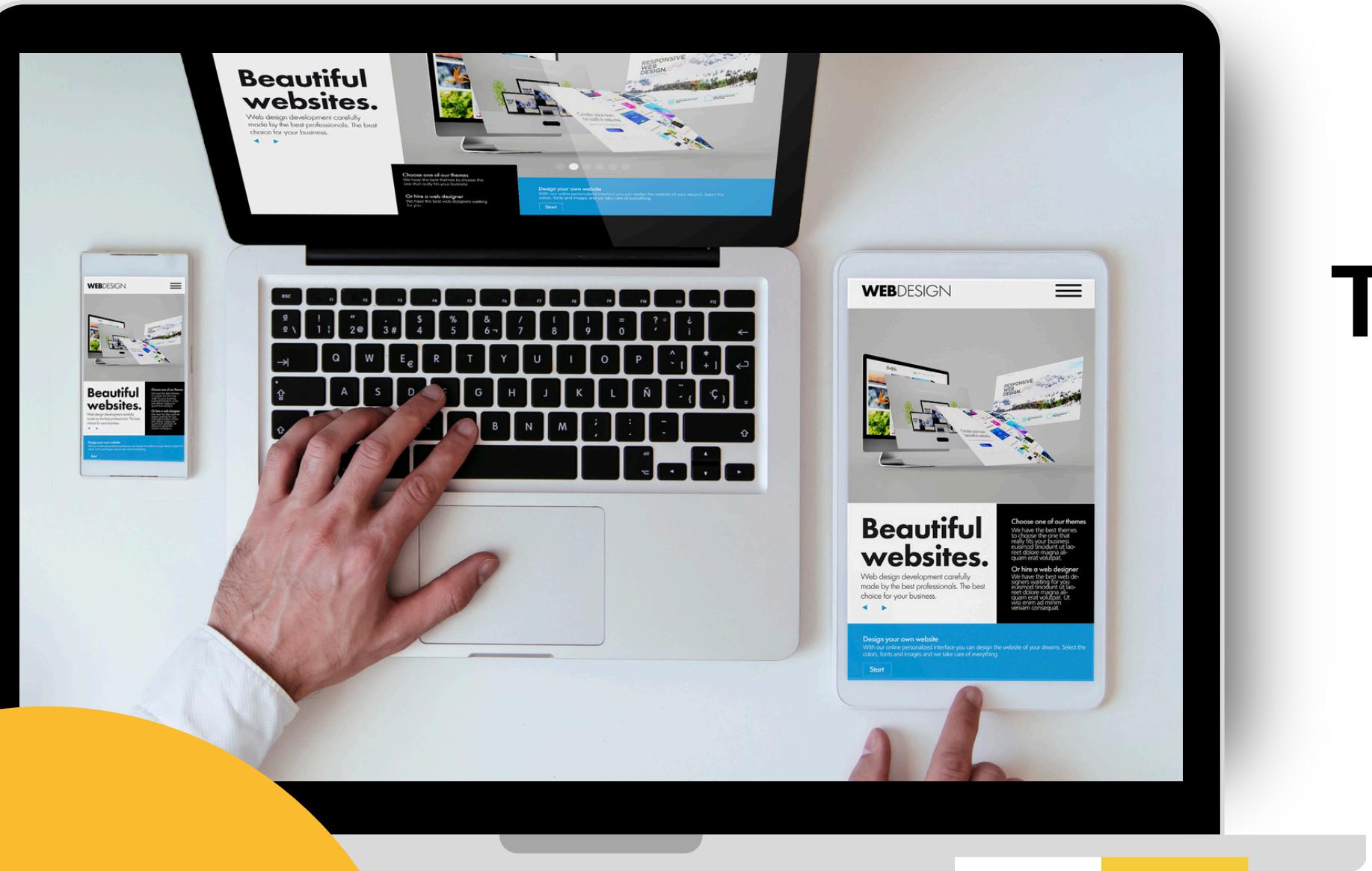


รหัสวิชา: ENGSE611

หน่วยกิต: 3 (1-4-4)

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา



การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

MODERN WEB TECHNOLOGY DEVELOPMENT

Responsive Design

ประเภทวิชา: กลุ่มวิชาชีพเลือก

อาจารย์ผู้สอน

อาจารย์นริศ กำแพงแก้ว



E-mail : naris@rmutl.ac.th



Tel : 091-7915355



รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

หัวข้อ :

การออกแบบเว็บไซต์ แบบ Responsive Responsive Design

วัตถุประสงค์การเรียนรู้ (Learning Objectives)

1. อธิบายความหมายและความสำคัญของ Responsive Design ได้
2. เข้าใจการปรับหน้าเว็บให้เหมาะสมกับขนาดหน้าจอที่แตกต่างกัน
3. ใช้ CSS เป็นต้นเพื่อให้เว็บแสดงผลได้ทั้งบน Desktop และ Mobile
4. ออกแบบหน้าเว็บให้ใช้งานได้ดีบนอุปกรณ์หลายประเภท

ทำไม RESPONSIVE DESIGN ถึงสำคัญ? บริบทปัจจุบัน

ผู้ใช้เข้าถึงเว็บไซต์ผ่าน อุปกรณ์หลากหลาย

- สมาร์ทโฟน
- แท็บเล็ต
- คอมพิวเตอร์
- จอขนาดใหญ่



มากกว่า 60% ของการเข้าชมเว็บไซต์มาจาก โทรศัพท์มือถือ

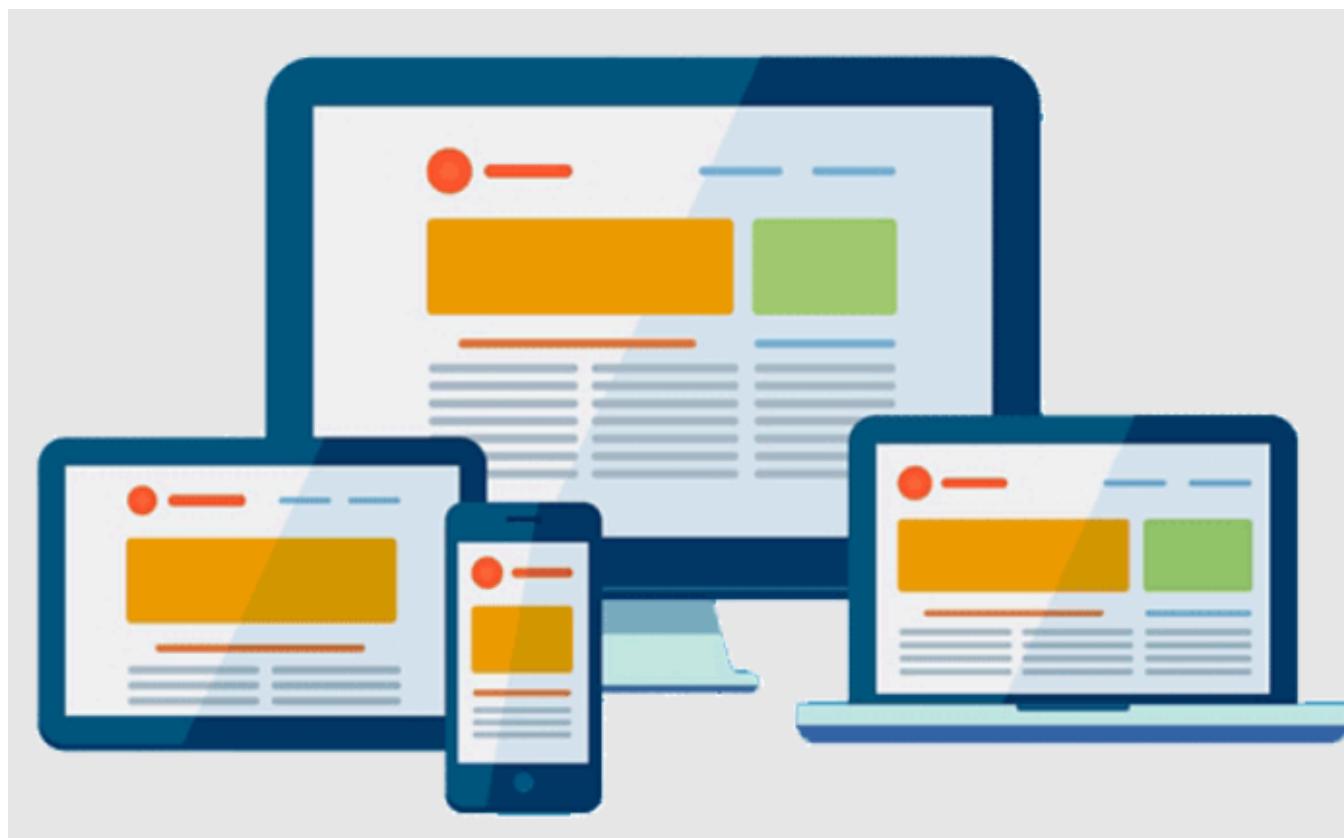
แนวคิดสำคัญของ Responsive

- หน้าเว็บสามารถปรับขนาดและรูปแบบตามความกว้างหน้าจอ
- ผู้ใช้ไม่ต้องซูม ไม่ต้องเลื่อนซ้าย-ขวา
- นักพัฒนาไม่ต้องสร้างเว็บหลายเวอร์ชัน
- เว็บไซต์ใช้งานได้ดีบนทุกอุปกรณ์



ปัญหาของเว็บที่ไม่ Responsive

- ผู้ใช้ต้อง ซูมเข้า-ออกตลอดเวลา
- ปุ่มและเมนู เล็กเกินไป กดยาก
- เกิด Horizontal Scrollbar
- ใช้งานไม่สะดวก
- สร้าง ประสบการณ์ที่ไม่ดีให้ผู้ใช้



ข้อดีของเว็บที่เป็น Responsive

- เนื้อหา อ่านง่ายพอดีจอ
- Layout ปรับตามขนาดหน้าจออัตโนมัติ
- รูปภาพ ไม่ล้าบจอ
- ใช้งานสะดวกทุกอุปกรณ์
- สร้าง ประสบการณ์ที่ดีให้ผู้ใช้ (Good UX)



เทคนิคการออกแบบเว็บไซต์แบบ Responsive

การออกแบบให้เรียบง่าย (Simple Design)

- เน้นความเรียบง่าย ใช้งานง่าย สะดวก (Flat Design)
- ลดความซับซ้อนของเอฟเฟกต์และแอนิเมชันที่ไม่จำเป็น
- ให้ความสำคัญกับโครงสร้าง HTML และ CSS มากกว่าลูกเล่น
- ช่วยให้เว็บไซต์โหลดเร็ว เข้าใจง่าย และใช้งานได้ดีทุกอุปกรณ์

เริ่มต้นจากหน้าจอเล็กที่สุด (Mobile First)

