุงรักษา

สรุป

การสร้าง Design System ด้วย Tailwind CSS

- ใช้ tailwind.config.js เพื่อกำหนด scale สี, spacing, font size เป็น centralized design tokens
- กำหนด variants และ screens เพื่อรองรับสถานะและ responsive
- รักษาความสม่ำเสมอในระบบ spacing และ typography เพื่อ UI ที่ดีและง่ายต่อทีม
- ช่วยให้การออกแบบและพัฒนาเร็วขึ้น และลดข้อผิดพลาดในการใช้ style

สร้าง Design System ด้วย Tailwind CSS (รายละเอียดเชิงลึก)

1. กำหนด Scale ของ Spacing, Font Size, Line Height, Color อย่างละเอียด

- **Spacing Scale (ระยะห่าง)**

  Tailwind ใช้หน่วย rem เป็นหลัก ทำให้ scale มีความสัมพันธ์กับขนาดฟอนต์ฐาน เช่น
  - 1 = 0.25rem (4px)
  - 2 = 0.5rem (8px)
  - 4 = 1rem (16px)
  - 8 = 2rem (32px)

    การใช้ spacing scale ที่เหมาะสมและคงที่ช่วยให้องค์ประกอบ UI มีระยะห่างสมดุลและทำให้ UI ดูเป็นระบบเดียวกัน
- **Font Size & Line Height**

  ฟอนต์แต่ละขนาดต้องจับคู่กับ line height ที่เหมาะสมเพื่อให้อ่านง่ายและสบายตา เช่น
- fontSize: {
- sm: ['0.875rem', { lineHeight: '1.25rem' }], // 14px font size + 20px line height
- base: ['1rem', { lineHeight: '1.5rem' }], // 16px + 24px
- lg: ['1.125rem', { lineHeight: '1.75rem' }],// 18px + 28px
- }

  การกำหนดคู่ font-size + line-height เป็น design token ช่วยให้ typography มีความสอดคล้องกัน
- **Color Palette**

  ควรกำหนดสีหลักและโทนสีเสริม (primary, secondary, accent, neutral) และใช้ scale เช่น gray-100 ถึง gray-900

  การใช้โทนสีที่มีค่าความสว่างและความเข้มคงที่ช่วยให้การจัดวางสีใน UI สวยงามและเข้ากันได้ดี

2. กำหนด Variants, Screens, Extend อย่างละเอียด

- **Variants** คือการขยาย utility classes ให้รองรับสถานะพิเศษ เช่น hover, focus, active, visited

  เช่นใน tailwind.config.js
- variants: {
- extend: {
- backgroundColor: ['hover', 'focus', 'active', 'group-hover'],
- textColor: ['visited', 'focus-visible'],
- borderColor: ['focus-visible'],
- }
- }

  การเพิ่ม variants ที่จำเป็นทำให้เราสามารถเขียน UI โต้ตอบได้หลากหลายแบบ พร้อมดูแล accessibility
- **Screens (Breakpoints)**

  Tailwind ใช้ระบบ breakpoint เพื่อ responsive design

  ค่าเริ่มต้นเช่น
- screens: {
- sm: '640px',
- md: '768px',
- lg: '1024px',
- xl: '1280px',
- '2xl': '1536px',
- }

  เราสามารถเพิ่มหรือลด breakpoint ตามความเหมาะสมโปรเจกต์ เช่น เพิ่ม breakpoint สำหรับมือถือขนาดเล็ก (xs: '475px')
- **Extend**