- ออกแบบสำหรับหน้าจอ มือถือ กีอิค่อน และอัปกรณ์จัดแสดงข้อมูลที่มีขนาดเล็ก
- เน้นฟังก์ชันที่จำเป็นต่อผู้ใช้เป็นอันดับแรก
- ตัดองค์ประกอบที่ไม่จำเป็น ออกแบบในหน้าจอขนาดเล็ก
- ช่วยให้เว็บไซต์ตอบโจทย์พฤติกรรมผู้ใช้ในยุคปัจจุบัน

กำหนดขนาดแบบ Relative (Fluid Layout)

- ใช้หน่วยสัมพัทธ์ เช่น % , em , rem แทน px
- กำหนด Layout ยึดหยุ่นตามขนาดหน้าจอ
- กำหนดขนาดรูปภาพแบบ Relative และใช้ max-width
- ป้องกันปัญหาภาพล้ำจอ หรือขยายเกินขนาดจริง

กำหนด Breakpoint และใช้ Media Queries

- ระบุจุดเปลี่ยน (Breakpoint) ตามความกว้างหน้าจอ
- ใช้ CSS Media Queries ปรับ Layout ให้เหมาะสม
- แยกรูปแบบการแสดงผลสำหรับ Mobile, Tablet และ Desktop
- เพิ่มความยืดหยุ่นและความเหมาะสมในการแสดงผลทุกขนาดหน้าจอ

4 ส่วนประกอบหลักของ Responsive Design

1) Viewport Meta Tag

กำหนดขอบเขตและสเกลการแสดงผลของหน้าเว็บให้สอดคล้องกับขนาดหน้าจออุปกรณ์

```
<meta name="viewport" ...>
```

2) Flexible Layouts

ใช้โครงสร้างเลเยอร์แบบยืดหยุ่นด้วยหน่วยสัมพัทธ์ เพื่อให้หน้าเว็บปรับขนาดตามหน้าจอได้

```
width: 100%;
```

3) Flexible Images

กำหนดให้รูปภาพปรับขนาดตามพื้นที่แสดงผลเพื่อป้องกันภาพลับจ่อหรือเสียรูป

```
img { max-width: 100%; }
```

4) Media Queries

ใช้ CSS กำหนดรูปแบบการแสดงผลตามขนาดและลักษณะของอุปกรณ์

```
@media (max-width: 768px) { ... }
```

VIEWPORT META TAG

Viewport คือ พื้นที่ที่ browser ใช้แสดงผลหน้าเว็บบนอุปกรณ์ต่าง ๆ

- Viewport Meta Tag ใช้กำหนดว่า หน้าเว็บควรแสดงผลกว้างแค่ไหน และขยายอย่างไร

ทำไมสำคัญ

ถ้าไม่กำหนด viewport

- เว็บบนมือถือจะถูกย่อให้มีองค์ประกอบน้อยลง
- ขนาดตัวอักษรและ layout แสดงผลถูกสัดส่วน

ผลลัพธ์

- เว็บพอดีกับหน้าจอมือถือกันที่
- ขนาดตัวอักษรและ layout แสดงผลถูกสัดส่วน

CSS

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

width=device-width: สั่งให้ความกว้างของหน้าเว็บเท่ากับความกว้างของหน้าจออุปกรณ์

initial-scale=1.0: กำหนดระดับการซูมเริ่มต้นเป็น 100% (ไม่ซูม)



รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

FLEXIBLE LAYOUTS

Flexible Layouts คือ การออกแบบโครงสร้างหน้าเว็บให้ ยืดหยุ่นตามขนาดหน้าจอ ไม่ใช้ค่าตายตัว เช่น px หากเกินไป

แนวคิดหลัก

- ใช้หน่วยแบบอัตราส่วน เช่น %, vw, rem
- ใช้ระบบ layout สมัยใหม่ เช่น Flexbox และ Grid

ผลลัพธ์

- หน้าเว็บขยาย-ย่อได้ตามหน้าจอ
- ลดปัญหาเนื้อหาลับหรือบีบเกินไป

CSS

```
.container {  
    width: 100%;  
}
```



รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

FLEXIBLE IMAGES

Flexible Images คือ การทำให้รูปภาพ ปรับขนาดตามพื้นที่แสดงผล เพื่อไม่ให้ภาพใหญ่เกินหน้าจอ

ปัญหาที่พบบ่อย

- ภาพลับจອในมือถือ
- ต้องเลื่อนแนวนอนเพื่อดูภาพ

ผลลัพธ์

- ภาพไม่ลับ container
- รักษาสัดส่วนภาพ ไม่ยืดหรือบิดเบี้ยว

CSS

```
img {  
    max-width: 100%;  
    height: auto;  
}
```

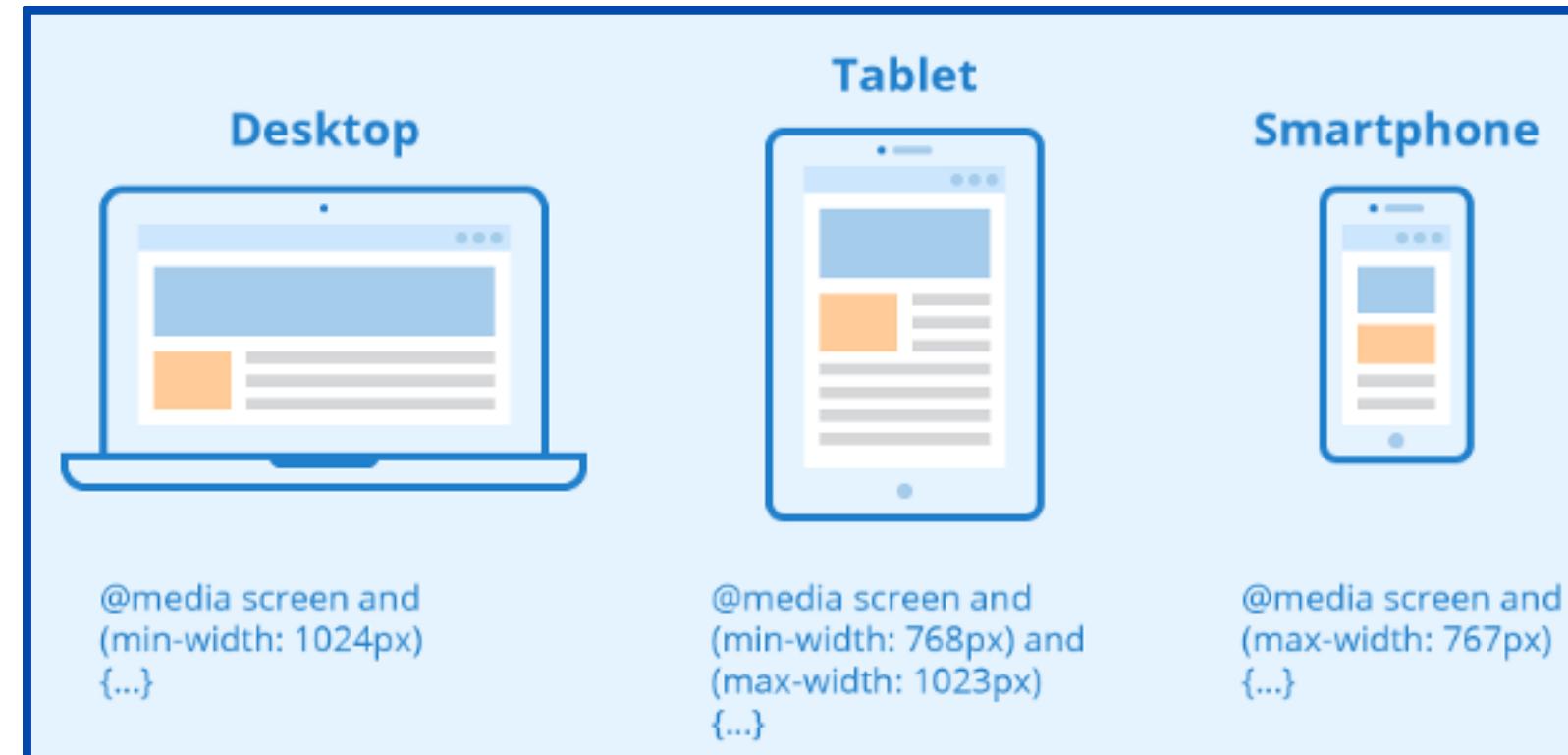


รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

MEDIA QUERIES

Media Queries คือ คำสั่ง CSS ที่ใช้ ตรวจสอบเงื่อนไขของอุปกรณ์ เช่น ขนาดหน้าจอ ความละเอียด หรือแนวตั้ง-แนวนอน



ใช้ทำอะไร

- แยก layout สำหรับมือถือ / แท็บเล็ต / คอมพิวเตอร์
- เปลี่ยนฟอนต์ เม뉴 หรือโครงสร้างตามหน้าจอ

ผลลัพธ์

- หน้าเว็บแสดงผลเหมาะสมกับแต่ละอุปกรณ์
- ประสบการณ์ผู้ใช้ดีขึ้น

CSS

```
@media (max-width: 768px) {  
    .menu {  
        display: none;  
    }  
}
```