  การใช้ extend ใน theme ช่วยเพิ่มค่าที่กำหนดเองโดยไม่ไปแทนที่ค่าเริ่มต้น

  เช่น
- theme: {
- extend: {
- spacing: {
- '72': '18rem',
- '84': '21rem',

- },
- colors: {
- brandBlue: '#0d6efd',
- },
- }
- }

ช่วยให้ Design System ของเราแยกชัดเจนและปรับเปลี่ยนง่าย

---

3. กำหนด Global Design Token ผ่าน Theme อย่างละเอียด

- ใช้ tailwind.config.js เป็นที่เก็บ design tokens ทั้งหมดของโปรเจกต์ เพื่อให้ทีมงานทุกคนใช้แบบเดียวกัน
- ตัวอย่างเช่น

```
module.exports = {
  theme: {
    colors: {
      primary: '#2563EB',        // สีหลัก
      secondary: '#4B5563',      // สีรอง
      accent: '#D97706',         // สีเน้น
      background: '#F3F4F6',     // สีพื้นหลัง
      text: '#1F2937',           // สีข้อความหลัก
    },
    spacing: {
      xs: '4px',
      sm: '8px',
      md: '16px',
      lg: '24px',
      xl: '32px',
    },
    fontSize: {
      base: ['16px', '24px'],    // font size + line height
      lg: ['18px', '28px'],
    },
  }
}
```

- การจัดเก็บ tokens เหล่านี้ในไฟล์เดียว ทำให้สามารถควบคุมแก้ไขและรีแฟคเตอร์ UI ได้รวดเร็วและมีประสิทธิภาพ

---

4. การใช้ระบบ Spacing และ Typography ที่สอดคล้องกัน

- **Spacing consistency**
  ควรใช้ scale เดียวกันสำหรับ padding, margin, gap ทุกที่ในโปรเจกต์ เช่น
- ...
  หลีกเลี่ยงการกำหนดค่า spacing แบบสุ่มหรือแตกต่างกันเพราะจะทำให้ UI ดูไม่สม่ำเสมอ
- **Typography consistency**
  เลือกใช้ชุด font-size กับ line-height ที่ตั้งไว้แล้วใน design token เช่น
- <h1 class="text-xl leading-snug">Title</h1>
- Paragraph text
  เพื่อให้ข้อความมีความสมดุลและอ่านง่าย

สรุป

- การสร้าง Design System ช่วยให้ UI สม่ำเสมอ ใช้งานง่าย และดูเป็นมืออาชีพ
- Tailwind CSS มีเครื่องมือและ config ที่ช่วยให้เราสร้างและดูแล design token ได้ดีมาก
- การใช้ scale ที่ชัดเจนและ extend config ช่วยให้งานง่ายและแก้ไขเร็ว
- การกำหนด variants และ screens ทำให้ UI รองรับทุกสถานะและทุกอุปกรณ์ได้ดี

กำหนด scale ของ Spacing, Font size, Line height, Color

---

1. กำหนด Scale ของ Spacing, Font Size, Line Height, Color

Spacing Scale

- Spacing หมายถึงค่าระยะห่าง เช่น margin, padding, gap
- ใน Tailwind CSS, spacing ใช้หน่วย rem ที่สัมพันธ์กับขนาดฟอนต์ฐาน (1rem = 16px โดยปกติ)
- Tailwind ให้ค่า spacing ที่สม่ำเสมอ เช่น

ชื่อ spacing class	ค่า rem	ค่า px
p-0	0rem	0px
p-1	0.25rem	4px
p-2	0.5rem	8px

ชื่อ spacing class	ค่า rem	ค่า px
p-4	1rem	16px
p-6	1.5rem	24px
p-8	2rem	32px
p-12	3rem	48px

- การใช้ spacing scale แบบนี้ช่วยให้ UI มีระยะห่างที่สมดุลและดูเรียบร้อย
- สามารถขยาย scale ได้ใน tailwind.config.js ถ้าต้องการ

Font Size Scale

- ขนาดฟอนต์ที่ใช้งานควรตั้งค่าที่ชัดเจนและครอบคลุมทุกระดับการใช้งาน เช่น
  - เล็ก (caption, helper text)
  - ปกติ (paragraph)
  - ใหญ่ (หัวข้อ, title)
- ตัวอย่างค่า font size ใน Tailwind (โดยค่าเริ่มต้น):