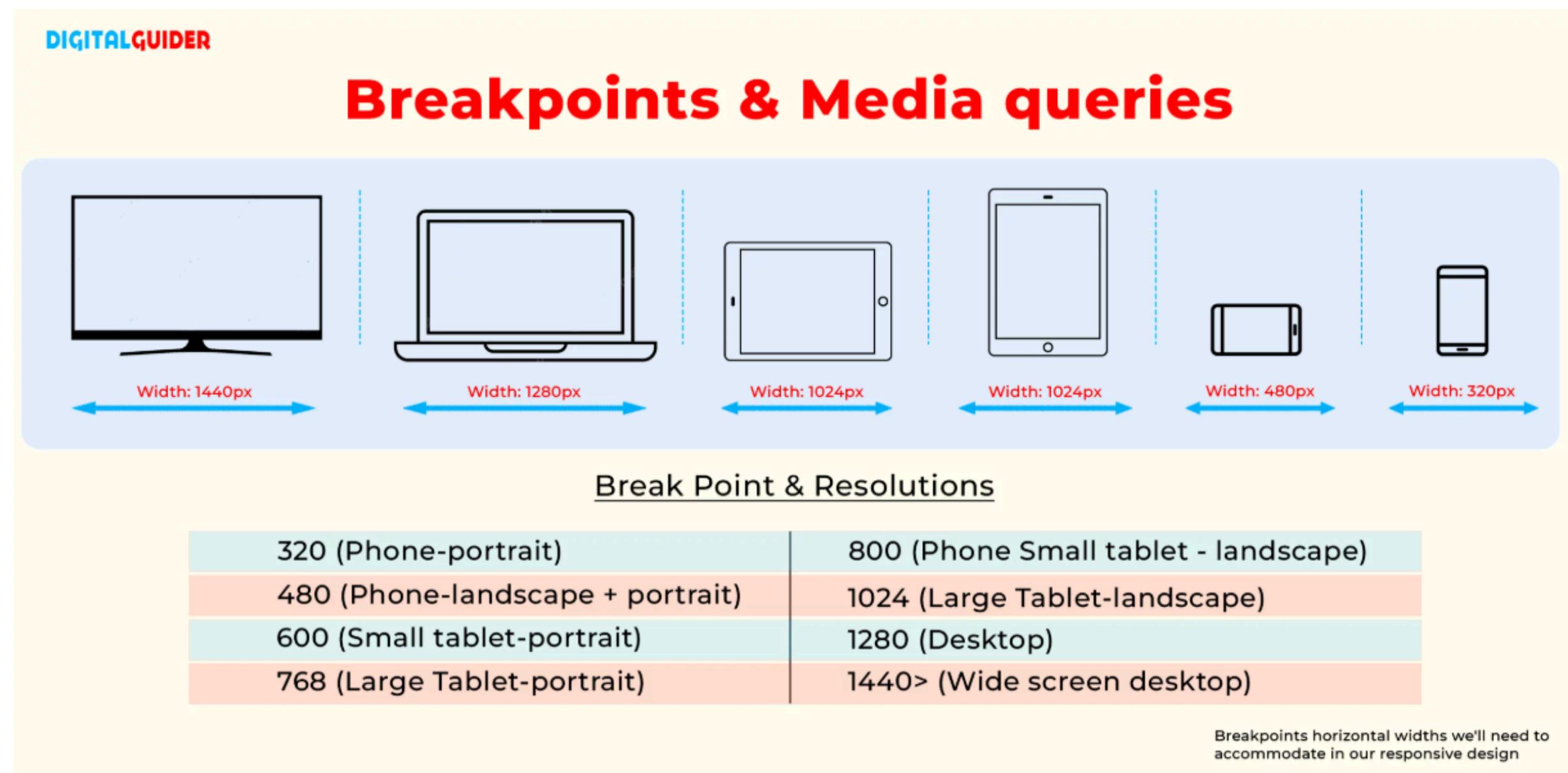
- ใช้ Media Query ตรวจสอบขนาดหน้าจอ
- ถ้าหน้าจอ กว้างไม่เกิน 768px (มือถือ/แท็บเล็ต)
- จะ ซ่อนเมนู .menu ออกจากหน้าเว็บ

รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

BREAKPOINTS มาตรฐาน

คือ “จุดเปลี่ยนขนาดหน้าจอ” ที่เราใช้ Media Query เพื่อปรับรูปแบบเว็บให้เหมาะสมกับอุปกรณ์แต่ละชนิด



เทคนิคสำคัญ: เริ่มเขียนสไตล์สำหรับขนาดเล็กที่สุดก่อนเสมอ (Mobile-First)

MOBILE-FIRST APPROACH

Mobile-First Approach คือ แนวคิดการออกแบบและพัฒนาเว็บไซต์โดยเริ่มต้นจากการออกแบบสำหรับหน้าจอมือถือก่อน จากนั้นจึงค่อย “ขยายความสามารถและรูปแบบการแสดงผล” สำหรับหน้าจอที่ใหญ่ขึ้น เช่น แท็บเล็ต และคอมพิวเตอร์

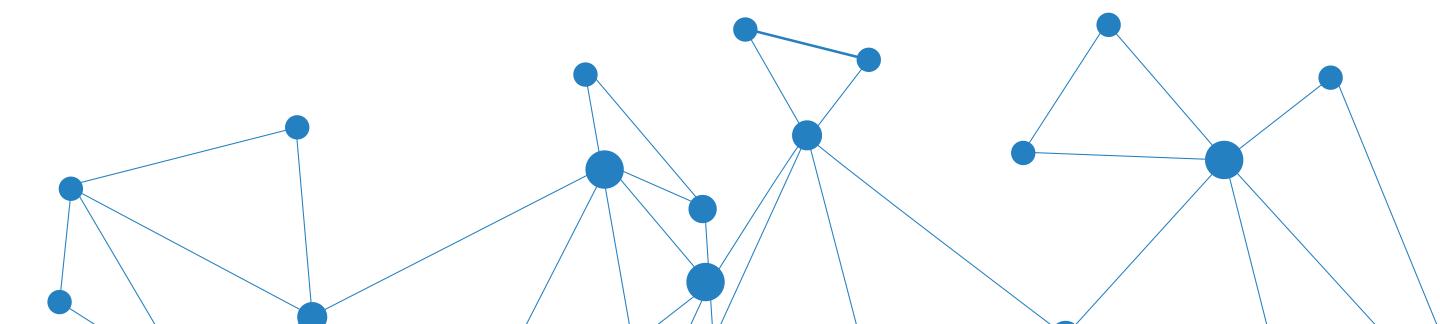
แนวคิดนี้สะท้อนพฤติกรรมผู้ใช้ยุคปัจจุบันที่เข้าถึงเว็บไซต์ผ่านมือถือเป็นหลัก

หลักการทำงานของ Mobile-First

- เขียนสโตร์พื้นฐานให้เหมาะสมกับหน้าจอขนาดเล็กก่อน
- ใช้เงื่อนไขกำหนดขนาดหน้าจอ เพื่อปรับรูปแบบเมื่อจอใหญ่ขึ้น
- เพิ่มรายละเอียดและความซับซ้อนตามลำดับไม่เขียนเกินความจำเป็นตั้งแต่แรก

ผลลัพธ์คือเว็บไซต์

- โหลดเร็วบนมือถือ
- ใช้กรัพยากรณ์น้อย
- แสดงผลได้ดีในทุกอุปกรณ์



รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

MOBILE-FIRST APPROACH

Mobile-First Approach

เริ่มออกแบบจากมือถือ และค่อยขยายสู่จอใหญ่

หน้าแรก
เกี่ยวกับ
บทเรียน
ติดต่อ

แนวคิด Mobile-First

เริ่มออกแบบและเขียนโค้ดสำหรับหน้าจอมือถือก่อน จากนั้นจึงเพิ่มเงื่อนไขสำหรับหน้าจอที่ใหญ่ขึ้น

ข้อดี

- เว็บไซต์โหลดเร็ว
- เหมาะสมกับผู้ใช้มือถือ
- โค้ดเป็นระเบียบ

Mobile-First Approach

เริ่มออกแบบจากมือถือ และค่อยขยายสู่จอใหญ่

หน้าแรก
เกี่ยวกับ
บทเรียน
ติดต่อ

แนวคิด Mobile-First

เริ่มออกแบบและเขียนโค้ดสำหรับหน้าจอมือถือ ก่อนจากนั้นจึงเพิ่มเงื่อนไขสำหรับหน้าจอที่ใหญ่ขึ้น

ข้อดี

- เว็บไซต์โหลดเร็ว
- เหมาะสมกับผู้ใช้มือถือ
- โค้ดเป็นระเบียบ

การใช้งานจริง

ใช้ร่วมกับ Responsive Design และ มาตรฐานของการพัฒนาเว็บยุคปัจจุบัน

รายวิชา ENGSE611 การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

MOBILE-FIRST APPROACH

```
/* =====
 1) สไตล์พื้นฐานของหน้าเว็บ (เริ่มจากมือถือ)
===== */

body {
  margin: 0;
  font-family: Tahoma, sans-serif;
  background: #f5f5f5;
}

header {
  background: #2c3e50;
  color: white;
  text-align: center; /* มือถือ: จัดกลาง */
  padding: 15px;
}

nav {
  display: flex;
  flex-direction: column; /* มือถือ: เมนูเรียงลง */
  gap: 10px;
  margin-top: 10px;
}

nav a {
  background: #34495e;
  color: white;
  text-decoration: none;
  padding: 10px;
  border-radius: 5px;
}
```

1) สไตล์พื้นฐานของหน้าเว็บ (เริ่มจากมือถือ)

แนวคิด : มือถือควรเริ่มจากหน้าตา “สะอาด อ่านง่าย และไม่รก”

- รีเซ็ตระยะขอบเริ่มต้นของเบราว์เซอร์
- เลือกฟอนต์ที่อ่านง่ายบนหน้าจอขนาดเล็ก
- ใช้พื้นหลังสีเรียบ ลดความซับซ้อน

2) ส่วนหัวเว็บไซต์ (Header)

แนวคิด : มือถือมักถือใช้งานมือเดียว การจัดกลางช่วยให้ผู้ใช้รับรู้เนื้อหาได้เร็ว

- จัดข้อความกึ่งกลาง เหมาะกับจอเล็ก
- ใช้ระยะ padding พอดีกับการอ่านและการแตะ
- ไม่เน้นเลยเอ่าต์ซับซ้อนในขั้นแรก

3) เมนูนำทาง

แนวคิด : มือถือ “ไม่เหมา” กับเมนูเรียงแบบบอนยาฯ ๆ

- เมนูเรียงจากบนลงล่าง
- เพิ่มช่องว่างระหว่างปุ่ม ป้องกันการกดผิด
- รองรับการใช้งานด้วยนิ้วมือ

4) ปุ่มเมนู (ลิงก์นำทาง)

- ออกแบบให้ดูเหมือนปุ่ม
- พื้นที่คลิกกว้าง กดง่าย
- มุ่งโถงช่วยให้ดูเป็นมิตรภัยใช้

รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

MOBILE-FIRST APPROACH

```
main {  
    padding: 15px;  
}  
  
.card {  
    background: #ffffff;  
    padding: 15px;  
    margin-bottom: 15px;  
    border-radius: 8px;  
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);  
}  
  
footer {  
    background: #dddddd;  
    text-align: center;  
    padding: 10px;  
    font-size: 14px;  
}
```

5) พื้นที่เนื้อหาหลัก

- มือถือใช้โครงสร้างคอนลัมบ์เดียว
- เนื้อหาเรียงจากบนลงล่าง
- ไม่แบ่งหลายคอนลัมบ์ในขั้นแรก

6) การ์ดเนื้อหา

แนวคิด : Card เป็นรูปแบบที่เหมาะสมกับ Mobile-First มากที่สุด

- แยกเนื้อหาเป็นส่วน ๆ ชัดเจน
- ใช้เงาเล็กน้อยเพื่อแยกชั้นข้อมูล
- อ่านง่ายบนจอเล็ก

7) ส่วนท้ายเว็บไซต์

- แสดงข้อมูลสรุปหรือข้อมูลรายวิชา
- ขนาดตัวอักษรเล็กลงเพื่อไม่เบ่งความสนใจ
- จัดกลางตามแนวคิดมือถือ

MOBILE-FIRST APPROACH

```
/* =====
 2) แท็บเล็ตขึ้นไป
----- */
@media (min-width: 768px) {

  header {
    text-align: left;      /* จ่อให้ญี่ปุ่น: ชิดซ้าย */
    padding: 20px;
  }

  nav {
    flex-direction: row;  /* เมนูเรียงแนวอน */
  }

  main {
    display: grid;        /* เริ่มใช้เลเยอร์เอาต์แบบกริด */
    grid-template-columns: 1fr 1fr;
    gap: 20px;
    padding: 20px;
  }
}
```