ชื่อ class	ขนาด (rem)	ขนาด (px)	คำอธิบาย
text-xs	0.75rem	12px	ขนาดเล็กมาก
text-sm	0.875rem	14px	เล็ก
text-base	1rem	16px	ปกติ
text-lg	1.125rem	18px	ใหญ่เล็กน้อย
text-xl	1.25rem	20px	ใหญ่
text-2xl	1.5rem	24px	ใหญ่มาก

- ควรกำหนดคู่กับ **Line Height** เพื่อให้อ่านง่าย

Line Height Scale

- ความสูงบรรทัด (line height) เป็นส่วนสำคัญในการทำให้ข้อความอ่านง่ายและดูสมดุล
- Tailwind ให้ class เช่น

ชื่อ class	ค่า line height	คำอธิบาย
leading-none	1	ค่าต่ำสุด
leading-tight	1.25	กระชับ
leading-snug	1.375	กระชับแต่พอดี

ชื่อ class	ค่า line height	คำอธิบาย
leading-normal	1.5	ปกติ (ค่า default)
leading-relaxed	1.625	ผ่อนคลาย, โปร่ง
leading-loose	2	โปร่งมาก

- มักจับคู่กับ font size เพื่อความสวยงามและความอ่านง่าย เช่น
  - text-base leading-relaxed
  - text-xl leading-snug

Color Scale

- สีในระบบ Design System ควรถูกกำหนดแบบมี **โทนสีและระดับความเข้มต่าง ๆ** (Shades) เพื่อความสม่ำเสมอในการใช้งาน
- Tailwind มีโทนสีหลัก ๆ เช่น gray, red, blue, green ที่ไล่ระดับจากอ่อน (100) ถึงเข้ม (900)
- ตัวอย่างเช่น

สี	สีสว่าง (100)	สีปานกลาง (500)	สีเข้ม (900)
Gray	#F3F4F6	#6B7280	#111827
Blue	#DBEAFE	#3B82F6	#1E40AF
Red	#FEE2E2	#EF4444	#7F1D1D

- ในการสร้าง Design System เราควรกำหนดสีหลัก (primary), สีรอง (secondary), สีเน้น (accent), สีพื้นหลัง (background), สีข้อความ (text) ให้ชัดเจนและใช้ซ้ำในโปรเจกต์

สรุป

Scale	ตัวอย่างค่ามาตรฐาน Tailwind CSS	เหตุผลการกำหนด scale
Spacing	0, 0.25rem, 0.5rem, 1rem, 1.5rem	สร้างระยะห่างที่สม่ำเสมอใน UI
Font Size	0.75rem, 1rem, 1.25rem, 1.5rem	สร้าง hierarchy และความชัดเจนในการอ่าน
Line Height	1, 1.25, 1.5, 1.625	เพิ่ม readability และความสวยงาม
Color	สีหลักและโทนสีต่างระดับ (100-900)	สร้างความสม่ำเสมอของสีในโปรเจกต์

นี่คือตัวอย่างโปรแกรม 3 ตัวอย่างแบบเต็มไฟล์ พร้อมโครงสร้างและคำอธิบายโค้ด เน้นเรื่องการกำหนด scale ของ Spacing, Font size, Line height, Color ด้วย Tailwind CSS

ตัวอย่างที่ 1: Basic Typography & Spacing Scale

โครงสร้างไฟล์

```
spacing-fontsize-1/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Typography & Spacing Scale</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-gray-50 p-8">
  <section class="max-w-xl mx-auto bg-white p-8 rounded shadow">
    <h1 class="text-3xl font-bold mb-6 leading-snug text-gray-900">Heading 1 (text-3xl, leading-snug)</h1>
    <p class="text-base leading-relaxed mb-4 text-gray-700">This paragraph uses <code>text-base</code> and <code>leading-relaxed</code> for comfortable reading.</p>
    <p class="text-sm leading-normal mb-2 text-gray-600">Smaller paragraph with <code>text-sm</code> and <code>leading-normal</code>.</p>
    <p class="text-xs leading-tight text-gray-500">Smallest text with tighter line height.</p>
  </section>
</body>
</html>
```

คำอธิบายโค้ด

- ใช้ Tailwind class เพื่อกำหนดขนาดตัวอักษรและระยะห่างบรรทัด
- text-3xl กับ leading-snug ทำให้หัวข้อใหญ่และกระชับ
- text-base กับ leading-relaxed ในย่อหน้าช่วยให้อ่านง่าย
- การใช้ mb-6, mb-4 เป็นการกำหนดระยะห่างระหว่างบล็อกข้อความตาม scale ของ Tailwind

ผลการรัน

- หน้าเว็บมีหัวข้อใหญ่ และย่อหน้าที่ใช้ขนาดและระยะบรรทัดแตกต่างกัน ทำให้ UI สวยงามและอ่านง่าย

ตัวอย่างที่ 2: Card Layout ใช้ Spacing และ Color Scale

โครงสร้างไฟล์

```
spacing-fontsize-2/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Card Layout with Color & Spacing</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-gray-100 p-10">
  <div class="max-w-4xl mx-auto grid grid-cols-1 sm:grid-cols-2 gap-6">
    <div class="bg-white p-6 rounded-lg shadow-md border border-gray-200">
      <h2 class="text-xl font-semibold mb-3 text-primary">Card Title 1</h2>
      <p class="text-gray-700 leading-relaxed mb-4">This card uses <code>text-primary</code>
and spacing utilities like <code>p-6</code> and <code>mb-3</code>.</p>
      <button class="bg-primary text-white px-4 py-2 rounded hover:bg-primary-dark
transition">Action</button>
    </div>
    <div class="bg-white p-6 rounded-lg shadow-md border border-gray-200">
      <h2 class="text-xl font-semibold mb-3 text-primary">Card Title 2</h2>
      <p class="text-gray-700 leading-relaxed mb-4">Second card with the same design tokens
for consistency.</p>
      <button class="bg-primary text-white px-4 py-2 rounded hover:bg-primary-dark
transition">Action</button>
    </div>
  </div>
</body>
</html>
```

คำอธิบายโค้ด

- ใช้ระบบ grid แบ่งเป็น 1 หรือ 2 คอลัมน์ตามหน้าจอ

- ใช้สี primary และสีเข้มสำหรับปุ่ม hover ซึ่งต้องกำหนดใน tailwind.config.js (ดูตัวอย่างข้างล่าง)
- spacing เช่น p-6, mb-3, px-4 py-2 ช่วยจัด layout ให้ดูมีช่องว่างพอเหมาะ

tailwind.config.js (จำลอง)

```
// สมมติเราเพิ่มสี custom ใน config
tailwind.config = {
  theme: {
    extend: {
      colors: {
        primary: '#2563EB',
        'primary-dark': '#1E40AF',
      },
    },
  },
};
```

ผลการรัน

- หน้าเว็บแสดงการ์ดสองใบที่มีระยะห่างและสีสม่ำเสมอ
- ปุ่มมีเอฟเฟกต์ hover สีเข้ม

ตัวอย่างที่ 3: Responsive Typography และ Color Mode

โครงสร้างไฟล์

```
spacing-fontsize-3/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en" class="dark">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Responsive Typography & Dark Mode</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-white dark:bg-gray-900 text-gray-900 dark:text-gray-100 p-8">
  <article class="max-w-3xl mx-auto">
```

```
    <h1 class="text-4xl md:text-5xl font-extrabold mb-6 leading-tight">Responsive Heading</h1>
    <p class="text-base md:text-lg leading-relaxed mb-4">
      This paragraph adjusts font size responsively and changes color in dark mode.
    </p>
    <p class="text-sm md:text-base leading-normal">
      Smaller text that also adapts responsively and to color scheme.
    </p>
  </article>
</body>
</html>
```