1) เงื่อนไขสำหรับแท็บเล็ตขึ้นไป

- ใช้เป็นจุดเปลี่ยนจากมือถือ → แท็บเล็ต
- โค้ดภายในจะทำงานเฉพาะเมื่อหน้าจอกว้างพอ
- ค่าพื้นฐานของมือถืออยังคงทำงานอยู่

2) ปรับส่วนหัวให้เหมาะสมกับจอใหญ่ขึ้น

- จ่อให้ญี่ปุ่นไม่จำเป็นต้องจัดกลางเสมอไป
- การจัดซิดซ้ายทำให้ดูเป็นการการและอ่านง่าย
- เพิ่มระยะ padding เพื่อให้เนื้อหาโปร่งขึ้น

3) เปลี่ยนเมนูจากแนวตั้งเป็นแนวนอน

- เมื่อมีพื้นที่แนวนอนมากขึ้น เมนูควรเรียงซ้าย-ขวา
- ลดการเลื่อนหน้าจอ
- เหมาะสมกับพฤติกรรมการใช้งานแท็บเล็ตและคอมพิวเตอร์

4) เริ่มใช้เลเยอร์เอาต์แบบกริดสำหรับเนื้อหา

- เปลี่ยนจากคอลัมน์เดียว → 2 คอลัมน์
- ใช้พื้นที่จ่อให้คุ้มค่า
- ยังไม่ซับซ้อนเกินไปสำหรับแท็บเล็ต

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

MOBILE-FIRST APPROACH

```
/* =====
  3) คอมพิวเตอร์ขึ้นไป
===== */
@media (min-width: 1024px) {

  body {
    max-width: 1200px;      /* จำกัดความกว้าง */
    margin: 0 auto;         /* จัดกลางหน้าจอ */
  }

  header {
    padding: 30px 40px;
  }

  main {
    grid-template-columns: repeat(3, 1fr); /* 3 คอลัมน์ */
  }
}
```

1) เงื่อนไขสำหรับคอมพิวเตอร์ขึ้นไป

- กำหนดจุดเปลี่ยนสำหรับหน้าจอขนาดใหญ่
- โค้ดภายในจะทำงานเฉพาะเมื่อจอกว้างพอ
- ค่าเริ่มต้นจากมือถือและแท็บเล็ตยังคงทำงานอยู่

2) ควบคุมความกว้างของหน้าเว็บ

แนวคิด : เว็บไซต์ที่กว้างเต็มจอมากเกินไป จะทำให้อ่านยากและดูไม่เป็นระเบียบ

- จำกัดความกว้างสูงสุดของเว็บไซต์
- ป้องกันบรรทัดข้อความยาวเกินไป
- จัดเนื้อหาให้อยู่กึ่งกลางหน้าจอ

3) เพิ่มพื้นที่ส่วนหัวให้เหมาะสมกับจอใหญ่

- จอใหญ่ควรมีพื้นที่หายใจมากขึ้น
- เพิ่มระยะช้ำย-ขวาเพื่อความสมดุล
- ทำให้ส่วนหัวดูโดดเด่นและเป็นมืออาชีพ

4) ขยายเนื้อหาเป็น 3 คอลัมน์

- ใช้พื้นที่แนวนอนให้เกิดประโยชน์สูงสุด
- แสดงเนื้อหาได้มากขึ้นในครั้งเดียว
- เหมาะสมกับการอ่านเชิงเปรียบเทียบหรือหลายหัวข้อพร้อมกัน

รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

MOBILE-FIRST APPROACH

```
<!-- ส่วนหัวเว็บไซต์ -->
<header>
  <h1>Mobile-First Approach</h1>
  <p>เริ่มออกแบบจากมือถือ และค่อยขยายสู่จอยใหญ่</p>

  <!-- เมนูนำทาง -->
  <nav>
    <a href="#">หน้าแรก</a>
    <a href="#">เกี่ยวกับ</a>
    <a href="#">บทเรียน</a>
    <a href="#">ติดต่อ</a>
  </nav>
</header>
```

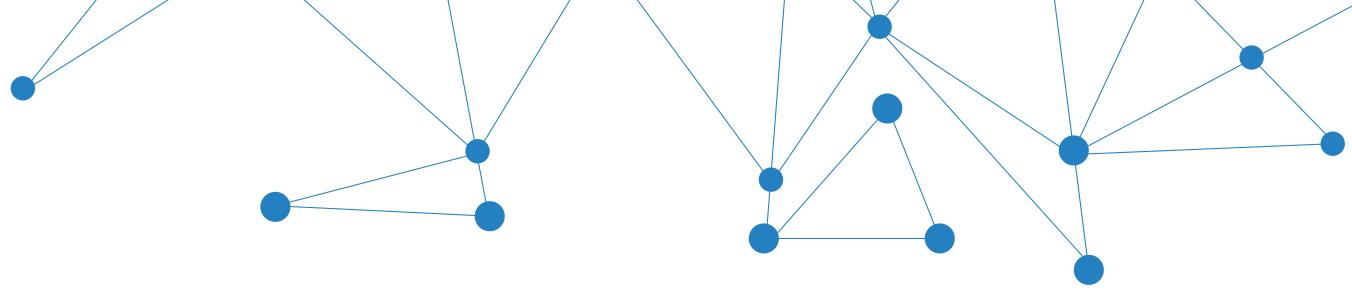
```
<!-- เนื้อหาหลัก -->
<main>
  <div class="card">
    <h3>แนวคิด Mobile-First</h3>
    <p>
      เริ่มออกแบบและเขียนโค้ดสำหรับหน้าจอมือถือก่อน
      จากนั้นจึงเพิ่มเงื่อนไขสำหรับหน้าจอที่ใหญ่ขึ้น
    </p>
  </div>

  <div class="card">
    <h3>ข้อดี</h3>
    <ul>
      <li>เว็บไซต์โหลดเร็ว</li>
      <li>เหมาะสมกับผู้ใช้มือถือ</li>
      <li>โค้ดเป็นระเบียบ</li>
    </ul>
  </div>

  <div class="card">
    <h3>การใช้งานจริง</h3>
    <p>
      ใช้ร่วมกับ Responsive Design
      และเป็นมาตรฐานของการพัฒนาเว็บยุคปัจจุบัน
    </p>
  </div>
</main>
```

```
<!-- ส่วนท้าย -->
<footer>
  รายวิชา ENGSE611 การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่
</footer>
```

HTML CODE



RESPONSIVE IMAGES

Responsive Images คือ การทำให้ รูปภาพปรับขนาดตามพื้นที่หน้าจอโดยอัตโนมัติ เพื่อป้องกันปัญหารูปภาพลับจ่อ บิดเบี้ยว หรือทำให้ผู้ใช้ต้องเลื่อนหน้าจอในแนวนอน

หลักคิดสำคัญคือ รูปภาพต้อง “ยืด–หดได้ตามพื้นที่” แต่ ยังคงสัดส่วนเดิม

CSS

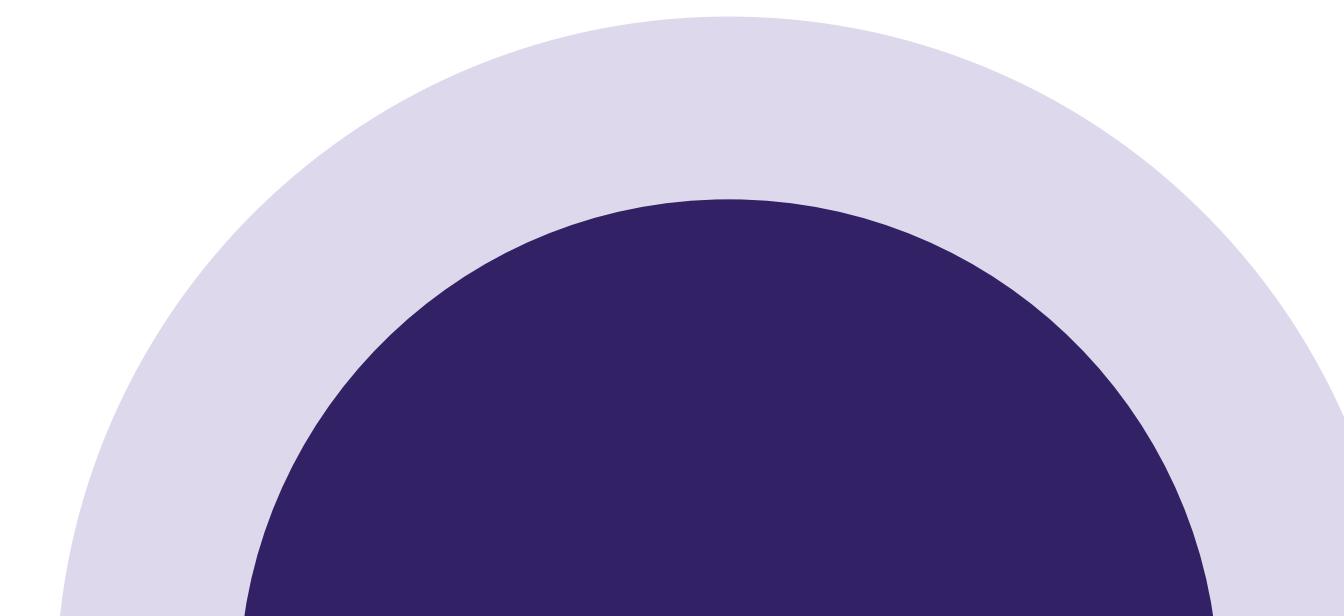
```
img {  
    max-width: 100%;  
    height: auto;  
}
```

บทบาทของ Responsive Images ใน Mobile-First

Responsive Images เป็น ส่วนเสริมที่ขาดไม่ได้ของ Mobile-First เพราะ

- Mobile-First เริ่บจากหน้าจอเล็ก
- รูปภาพมักเป็นสาเหตุหลักของการลับจ่อ
- การจัดการรูปภาพให้ยืดหยุ่นช่วยลดปัญหาทึ่งหมดตั้งแต่ต้น