คำอธิบายโค้ด

- ใช้ responsive typography โดยกำหนดขนาดฟอนต์ text-4xl เป็นค่าเริ่มต้น และเพิ่มเป็น text-5xl ในจอขนาดกลางขึ้นไป (md:)
- ใช้ dark mode class เช่น dark:bg-gray-900 และ dark:text-gray-100 เพื่อเปลี่ยนสีพื้นหลังและข้อความตามโหมด
- spacing เช่น mb-6, mb-4 ควบคุมระยะห่างให้เหมาะสม

ผลการรัน

- ข้อความหัวข้อและย่อหน้าปรับขนาดตามหน้าจอ
- สีเปลี่ยนตามโหมด light/dark โดยอัตโนมัติ

การกำหนด Variants, Screens และ Extend ใน Tailwind CSS

---

1. Variants

- Variants คือ การขยาย utility classes ให้รองรับสถานะ (state) หรือเงื่อนไขพิเศษ เช่น

Variant	ความหมาย	ตัวอย่างใช้งาน
hover	เมื่อเมาส์ชี้บน element	hover:bg-blue-500
focus	เมื่อ element ได้ focus	focus:outline-none
active	เมื่อ element ถูกคลิก	active:scale-95
group-hover	เมื่อกลุ่ม parent ถูก hover	group-hover:text-red-500
dark	โหมด Dark	dark:bg-gray-900

- Tailwind มี variants เริ่มต้นให้ แต่เราสามารถขยายเพิ่มเติมได้ใน tailwind.config.js

*ตัวอย่างขยาย **variants***

```
module.exports = {
  variants: {
    extend: {
      backgroundColor: ['active', 'group-hover', 'dark'],
      textColor: ['visited', 'focus-visible'],
      borderColor: ['focus-visible'],
      opacity: ['disabled'],
    }
  }
}
```

- ตัวอย่างนี้ทำให้เราใช้ class เช่น active:bg-red-600 หรือ disabled:opacity-50

---

2. Screens (Breakpoints)

- Screens คือการกำหนด breakpoint สำหรับ Responsive Design
- Tailwind มี breakpoint เริ่มต้น (ตามขนาดหน้าจอ):

Breakpoint	ขนาดหน้าจอขั้นต่ำ	ความหมาย
sm	640px	โทรศัพท์ขนาดใหญ่ขึ้น
md	768px	แท็บเล็ต
lg	1024px	เดสก์ท็อปขนาดกลาง
xl	1280px	เดสก์ท็อปขนาดใหญ่
2xl	1536px	หน้าจอขนาดใหญ่มาก

- สามารถกำหนดหรือเพิ่ม breakpoint เองได้ใน tailwind.config.js

*ตัวอย่างเพิ่ม **breakpoint** ใหม่*

```
module.exports = {
  theme: {
    screens: {
      xs: '475px',  // เพิ่ม breakpoint สำหรับมือถือเล็ก
      sm: '640px',
      md: '768px',
      lg: '1024px',
      xl: '1280px',
      '2xl': '1536px',
```

```
    }
  }
}
```

- การใช้ใน HTML: xs:text-sm md:text-lg lg:text-xl

---

3. Extend

- extend คือการเพิ่มค่าหรือ properties เข้าไปใน theme โดยไม่เขียนทับค่าที่มีอยู่แล้ว
- ใช้ใน tailwind.config.js เพื่อเพิ่มสี, spacing, font size, animation, ฯลฯ

ตัวอย่าง extend theme

```
module.exports = {
  theme: {
    extend: {
      colors: {
        brandBlue: '#1E40AF',
        brandPink: '#DB2777',
      },
      spacing: {
        72: '18rem',
        84: '21rem',
      },
      fontSize: {
        'xxs': '0.65rem',
      },
      borderRadius: {
        'xl': '1rem',
      },
    }
  }
}
```

- ขยายสี, spacing, font size, border radius เพิ่มขึ้น
- ตัวอย่างใช้ class: bg-brandBlue, p-72, text-xxs, rounded-xl

---

สรุป

คำสั่ง	ความหมาย	ตัวอย่างในโค้ด

คำสั่ง	ความหมาย	ตัวอย่างในโค้ด
variants.extend	เพิ่มสถานะพิเศษให้ utility classes	hover:bg-blue-500, active:scale-95
theme.screens	กำหนด breakpoint สำหรับ responsive	md:text-lg, lg:grid-cols-4
theme.extend	ขยาย theme ไม่แทนที่ของเดิม	เพิ่มสี, spacing, font size, animation

---

นี่คือตัวอย่างโปรแกรม 3 ตัวอย่าง (พื้นฐาน) และ 3 ตัวอย่าง (แนวประยุกต์) ที่ใช้การกำหนด **variants, screens, extend** ใน Tailwind CSS พร้อมโครงสร้างไฟล์ คำอธิบายโค้ด และผลการรัน

ตัวอย่างพื้นฐาน (3 โปรแกรม)

ตัวอย่างที่ 1: เพิ่ม Variant active กับ focus-visible

โครงสร้างไฟล์

```
variants-screens-extend/basic-1/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Variants Example</title>
<script src="https://cdn.tailwindcss.com"></script>
<script>
  tailwind.config = {
    variants: {
      extend: {
        backgroundColor: ['active', 'focus-visible']
      }
    }
  }
</script>
</head>
<body class="p-10 bg-gray-50 flex justify-center items-center min-h-screen">
```

```
  <button class="bg-blue-500 text-white px-6 py-3 rounded-md
                 active:bg-blue-700 focus-visible:ring-4 focus-visible:ring-blue-300
                 transition">
    Click Me
  </button>
</body>
</html>
```

คำอธิบายโค้ด

- ขยาย variants เพื่อรองรับ active และ focus-visible กับ backgroundColor
- ปุ่มจะเปลี่ยนสีเมื่อถูกกด (active) และแสดงวงแหวนเมื่อโฟกัส (focus-visible)
- ใช้ transition เพื่อความนุ่มนวล

ตัวอย่างที่ 2: เพิ่ม breakpoint ใหม่ xs และ responsive typography

โครงสร้างไฟล์

```
variants-screens-extend/basic-2/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en" class="scroll-smooth">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Screens Example</title>
<script src="https://cdn.tailwindcss.com"></script>
<script>
  tailwind.config = {
    theme: {
      screens: {
        xs: '475px',
        sm: '640px',
        md: '768px',
        lg: '1024px',
        xl: '1280px',
        '2xl': '1536px',
```

```
      },
    },
  }
</script>
</head>
<body class="p-8 bg-gray-100">
  <h1 class="text-base xs:text-lg sm:text-xl md:text-2xl lg:text-3xl xl:text-4xl font-bold mb-4">
    Responsive Heading with custom xs breakpoint
  </h1>
  <p class="text-sm xs:text-base sm:text-lg md:text-xl leading-relaxed text-gray-700">
    This text resizes based on the screen width including a custom xs breakpoint.
  </p>
</body>
</html>
```

คำอธิบายโค้ด

- กำหนด breakpoint xs ที่ 475px เพิ่มเข้ามา
- ขนาดตัวอักษรปรับตามขนาดหน้าจอ
- ทำให้ responsive ได้ละเอียดมากขึ้น

ตัวอย่างที่ 3: Extend theme เพิ่มสีและ spacing

โครงสร้างไฟล์

```
variants-screens-extend/basic-3/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Extend Theme Example</title>
<script src="https://cdn.tailwindcss.com"></script>
<script>
  tailwind.config = {
    theme: {
```

```
      extend: {
        colors: {
          brand: '#1E40AF',
          'brand-light': '#3B82F6',
        },
        spacing: {
          72: '18rem',
          84: '21rem',
        },
      },
    },
  }
</script>
</head>
<body class="bg-gray-50 p-10">
  <div class="max-w-3xl mx-auto bg-brand-light p-8 rounded-lg">
    <h1 class="text-white text-4xl mb-6">Brand Section</h1>
    <p class="text-white mb-4">This section uses custom colors and spacing extended in
Tailwind config.</p>
    <div class="bg-brand h-72 w-full rounded-md"></div>
  </div>
</body>
</html>
```