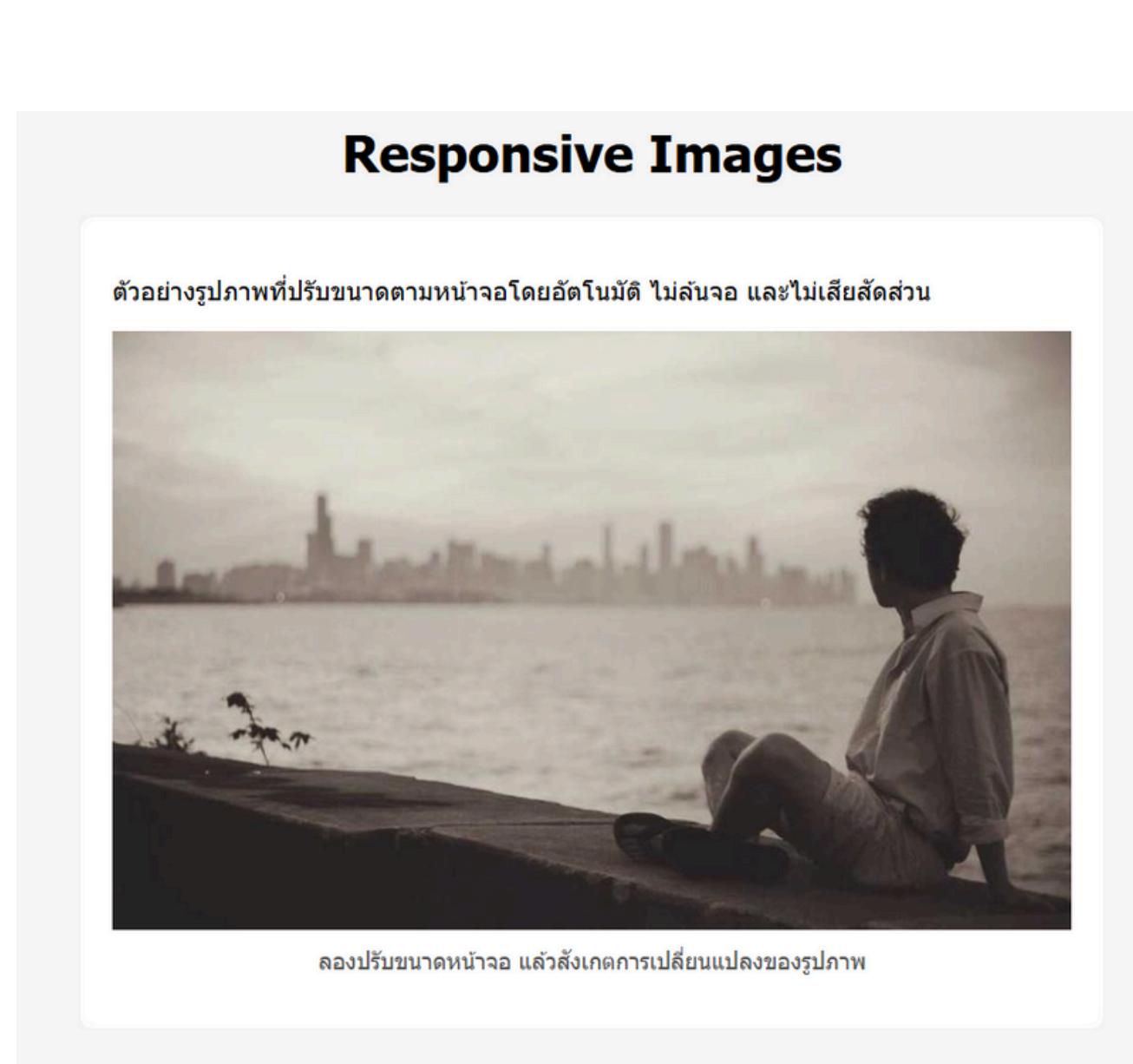
- 1) กำหนดความกว้างสูงสุดของรูปภาพ
- 2) รักษาสัดส่วนของรูปภาพ



รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

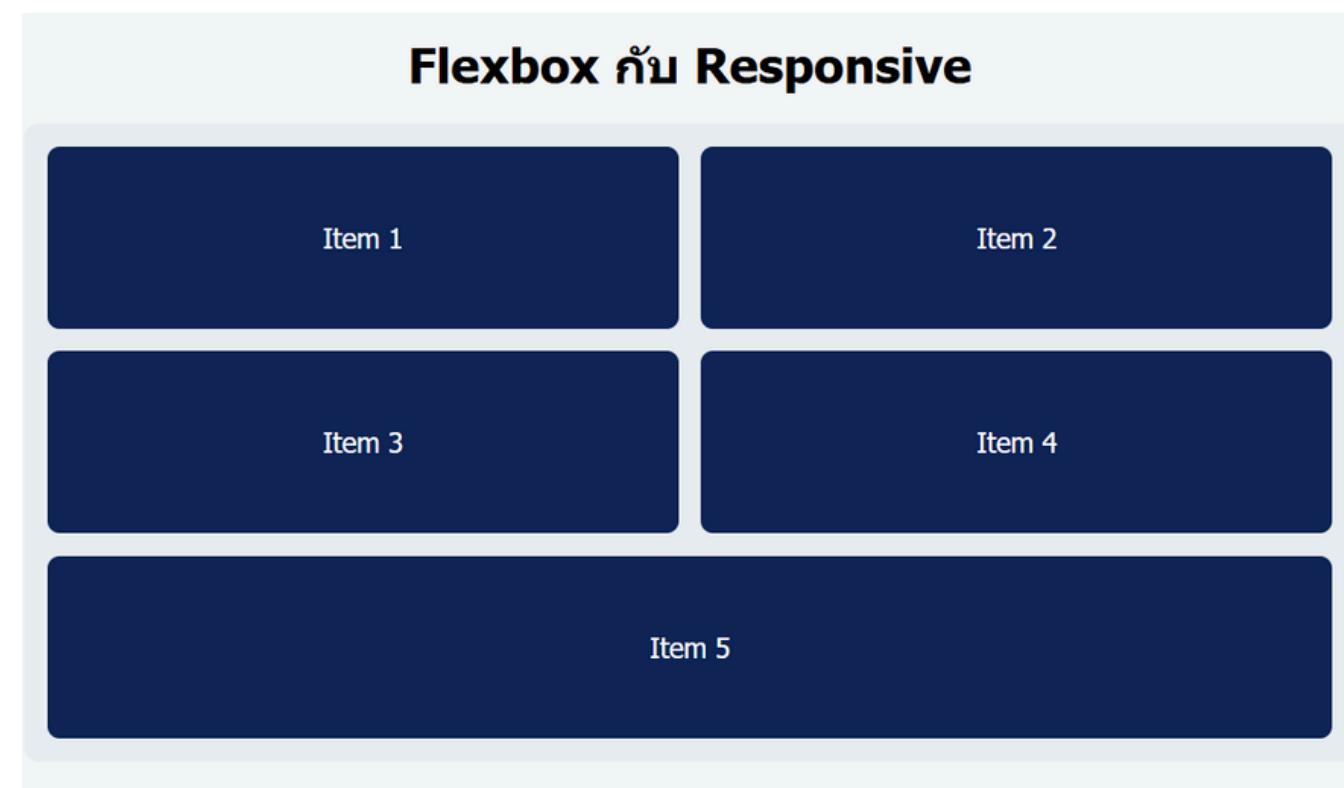
RESPONSIVE IMAGES

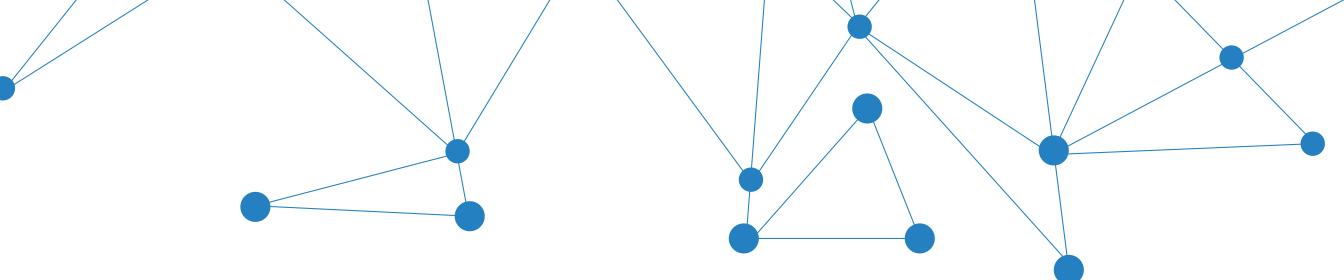


FLEXBOX กับ RESPONSIVE

- Flexbox คือเครื่องมือจัดวางองค์ประกอบในແລ້ວ/ຄອລັນນໍາ
- Responsive คือการปรับการแสดงผลตามขนาดหน้าจอ

เมื่อรวมกัน → เปลี่ยนທີສາກາງແລະການຕັດ
ບຣກັດຂອງກລ່ອງຕາມບົບຖອງປະໂຫຍດ





FLEXBOX กับ RESPONSIVE

```
/* =====
  กล่องแม่ (Flex Container)
  ===== */
.flex-container {
  display: flex;
  flex-direction: column; /* มือถือ: เรียงแนวตั้ง */
  gap: 15px;
  background: #e9edf2;
  padding: 15px;
  border-radius: 10px;
}

/* กล่องลูก */
.item {
  background: #0f2557;
  color: #fffff;
  height: 120px;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 18px;
  border-radius: 8px;
}
```

- • • • • • •
- • • • • • •
- • • • • • •

```
/* =====
  แท็บเล็ตขึ้นไป
  ===== */
@media (min-width: 768px) {
  .flex-container {
    flex-direction: row; /* เรียงแนวอน */
    flex-wrap: wrap; /* อนุญาตให้ตัดบรรทัด */
  }

  .item {
    flex: 1 1 45%; /* 2 กล่องต่อแถว */
  }
}
```

```
/* =====
  คอมพิวเตอร์ขึ้นไป
  ===== */
@media (min-width: 1024px) {
  .flex-container {
    flex-wrap: nowrap; /* ไม่ตัดบรรทัด */
  }

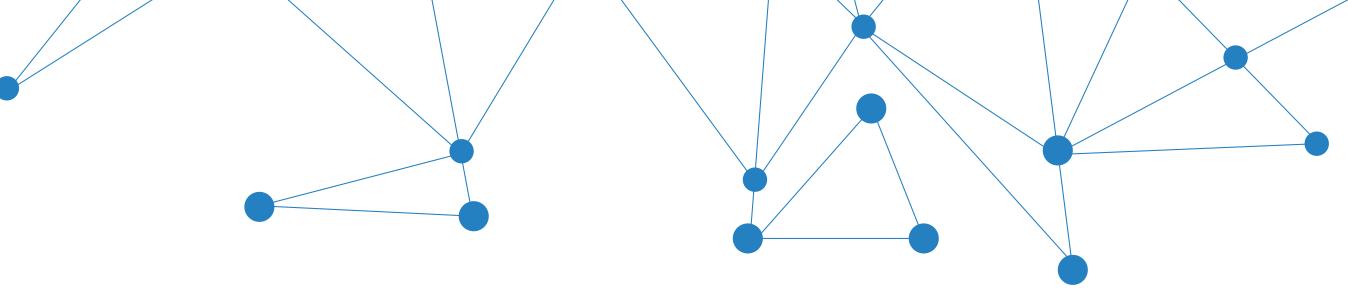
  .item {
    flex: 1; /* ทุกกล่องกว้างเท่ากัน */
  }
}
```

```
<h1>Flexbox กับ Responsive</h1>

<div class="flex-container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

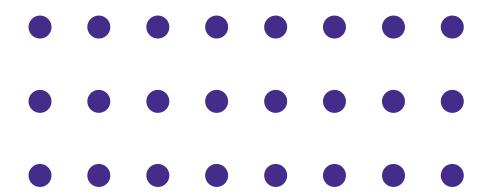
รหัสวิชา: ENGSE611

การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่

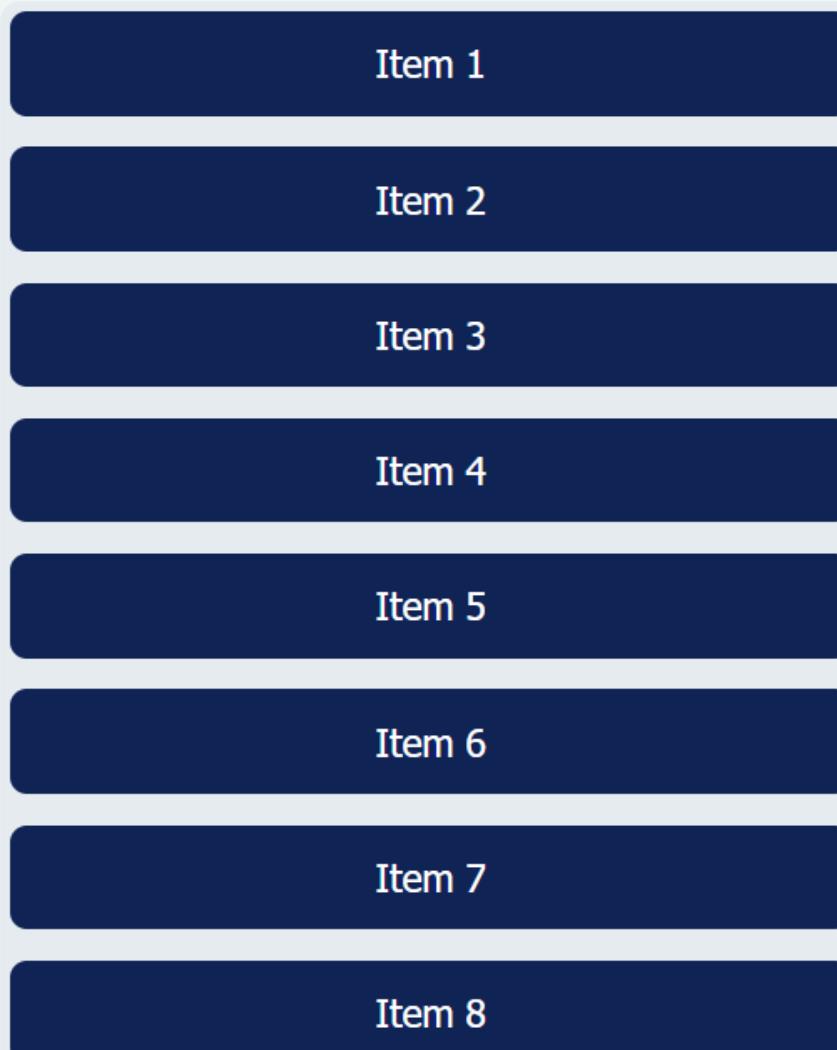


GRID กับ RESPONSIVE

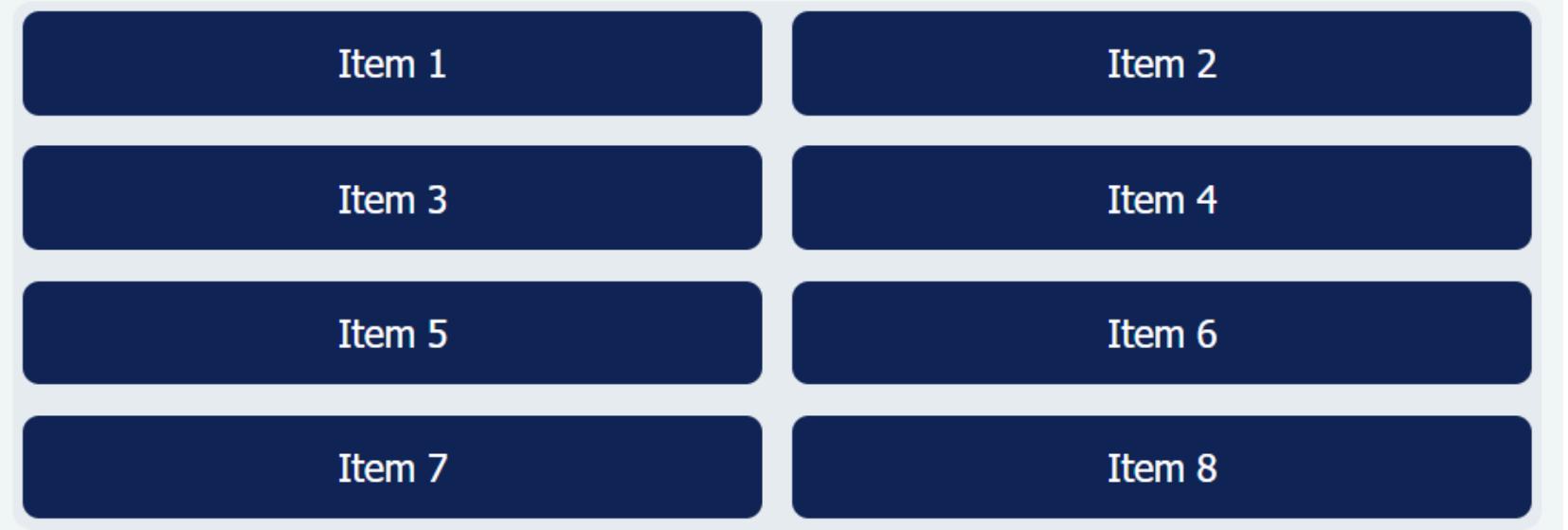
- Flexbox คือ เหมาะกับการจัดレイアウトแบบเป็นโครงสร้าง (ແຄວ-ຄອລັນນີ)
- Responsive คือการปรับการแสดงผลตามขนาดหน้าจอ



Grid Responsive



Grid Responsive

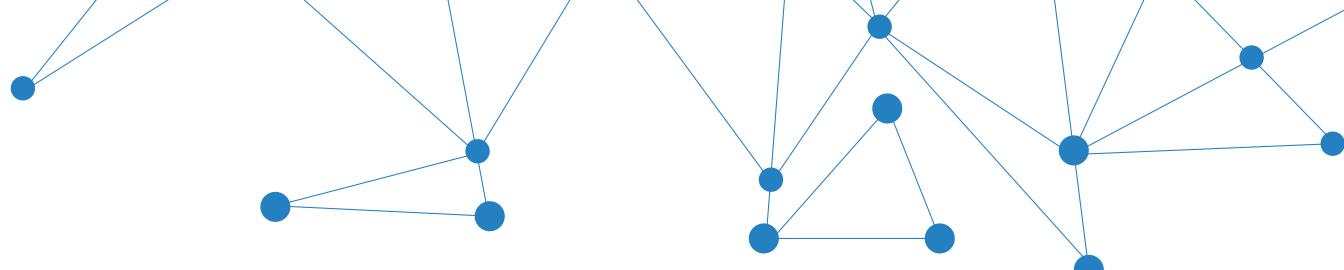


Grid Responsive



รหัสวิชา: ENGSE611

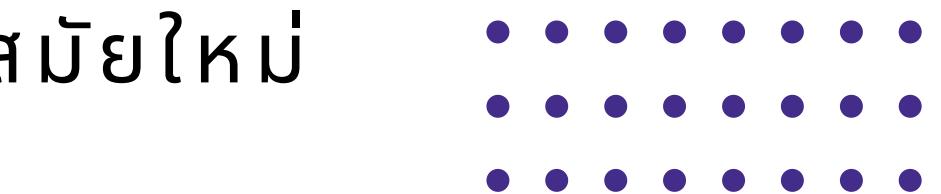
การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่



GRID กับ RESPONSIVE

```
/* =====
   Grid Container (มือถือเป็นหลัก)
   ===== */
.grid-container {
  display: grid;
  grid-template-columns: 1fr; /* มือถือ: 1 คอลัมน์ */
  gap: 15px;
  background: #e9edf2;
  padding: 5px;
  border-radius: 10px;
}

/* กล่องเนื้อหา */
.item {
  background: #0f2557;
  color: #ffffff;
  height: 50px;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 18px;
  border-radius: 8px;
}
```



```
/* =====
   แท็บเล็ตขึ้นไป
   ===== */
@media (min-width: 768px) {
  .grid-container {
    grid-template-columns: repeat(2, 1fr); /* 2 คอลัมน์ */
  }
}

/* =====
   คอมพิวเตอร์ขึ้นไป
   ===== */
@media (min-width: 1024px) {
  .grid-container {
    grid-template-columns: repeat(4, 1fr); /* 4 คอลัมน์ */
    max-width: 1200px;
    margin: 0 auto; /* จัดกลางหน้าจอ */
  }
}
```

```
<h1>Grid Responsive</h1>

<div class="grid-container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
  <div class="item">Item 6</div>
  <div class="item">Item 7</div>
  <div class="item">Item 8</div>
</div>
```

RESPONSIVE TYPOGRAPHY

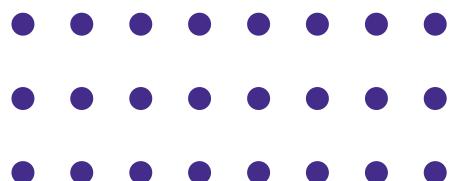
Responsive Typography คือ แนวคิดการ ปรับขนาดตัวอักษรให้เหมาะสมกับขนาดหน้าจอ เพื่อให้ผู้ใช้สามารถอ่านเนื้อหาได้สบายตา ไม่เล็กเกินไป บบมือถือและไม่แน่นหรืออัดอัดเกินไปบนหน้าจอขนาดใหญ่

หลักคิดสำคัญ คือ

- หน้าจอเล็ก → ตัวอักษรเล็กลงอย่างพอดี
- หน้าจอใหญ่ → ตัวอักษรใหญ่ขึ้น อ่านสบายขึ้น

เหตุผลที่ไม่ควรใช้ขนาดตัวอักษรเดียวกันทุกหน้าจอ

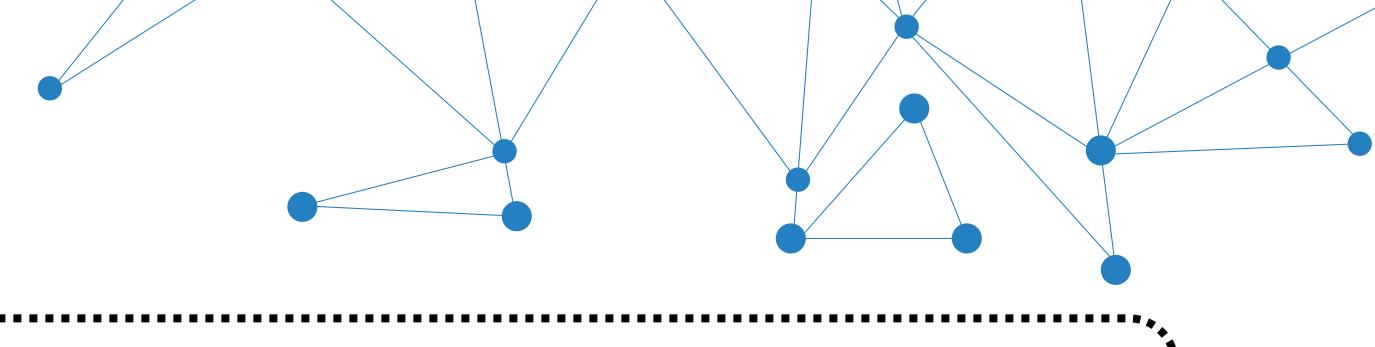
- มือถือ → ตัวอักษรใหญ่เกินไปจะทำให้เลื่อนบ่อย
- เดสก์ท็อป → ตัวอักษรเล็กเกินไปทำให้อ่านยาก
- ผู้ใช้แต่ละอุปกรณ์มี “ระดับการมอง” ต่างกัน