คำอธิบายโค้ด

- เพิ่มสี brand และ brand-light เข้าไปในระบบสี
- เพิ่มค่า spacing ขนาด 72 (18rem) และ 84 (21rem)
- ใช้สีและ spacing ใหม่ในองค์ประกอบ

ตัวอย่างแนวประยุกต์ (3 โปรแกรม)

ตัวอย่างที่ 4: Responsive Card with custom breakpoints & active variant

โครงสร้างไฟล์

```
variants-screens-extend/apply-1/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Responsive Card</title>
<script src="https://cdn.tailwindcss.com"></script>
<script>
  tailwind.config = {
    theme: {
      screens: {
        xs: '480px',
        sm: '640px',
        md: '768px',
        lg: '1024px',
        xl: '1280px',
      },
      extend: {
        colors: {
          primary: '#2563EB',
          'primary-dark': '#1D4ED8',
        },
      },
    },
    variants: {
      extend: {
        backgroundColor: ['active'],
        boxShadow: ['active'],
      }
    }
  }
</script>
</head>
```

```
<body class="bg-gray-100 p-8">
  <div class="max-w-5xl mx-auto grid grid-cols-1 xs:grid-cols-2 lg:grid-cols-3 gap-6">
    <div class="bg-white p-6 rounded-lg shadow-md cursor-pointer
                active:bg-primary-dark active:shadow-lg transition duration-200">
      <h2 class="text-xl font-semibold mb-2 text-primary">Card 1</h2>
      <p class="text-gray-700">Click and hold to see active background and shadow effect.</p>
    </div>
    <div class="bg-white p-6 rounded-lg shadow-md cursor-pointer
                active:bg-primary-dark active:shadow-lg transition duration-200">
      <h2 class="text-xl font-semibold mb-2 text-primary">Card 2</h2>
      <p class="text-gray-700">Responsive grid adjusts columns on xs, sm, lg breakpoints.</p>
    </div>
    <div class="bg-white p-6 rounded-lg shadow-md cursor-pointer
                active:bg-primary-dark active:shadow-lg transition duration-200">
      <h2 class="text-xl font-semibold mb-2 text-primary">Card 3</h2>
      <p class="text-gray-700">Try resizing the window to see responsiveness.</p>
    </div>
  </div>
</body>
</html>
```

คำอธิบายโค้ด

- เพิ่ม breakpoint xs ที่ 480px
- ขยาย variants ให้ backgroundColor และ boxShadow รองรับ active
- ใช้ active: class เพื่อให้มีเอฟเฟกต์เมื่อคลิกการ์ด
- ใช้ grid แบบ responsive เปลี่ยนจำนวนคอลัมน์ตามขนาดหน้าจอ

ตัวอย่างที่ 5: Navigation Bar with focus-visible & hover variants

โครงสร้างไฟล์

```
variants-screens-extend/apply-2/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Navbar with Variants</title>
<script src="https://cdn.tailwindcss.com"></script>
<script>
  tailwind.config = {
    variants: {
      extend: {
        backgroundColor: ['hover', 'focus-visible'],
        textColor: ['focus-visible'],
      }
    }
  }
</script>
</head>
<body class="bg-gray-50">
  <nav class="flex justify-center space-x-8 p-6 bg-white shadow">
    <a href="#" class="px-3 py-2 rounded-md text-gray-700
                  hover:bg-blue-100 focus-visible:outline-none focus-visible:ring-2 focus-
visible:ring-blue-500
                  focus-visible:text-blue-700 transition">
      Home
    </a>
    <a href="#" class="px-3 py-2 rounded-md text-gray-700
                  hover:bg-blue-100 focus-visible:outline-none focus-visible:ring-2 focus-
visible:ring-blue-500
                  focus-visible:text-blue-700 transition">
      About
    </a>
    <a href="#" class="px-3 py-2 rounded-md text-gray-700
                  hover:bg-blue-100 focus-visible:outline-none focus-visible:ring-2 focus-
visible:ring-blue-500
                  focus-visible:text-blue-700 transition">
      Contact
```

```
    </a>
  </nav>
</body>
</html>
```