```
/* =====
Mobile First (มือถือ)
===== */
p {
    font-size: 14px;      /* มือถือ */
    line-height: 1.6;
}

/* =====
Tablet ขึ้นไป
=====
*/
@media (min-width: 768px) {
    p {
        font-size: 16px;    /* แท็บเล็ต */
    }
}

/* =====
Desktop ขึ้นไป
=====
*/
@media (min-width: 1024px) {
    p {
        font-size: 18px;    /* คอมพิวเตอร์ */
    }
}
```



CONTAINER และ MAX-WIDTH

แนวคิดของ Container คือ การสร้าง “กล่องกลาง” สำหรับครอบเนื้อหา กันหน้า เพื่อแก้ปัญหา เนื้อหากว้างเกินไปบนจอใหญ่ แต่ยัง ยืดหยุ่นเต็มจอ บบจอเล็ก

หลักคิดสำคัญ คือ

- จ่อเล็ก → ให้เนื้อหาเต็มหน้าจอ
- จ่อใหญ่ → จำกัดความกว้าง และจัดเนื้อหาให้อยู่กึ่งกลาง

CSS

```
.container {  
    width: 100%;  
    max-width: 1200px;  
    margin: 0 auto;  
    padding: 0 20px;  
}
```

Container + Max-Width
คือพื้นฐานของเรียนรู้ เว็บที่อ่านง่ายและดูเป็นมืออาชีพ โดยไม่ต้องใช้ Media Query เลย

- width: 100%
 - ให้ container กว้างเต็มพื้นที่หน้าจอ
 - สำคัญมากสำหรับมือถือ
 - ทำให้เนื้อหาไม่ถูกบีบในจอเล็ก
- max-width: 1200px
 - กำหนดความกว้าง “สูงสุด”
 - เมื่อจอใหญ่กว่า 1200px → container จะไม่ขยายตาม
 - ป้องกันบรรทัดข้อความยาวเกินไป
- margin: 0 auto
 - จัด container ให้อยู่กึ่งกลางหน้าจอ
 - ใช้ได้ก็ต่อเมื่อมีการกำหนดความกว้าง (หรือ max-width)
 - เป็นเทคนิคมาตรฐานของเว็บสมัยใหม่
- padding: 0 20px
 - เว้นขอบซ้าย-ขวาภายใน container
 - ป้องกันข้อความซิดขอบจอเกินไปบนมือถือ
 - ช่วยให้อ่านสบายตาขึ้น

คำามตดสอบความเข้าใจ

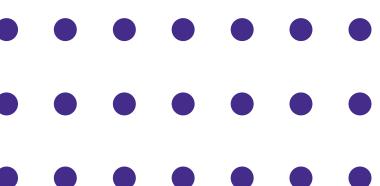
สำหรับรูปภาพที่ต้องการให้ปรับขนาดตามหน้าจอ แต่ไม่ใหญ่เกินขนาดจริงของมัน ควรใช้ CSS property ใด?

A) width: 100%;

B) max-width: 100%;
height: auto;

C) min-width: 100%;

D) width: 100%;
height: 100%;



คำแนะนำดีๆ ในการเขียน CSS

สำหรับรูปภาพที่ต้องการให้ปรับขนาดตามหน้าจอ แต่ไม่ใหญ่เกินขนาดจริงของมัน ควรใช้ CSS property ใด?



B) **max-width: 100%;
height: auto;**

max-width: 100%

- รูปภาพจะ ไม่กว้างเกิน container
- แต่ ไม่ถูกขยายเกินขนาดจริง

height: auto

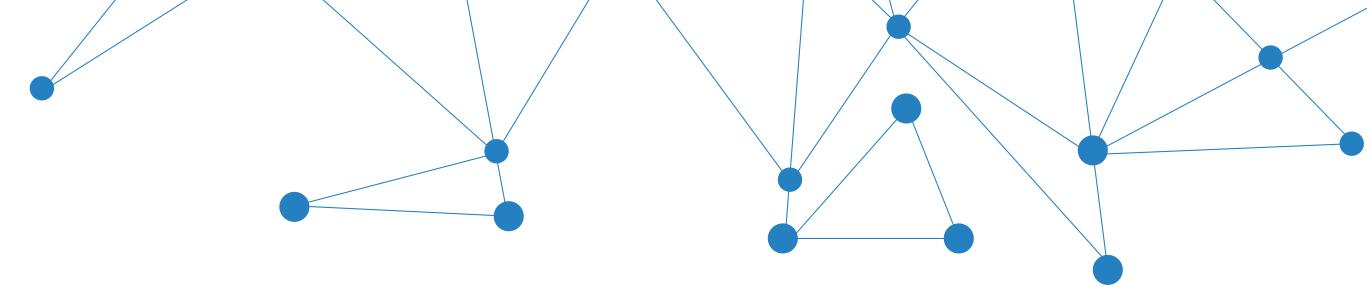
- รักษาสัดส่วนของรูป
- ไม่ยืดหรือบีบภาพ

รูป Responsive ที่ดี ต้อง^{.....}
“ย่อได้ แต่ไม่ขยายเกินจริง^{.....}
และไม่เสียสัดส่วน^{.....}”



หมายเหตุ :

- ข้อ A) width: 100%; รูปจะถูก “บังคับขยาย” ให้เต็ม container และรูปจริงจะเล็ก → ภาพแตกหัก
- ข้อ C) min-width: 100%; รูปจะ “เล็กไม่ได้” → เสี่ยงล้มจอ
- ข้อ D) width: 100%; height: 100%; -> รูปจะถูกยืดกึ่งกว้างและสูง → ภาพเสียสัดส่วน



ข้อผิดพลาดที่เจอบ่อยในการทำ RESPONSIVE WEB

1) ลืม Viewport Meta Tag

ปัญหา : เว็บจะแสดงผลเหมือนเวอร์ชันเดสก์ท็อป บนมือถือ ทำให้ต้องซูมออก ตัวอักษรเล็ก และ Layout เพี้ยบ

สาเหตุ : เบรราว์เซอร์มือถือไม่รู้ว่าควรปรับ สเกลหน้าเว็บตามความกว้างอุปกรณ์

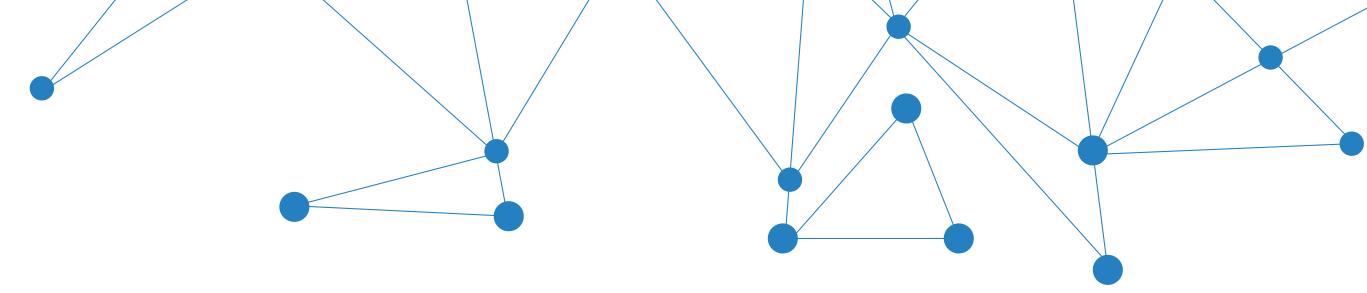
```
<meta name="viewport"  
content="width=device-width,  
initial-scale=1.0">
```

2) ใช้ความกว้างแบบ Fixed Width (px)

ปัญหา : กำหนดความกว้างตายตัว เช่น width: 960px; ทำให้เกิดการเลื่อนแนวนอน (horizontal scroll) บนจอเล็ก

สาเหตุ : ความกว้างตายตัวไม่ยึดหยุ่นตามขนาดหน้าจอ

แนวทางที่ถูกต้อง : ใช้ %, max-width, fr หรือ layout แบบ Flexbox / Grid แทน



ข้อผิดพลาดที่เจอบ่อยในการทำ RESPONSIVE WEB

3) ใช้หน่วย px อย่างเดียว

ปัญหา : ผู้ใช้ที่ต้องการขยายตัวอักษร (เช่น ผู้มีปัญหาทางสายตา) ไม่สามารถปรับได้สะดวก

สาเหตุ : px เป็นหน่วยคงที่ ไม่สัมพันธ์กับการตั้งค่าของผู้ใช้

แนวทางที่ถูกต้อง : ใช้ rem หรือ em เพื่อให้ตัวอักษรยืดหยุ่นและเข้าถึงได้ดีขึ้น

4) ทดสอบแค่บนคอมพิวเตอร์

ปัญหา : เว็บดูปกติบนจอใหญ่ แต่มีปัญหาจิงบนมือถือ เช่น ล็อกจอ ปุ่มเล็ก กดลำบาก

สาเหตุ : ไม่ได้ทดสอบหลายขนาดหน้าจอ

แนวทางที่ถูกต้อง :

- ใช้ Developer Tools (Responsive Mode)
- ทดสอบบนอุปกรณ์จริงอย่างน้อย 1 เครื่อง

Responsive Web ไม่ได้พิสูจน์ตัวเอง แต่พิสูจน์ตัวเอง “ลีมพื้นฐาน” และ “ไม่ทดสอบจริง”

เครื่องมือทดสอบ Responsive

เครื่องมือทดสอบที่นิยม สามารถแบ่งได้เป็น 3 กลุ่มหลัก

1) Browser Developer Tools

เป็นเครื่องมือพื้นฐานที่นักพัฒนาเว็บทุกคนต้องใช้

วิธีใช้งาน

- กด F12 หรือ Ctrl + Shift + I
- คลิกไอคอน Device Toggle (จำลองมือถือ/แท็บเล็ต)

ข้อดี

- เปลี่ยนขนาดหน้าจอได้กันที
- เลือกรุ่นอุปกรณ์ได้หลายแบบ
- ตรวจสอบ CSS, Layout, Media Query ได้แบบเรียลไทม์

ข้อจำกัด

- เป็นการจำลอง ไม่ใช่อุปกรณ์จริง
- บางพฤติกรรมอาจไม่เหมือนการใช้งานจริง 100%

2) เว็บไซต์ทดสอบ Responsive

เช่น เว็บประเภท Responsive Design Checker

ลักษณะการใช้งาน

- ใส่ URL เว็บไซต์ อาทิ ResponsiveDesignChecker.com

ข้อดี

- เห็นภาพรวมหลายอุปกรณ์ในหน้าเดียว
- สะดวกสำหรับตรวจสอบโดยรวม
- เหมาะกับการรีวิวงานเร็ว

ข้อจำกัด

- ไม่สามารถตั้งค่าได้เหมือนของจริง
- ไม่เหมาะกับการทดสอบฟังก์ชันเชิงลึก

3) อุปกรณ์จริง

การทดสอบบนมือถือหรือแท็บเล็ตจริง

เหตุผล

- เห็นประสบการณ์ผู้ใช้จริง
- ตรวจสอบขนาดปุ่ม การสัมผัส การเลื่อน
- เห็นปัญหาที่เครื่องมือจำลองอาจไม่พบ

ข้อจำกัด

- อย่างน้อยควรทดสอบบนมือถือจริง 1 เครื่อง
- โดยเฉพาะก่อนส่งงานหรือขึ้นใช้งานจริง

Responsive Web ที่ดี
ต้อง “เขียนให้ยืดหยุ่น”
และ “ทดสอบให้รอบด้าน”

Performance สำหรับ Mobile

เว็บไซต์บนมือถือ ต้องให้ความสำคัญ กับ ความเร็วเป็นอันดับแรก เพราะผู้ใช้งานอาจใช้อินเทอร์เน็ตที่ช้ากว่า อุปกรณ์ประมวลผลต่ำกว่า และมีความอดทนน้อยกว่าผู้ใช้บนคอมพิวเตอร์

ประเด็นที่ 1: ปรับขนาดรูปภาพให้เหมาะสม

ใช้รูปความละเอียดสูงเกินความจำเป็น ทำให้โหลดช้าโดยไม่จำเป็น

ประเด็นที่ 2: โหลดเฉพาะสิ่งที่จำเป็น (Lazy Loading)

ประเด็นที่ 3: ลดขนาดไฟล์ CSS และ JavaScript

01

- เว็บช้า = ผู้ใช้ปิดหน้าเว็บ
- เว็บเร็ว = ประสบการณ์ดี และใช้งานต่อ

02

- มือถือไม่จำเป็นต้องใช้รูปใหญ่เก่าจocom
- ควรบีบอัดรูปก่อนนำมาใช้
- เลือกขนาดรูปให้เหมาะสมกับการแสดงผลจริง