คำอธิบายโค้ด

- ขยาย variants ให้รองรับ focus-visible และ hover สำหรับสีพื้นหลังและข้อความ
- ใช้ focus-visible เพื่อช่วยการเข้าถึง (accessibility) แสดงวงแหวนและเปลี่ยนสีเมื่อโฟกัสด้วยคีย์บอร์ด
- แถบนำทางใช้ transition เพื่อความนุ่มนวลเวลาเปลี่ยนสี

ตัวอย่างที่ 6: Custom Button with extended spacing and colors

โครงสร้างไฟล์

```
variants-screens-extend/apply-3/
└── index.html
```

ไฟล์ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Custom Button</title>
<script src="https://cdn.tailwindcss.com"></script>
<script>
  tailwind.config = {
    theme: {
      extend: {
        colors: {
          btnGreen: '#10B981',
          btnGreenDark: '#047857',
        },
        spacing: {
          72: '18rem',
          80: '20rem',
        },
```

```
        borderRadius: {
          'xl': '1rem',
        }
      }
    },
    variants: {
      extend: {
        backgroundColor: ['active', 'focus-visible'],
        boxShadow: ['active', 'focus-visible'],
      }
    }
  }
</script>
</head>
<body class="bg-gray-100 p-10 flex justify-center items-center min-h-screen">
  <button class="bg-btnGreen text-white px-8 py-4 rounded-xl shadow-md
                 active:bg-btnGreenDark focus-visible:ring-4 focus-visible:ring-btnGreenDark
                 transition duration-300">
    Custom Button
  </button>
</body>
</html>
```

คำอธิบายโค้ด

- ขยายสีปุ่ม btnGreen และ btnGreenDark ใน theme
- ขยาย spacing และ borderRadius
- เพิ่ม variants สำหรับ active และ focus-visible เพื่อให้ปุ่มตอบสนองกับสถานะต่าง ๆ
- มี transition ทำให้การเปลี่ยนสถานะดูนุ่มนวล

กำหนด Global Design Token ผ่าน theme ใน Tailwind CSS

---

1. Design Token คืออะไร?

- Design Token คือค่าที่ใช้กำหนดลักษณะการออกแบบหลักของระบบ เช่น สี (colors), ระยะห่าง (spacing), ขนาดตัวอักษร (font size), ความโค้งมุม (border radius), เงา (shadow) เป็นต้น

- เป็นแหล่งข้อมูลกลางสำหรับการกำหนดสไตล์ เพื่อให้สอดคล้องและง่ายต่อการดูแลรักษาในโปรเจกต์ใหญ่ ๆ

---

2. การกำหนด Global Design Token ใน Tailwind

- Tailwind CSS อนุญาตให้เรากำหนด design token ผ่าน theme ในไฟล์ tailwind.config.js
- โดยเราสามารถกำหนดค่าต่าง ๆ เช่น

```
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: '#1D4ED8',
        secondary: '#9333EA',
        accent: '#F59E0B',
        neutral: '#374151',
        'base-100': '#FFFFFF',
      },
      spacing: {
        'sm': '8px',
        'md': '16px',
        'lg': '24px',
        'xl': '32px',
      },
      fontSize: {
        base: ['16px', '24px'],
        lg: ['18px', '28px'],
        xl: ['20px', '30px'],
      },
      borderRadius: {
        sm: '4px',
        md: '8px',
        lg: '12px',
      },
      boxShadow: {
        sm: '0 1px 2px rgba(0,0,0,0.05)',
```