```

```

ปัญหาที่พบบ่อย

- ไฟล์ CSS/JS ยาว มีช่องว่าง คอมเม้นต์จำนวนมาก
- ส่งผลให้โหลดช้าโดยไม่จำเป็น

แนวคิด : ใช้การบีบอัดไฟล์ (Minify) ลบช่องว่างและโค้ดที่ไม่จำเป็น

Modern CSS Features

1) CSS Variables สำหรับ Responsive

CSS Variables (ตัวแปร CSS) ช่วยให้สามารถกำหนดค่าไว้จุดเดียว และนำไปใช้ซ้ำได้ทั่วไฟล์ และ สามารถเปลี่ยนค่าได้ตามขนาดหน้าจอ

```
:root {  
  -padding: 15px;  
}
```

```
@media (min-width: 768px) {  
  :root {  
    -padding: 30px;  
  }  
}
```

```
.container {  
  padding: var(--padding);  
}
```

- :root คือระดับบนสุดของเอกสาร
- กำหนดตัวแปรชื่อ --padding
- ค่าเริ่มต้นเหมาๆ กับมือถือ

- เมื่อหน้าจอใหญ่ขึ้น (แก้ไขเล็ตขึ้นไป)
- เปลี่ยนค่าตัวแปร --padding
- ทุกจุดที่ใช้ตัวแปรนี้จะเปลี่ยนตามอัตโนมัติ

- นำตัวแปรมาใช้งานจริง
- ไม่ต้องรู้ว่าหน้าจอขนาดเท่าไร
- ค่า padding จะปรับเองตาม breakpoint

2) CSS Clamp Function

clamp() ใช้สำหรับ ปรับขนาดอัตโนมัติตาม viewport โดยไม่ต้องเขียน Media Query หลายช่วง

clamp(ค่าต่ำสุด, ค่าที่ปรับตามหน้าจอ, ค่าสูงสุด)

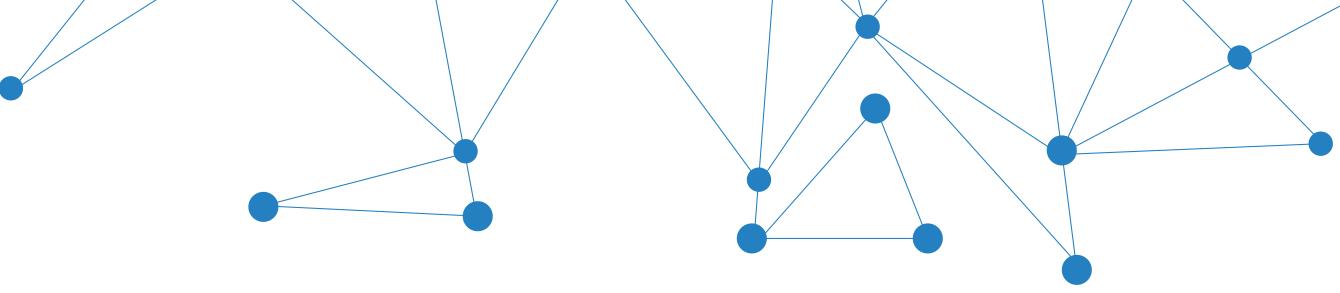
```
h1 {  
  font-size: clamp(1.5rem, 4vw, 2.5rem);  
}
```

- ต่ำสุด: 1.5 rem → เล็กสุดไม่ต่ำกว่านี้
- ปรับตามจอ: 4vw → เปลี่ยนตามความกว้าง viewport
- สูงสุด: 2.5 rem → ใหญ่สุดไม่เกินนี้

ผลลัพธ์ที่ได้

- มือถือ → ตัวอักษรไม่เล็กเกินไป
- จอใหญ่ → ตัวอักษรไม่ใหญ่เกินควบคุม
- ไม่ต้องเขียน Media Query

Modern CSS Features



```
/* =====
CSS Variables (Mobile First)
===== */
:root {
  --padding: 16px;
  --bg-color: #f4f6f8;
  --card-bg: #ffffff;
  --text-color: #1f2933;
}
```

```
/* ปรับค่าเมื่อจอใหญ่ขึ้น */
@media (min-width: 768px) {
  :root {
    --padding: 32px;
  }
}
```

```
body {
  margin: 0;
  font-family: Tahoma, sans-serif;
  background: var(--bg-color);
  color: var(--text-color);
}
```

```
/* =====
Container ใช้ CSS Variable
===== */
.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: var(--padding);
}

.card {
  background: var(--card-bg);
  padding: var(--padding);
  border-radius: 8px;
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
}
```

```
<div class="container">
  <div class="card">
    <h1>Modern CSS Features</h1>

    <p>
      ตัวอย่างนี้แสดงการใช้ CSS Variables และฟังก์ชัน clamp()
      เพื่อทำ Responsive Design โดยไม่ต้องเขียน Media Query จำนวนมาก
    </p>

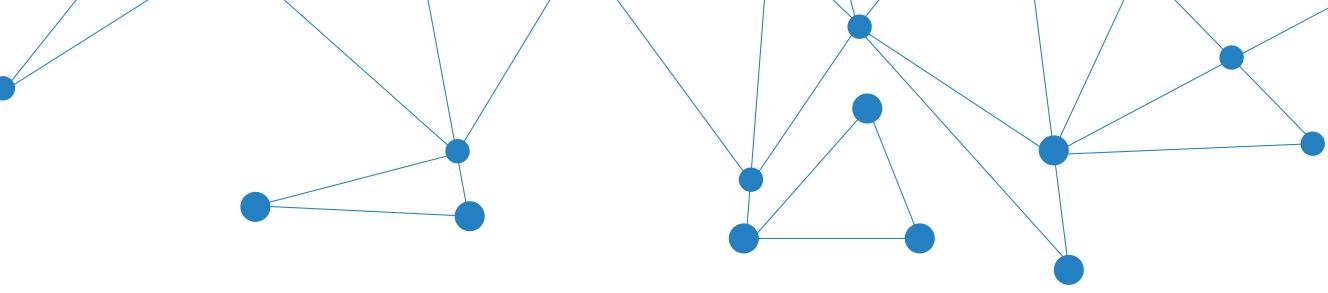
    <p>
      เมื่อขยายหรือย่อหน้าจอ จะเห็นว่าresponsive ของขอบและขนาดตัวอักษร
      ปรับเปลี่ยนอย่างอัตโนมัติและลื่นไหล
    </p>
  </div>
</div>
```

```
/* =====
Clamp สำหรับ Typography
===== */
h1 {
  font-size: clamp(1.6rem, 4vw, 2.8rem);
  margin-top: 0;
}

p {
  font-size: clamp(0.95rem, 2.5vw, 1.1rem);
  line-height: 1.6;
}
```

Best Practices สำหรับ Responsive

Responsive ที่ดี = คิดถูก + เขียนถูก + ทดสอบครบ



1. การวางแผน (Planning)

- ออกแบบ Mobile-First เป็นหลัก
- กำหนด Breakpoints เก่าที่จำเป็น (เปลี่ยน layout จริง)

3. การทดสอบ (Testing)

- ทดสอบทุก breakpoint
- ทดสอบบน อุปกรณ์จริง อย่างน้อย 1 เครื่อง
- ตรวจสอบ: ตัวอักษร, ปุ่มกด, การลับจอ, ความเร็วโหลด

2. การเขียนโค้ด (Implementation)

- ใช้หน่วยแบบยืดหยุ่น (rem, %, vw) แทน px
- เลือกใช้ Flexbox (เรียงแนวเดียว) และ Grid (โครงสร้างหน้า) ให้เหมาะสม
- ใช้ CSS Variables / clamp() ลดโค้ดซ้ำซ้อน (ถ้ามี)

4. Checklist สิ่งก่อนส่งงาน

- ไม่มี horizontal scroll
- ตัวอักษรอ่านง่ายทุกขนาดจอ
- รูปภาพไม่ลับและไม่แตก
- เว็บเปิดเร็วบนมือถือ